Acta Numerica (2026), pp. 001–
doi:10.1017/S09624929

Time parallelization for hyperbolic and parabolic problems

Martin J. Gander, Shu-Lin Wu, Tao Zhou

Time parallelization, also known under the name PinT (Parallel-in-Time) is a new research direction in algorithmic development for the solution of very large scale evolution problems on highly parallel computing architectures. Even though interesting theoretical PinT development can be traced back to 1964, the clock speed reaching its physical limit in 2004 launched PinT to the forefront of research in numerical analysis. Since the only way to increase computing speed is by having more and more cores, the most powerful HPC computing systems have now millions of cores, and even desktop computers have now many cores, both CPUs and also GPUs in their graphics cards. When solving evolution problems on massively parallel systems, parallelization in space often saturates, and the time dimension would be a further direction to parallelize in, leading to PinT. However, the time direction is different for the parallelization compared to the space directions, since information is always going forward in time, and never backward, and so the time evolution process seems to be entirely sequential. Nevertheless, many algorithms have been developed over the last two decades to do PinT computations, and they are often categorized into four classes according to their algorithmic techniques and functioning: shooting-type methods, waveform relaxation methods based on domain decomposition, multigrid methods in space-time, and direct time parallel methods. Many of the most recent and most powerful PinT methods also use combinations of these techniques. We want to follow in this up to date research monograph on PinT methods however a different classification, because over the past few years, the community realized that highly successful PinT algorithms for parabolic problems struggle when applied to hyperbolic problems. We therefore focus here first on explaining fundamental differences between parabolic and hyperbolic problems for time parallelization. We then present four effective PinT methods for hyperbolic problems, namely Schwarz Waveform Relaxation with its relation to Tent Pitching, Parallel Integral Deferred Correction, ParaExp, and ParaDiag. While these methods also work well for parabolic problems, we then present PinT methods especially designed for parabolic problems, like Parareal, the Parallel Full Approximation Scheme in Space-Time, Multigrid Reduction in Time, and Space-Time Multigrid. We

complement our analysis with numerical illustrations for four time-dependent PDEs: the heat equation, the advection-diffusion equation, Burgers' equation, and the second-order wave equation.

CONTENTS

1. Introduction

Time parallelization has been a very active field of research over the past two decades. The reason for this is that hardware development has reached its physical limit for clock speed, and faster computation is only possible using more and more cores. We see this development even in small-scale computing devices like smartphones that have become multicore, and high-performance computers now have millions of cores. Time parallelization methods, also referred to as Parallel in Time (PinT) methods, are methods that allow one to use for evolution problems more cores than when one would only parallelize the in space. Already 60 years ago, Nievergelt proposed in a visionary paper (Nievergelt 1964) such an approach, concluding with:

"*The integration methods introduced in this paper are to be regarded as tentative examples of a much wider class of numerical procedures in which parallelism is introduced at the expense of redundancy of computation. As such, their merits lie not so much in their usefulness as numerical algorithms as in their potential as prototypes of better methods based on the same principle. It is believed that more general and improved versions of these methods will be of great importance when computers capable of executing many computations in parallel become available.*"

Several new methods like these were then developed over the decades that followed Nievergelt, until PinT methods were brought to the forefront of research with the advent of the Parareal algorithm (Lions, Maday and Turinici 2001); see the historical review (Gander 2015), the review focusing on PinT applications (Ong and Schröder 2020), and also the recent research monograph (Gander and Lunet 2024).

Parallelization in time for evolution problems may, at first glance, seem impossible due to the causality principle: solutions at later times are determined by solutions at earlier times, and not vice versa. Evolution problems

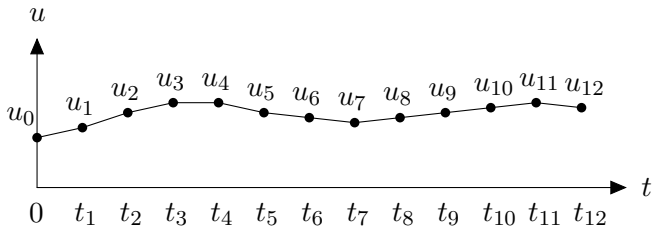

Figure 1.1. Sequential nature of time integration using Forward Euler.

thus have an inherent sequential nature. This becomes clear when we consider a simple ordinary differential equation as our evolution problem, along with its forward Euler discretization,

$$\partial_t u = f(u), \quad u(0) = u_0, \qquad u_{n+1} = u_n + \Delta t f(u_n). \tag{1.1}$$

The recurrence formula of Forward Euler clearly shows that we must know u_n before we can compute u_{n+1}, as illustrated in Figure 1.1. It is not clear for example if one can do useful computational work for the approximations u_{10} to u_{12} before knowing the approximation u_9.

Nevertheless, many new PinT methods have been developed since 2001, and they are often classified into four groups based on the algorithmic techniques used; see (Gander 2015) and also the recent research monograph (Gander and Lunet 2024), and later text for more complete references:

1 Methods based on *Multiple Shooting* going back to the work of (Nievergelt 1964, Bellen and Zennaro 1989, Chartier and Philippe 1993), leading to (Saha, Stadel and Tremaine 1997) and culminating in the *Parareal* algorithm (Lions et al. 2001) and many variants.
2 Methods based on *Domain Decomposition* (Schwarz 1870) and *Waveform Relaxation* (Lelarasmee, Ruehli and Sangiovanni-Vincentelli 1982*a*) that were combined in (Bjørhus 1995), resulting in *Schwarz Waveform Relaxation (SWR)* (Gander, Halpern and Nataf 1999).
3 Methods based on *Multigrid* going back to the parabolic multigrid method (Hackbusch 1984), and developed into fully parallel *Space-Time MultiGrid (STMG)* methods (Gander and Neumüller 2016).
4 *Direct time parallel methods*, which started with parallel time stepping techniques (Miranker and Liniger 1967), and led to the modern *Revisionist Integral Deferred Correction (RIDC)* method (Christlieb, Macdonald and Ong 2010). Currently very successful methods in this class we will see are *ParaExp* (Gander and Güttel 2013), and parallelization by diagonalization (Maday and Rønquist 2008), which led to *ParaDiag* (Gander, Liu, Wu, Yue and Zhou 2021*c*).

The first three groups contain iterative methods, whereas the last one contains non-iterative ones, but the boundaries in this classification are not

strict. A good example are the ParaDiag methods, which were first exclusively direct solvers based on the diagonalization of the time stepping matrix, but then rapidly iterative variants appeared, within WR methods from the second group (Gander and Wu 2019) or within parareal from the first group (Gander and Wu 2020), and approximate ParaDiag methods were also applied as stationary iterations or preconditioners for Krylov methods directly to the all-at once system derived from the space time discretization in the third group (McDonald, Pestana and Wathen 2018, Liu and Wu 2020). Parareal from the first group can also be interpreted as a multigrid method from the third group with aggressive coarsening (Gander and Vandewalle 2007), and in turn MGRiT as a Parareal algorithm with overlap (Gander, Kwok and Zhang 2018*c*).

We would like to adopt a different approach here, however, to classify PinT methods, specifically based on the types of problems that they can effectively solve:

1 Effective PinT methods for hyperbolic problems.
2 PinT methods designed for parabolic problems.

To achieve this, we explain in Section 2 intuitively why there must be a fundamental distinction in PinT methods when solving hyperbolic or parabolic problems, and introduce model problems that will later serve to illustrate this for PinT methods. We then describe effective PinT methods for hyperbolic problems in Section 3, which in general work even better for parabolic problems, and in Section 4, we present PinT methods designed for parabolic problems, which in general struggle when applied to hyperbolic problems. We will draw conclusions in Section 5. The Matlab codes for the numerical results in Sections 2-4 can be obtained from `https://github.com/wushulin/ActaPinT`.

2. Model Problems linking the Parabolic and Hyperbolic world

We will often use as our test problems Partial Differential Equations (PDEs) that allow us to link the parabolic and hyperbolic worlds. A typical example is the linear advection-diffusion equation we will first see in subsection 2.2, which contains both parabolic and hyperbolic components. We will also frequently use the system of ordinary differential equations (ODEs)

$$\begin{array}{rcl} \boldsymbol{u}'(t) & = & A\boldsymbol{u}(t) + \boldsymbol{g}(t),\ t \in (0,T], \\ \boldsymbol{u}(0) & = & \boldsymbol{u}_0, \end{array} \tag{2.1}$$

where $A \in \mathbb{R}^{N_x \times N_x}$ is the discrete matrix arising from semi-discretizing the PDE in space, because many time parallel methods (except the domain decomposition based methods) are described and analyzed for ODEs. A nonlinear important PDE variant of advection-diffusion is the so-called Burgers'

equation that we will first see in subsection 2.3. Similarly to the linear case, in order to discuss time parallel methods in the nonlinear setting, we will use the nonlinear system of ODEs

$$\begin{aligned} \boldsymbol{u}'(t) &= f(\boldsymbol{u}(t), t),\ t \in (0, T], \\ \boldsymbol{u}(0) &= \boldsymbol{u}_0, \end{aligned} \tag{2.2}$$

where $f : \mathbb{R}^{N_x} \times \mathbb{R} \to \mathbb{R}^{N_x}$ depends on its first variable in a nonlinear manner, such as $f(\boldsymbol{u}(t), t) = A\boldsymbol{u}(t) + B\boldsymbol{u}^2(t) + \boldsymbol{g}(t)$ for Burgers' equation.

We introduce however also simpler equations that are either of parabolic or hyperbolic nature, like the heat equation and the second order wave equation. To avoid complicated notation, we will only consider model problems in one spatial dimension on the unit interval $\Omega = (0, 1)$, which is not really a restriction, since the applicability and convergence properties of PinT methods do in general not depend on the space dimension.

2.1. Heat equation

As parabolic model problem, we consider the one dimensional heat equation,

$$\partial_t u(x, t) = \partial_{xx} u(x, t) + g(x, t) \quad \text{in } \Omega \times (0, T], \tag{2.3}$$

with initial value $u(x, 0) = u_0(x)$ and either homogeneous Dirichlet or Neumann boundary conditions. An example solution with $u_0(x) = 0$ and a forcing function that heats at four different time instances $t_1 = 0.1, t_2 = 0.6$, $t_3 = 1.35$ and $t_4 = 1.85$ in the middle of the space domain $\Omega = (0, 1)$,

$$g(x, t) = 10 \sum_{j=1}^{4} \exp(-\sigma[(t - t_j)^2 + (x - 0.5)^2]), \tag{2.4}$$

is shown for $\sigma = 200$ in Figure 2.2 in the first three panels, for homogeneous Dirichlet, Neumann, and also periodic boundary conditions. We observe that with Dirichlet conditions, the solution does not propagate far in time, and thus we can compute for example the solution for $t \in (1.7, 2.2)$ for the fourth source term independently from the solution at earlier times! This is a prime example where, despite the causality principle, we can perform useful computations for later time instances before knowing the earlier ones. This concept can be naturally understood from daily life experience: it is straightforward to predict the temperature in your living room in winter a week or a month in advance; you simply need to know if the heater will be on and the windows closed.

However, this changes significantly in our simple model problem when Neumann conditions are applied, as shown in the second plot of Figure 2.2. Here, the solution for $t \in (1.7, 2.2)$ is influenced by the first, second and third source term at earlier times, since heat is now accumulating, nicely illustrating the causality principle. This scenario corresponds to a perfectly insulated room where heat cannot escape, and in this situation it is crucial to

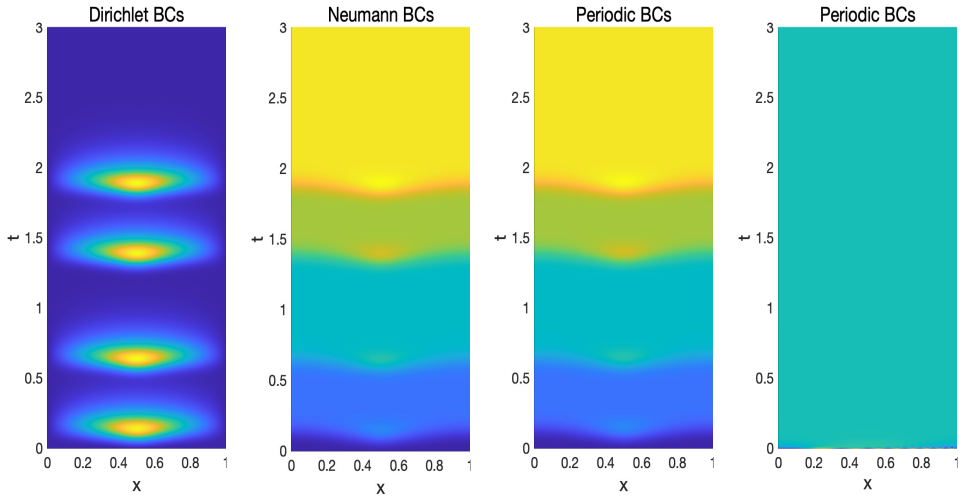

Figure 2.2. Heat equation with homogeneous Dirichlet, Neumann, and periodic boundary conditions (first three panels), using the source term (2.4) and zero initial condition $u_0(x) = 0$. In the last panel we still use periodic boundary conditions, but a zero source term, with initial condition $u_0(x) = \sin^2(8\pi(1-x)^2)$.

know how frequently or for how long the heating was on, since this heat will stay forever in the perfectly insulated room. Note however that in practice it is difficult to have a perfectly insulated room, and heat will always eventually escape, which one would model with a Robin boundary condition.

The situation in the third panel of Figure 2.2 with periodic boundary conditions is similar to the case with Neumann boundary conditions in the second panel, the solution for $t \in (1.7, 2.2)$ is also influenced by the first, second and third source term at earlier times, and with periodic conditions, heat can never escape.

In the fourth panel of Figure 2.2 we show a solution with zero source term and periodic boundary conditions, but now imposing an initial condition with a precise, oscillating signal, namely $u_0(x) = \sin^2(8\pi(1-x)^2)$. We see that the only information left from this signal after a very short time already is a constant, about the same constant as from the first two source terms in the second and third panel of Figure 2.2.

In spite of the causality principle, time parallelization and thus PinT computations for a heat equation, and also more general parabolic problems, should thus be rather easily possible in the case of Dirichlet boundary conditions, since then the solutions are completely local in time (Gander, Ohlberger and Rave 2024), as it is the case in space with solvation models in computational chemistry, see (Ciaramella and Gander 2017, Ciaramella and

Gander 2018*a*, Ciaramella and Gander 2018*b*). With Neumann or periodic boundary conditions, it should still be well possible to do PinT computations, provided one can propagate low frequency solution components, like the constant in our example, effectively over long time, using a coarse grid for example.

2.2. Advection-diffusion equation

We now consider the advection-diffusion equation with homogeneous Dirichlet and periodic boundary conditions[1] on the unit domain $\Omega = (0, 1)$,

$$\partial_t u(x,t) + \partial_x u(x,t) - \nu \partial_{xx} u(x,t) = g(x,t) \quad \text{in } \Omega \times (0,T], \tag{2.5}$$

with initial condition $u(x, 0) = u_0(x)$, where $\nu > 0$ is the diffusion parameter. We show in Figure 2.3 in the top row the solution obtained with zero Dirichlet boundary conditions, and in the bottom row with periodic boundary conditions. In the first three panels, we use a zero initial condition, $u_0(x) = 0$ and the same source term (2.4) we had used for the heat equation for three different values of the diffusion parameter, $\nu = 1, 10^{-2}$ and 5×10^{-4}. We see that when ν is large, then the diffusion part dominates and the solution has similar properties as the solution of the heat equation. If ν is small however, i.e., the advection part plays a dominant role, then the solution is transported from left to right over much longer time, as we see in the top middle two panels in Figure 2.3. In the top right panel, we use a zero source term, but a non-zero initial condition $u_0(x) = \sin^2(8\pi(1-x)^2)$ and again the small diffusion parameter 5×10^{-4}. We see that now all the fine features present in the high frequency components of the initial condition are transported far in time. Nevertheless, for both ν large and ν small, we can still compute the solution for $t \in (1.25, 2.5)$ before we obtain the solution earlier in time, because all solution components eventually are diffused or leave the domain.

For periodic boundary conditions however, we see in Figure 2.3 in the bottom row that the advection-diffusion equation transports information over long time: in the first panel with large diffusion this information is only low frequency, a constant, like for the heat equation, and PinT computations are still possible if one has a way of transporting coarse solution components far in time, using a coarse grid for example. In the next two panels however, we see that when the diffusion parameter becomes small, more and more fine information is transported very far in time, and for successful PinT computation there must be a mechanism to propagate this information effectively far in time. The last panel without source and just a non-zero initial condition shows that for small diffusion, a lot of fine, high frequency

[1] We would not learn anything new with Neumann conditions.

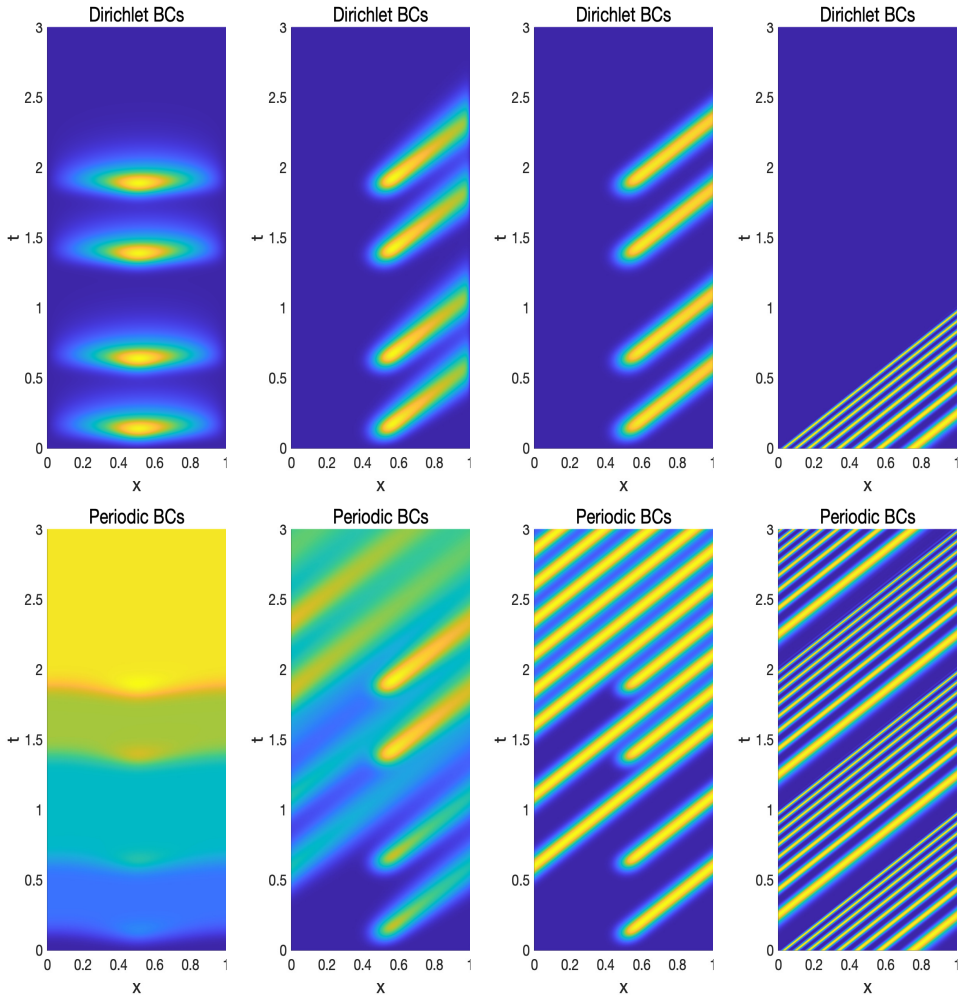

Figure 2.3. Advection-diffusion equation with zero Dirichlet boundary conditions (top row) and periodic boundary conditions (bottom row). In the first three pannels, we use a zero initial condition, $u_0(x) = 0$, and the same source term as in Figure 2.2 for the heat equation, and a smaller and smaller diffusion parameter $\nu = 1, 10^{-2}$ and 5×10^{-4}. Last column: solution for zero source term and initial condition $u_0(x) = \sin^2(8\pi(1-x)^2)$ with small diffusion $\nu = 5 \times 10^{-4}$.

information propagates very far in time, and we cannot pre-compute the solution later in time any more without knowing the solution earlier in time when the diffusion parameter becomes small. It is therefore difficult to do PinT computations, especially when $\nu \to 0$, in the hyperbolic limit. This is fundamentally different from the heat equation case, and only becomes manifest with periodic boundary conditions and small diffusion, an important point when testing the performance of PinT methods on advection dominated problems.

2.3. Burgers' equation

To illustrate the difference between the various PinT methods in a non-linear setting, we will use Burgers' equation,

$$\begin{aligned} \partial_t u(x,t) - \nu \partial_{xx} u(x,t) + \tfrac{1}{2}\partial_x(u^2(x,t)) &= g(x,t) \quad \text{in } \Omega \times (0,T], \\ u(x,0) &= u_0(x) \quad \text{in } \Omega, \end{aligned} \tag{2.6}$$

with $\nu > 0$. We show in Figure 2.4 in the top row the solution obtained with zero Dirichlet boundary conditions, and in the bottom row with periodic boundary conditions. In the first three panels, we use a zero initial condition, $u_0(x) = 0$ and the same source term (2.4) we had used for the heat and advection-diffusion equation, also for three different values of the diffusion parameter, $\nu = 1, 10^{-2}$ and 5×10^{-4}. We see that when ν is large, then the diffusion part dominates and the solution has similar properties as the solution of the heat equation. If ν is small and the non-linear advection part starts playing a dominant role, then the solution is transported from left to right over much longer time, as we see in the top middle two panels in Figure 2.3, like for advection-diffusion. However, we see a further very important new phenomenon in the non-linear case: the solution shape changes as well, and even with the smooth source term, very sharp edges are forming in the solution, so called shock waves, containing very high frequency components that travel fare in space and time. In the top right panel, we use a zero source term, but a non-zero initial condition $u_0(x) = \sin^2(8\pi(1-x)^2)$ as for advection-diffusion before, and again the small diffusion parameter 5×10^{-4}. We see that also from the already fine features present in the high frequency components of the initial condition even sharper edges are formed in shock waves, and all are transported far in time. Nevertheless, for both ν large and ν small, we can still compute the solution for $t \in (1.25, 2.5)$ before we obtain the solution earlier in time, because all solution components eventually are diffused or leave the domain, as in the advection-diffusion case in the top row in Figure 2.3.

In contrast, for periodic boundary conditions, we see in the bottom row of Figure 2.4 that our observations from the bottom row for the advection-diffusion equation in Figure 2.3 are further accentuated, as soon as the diffusion parameter becomes small: more and more fine information is generated

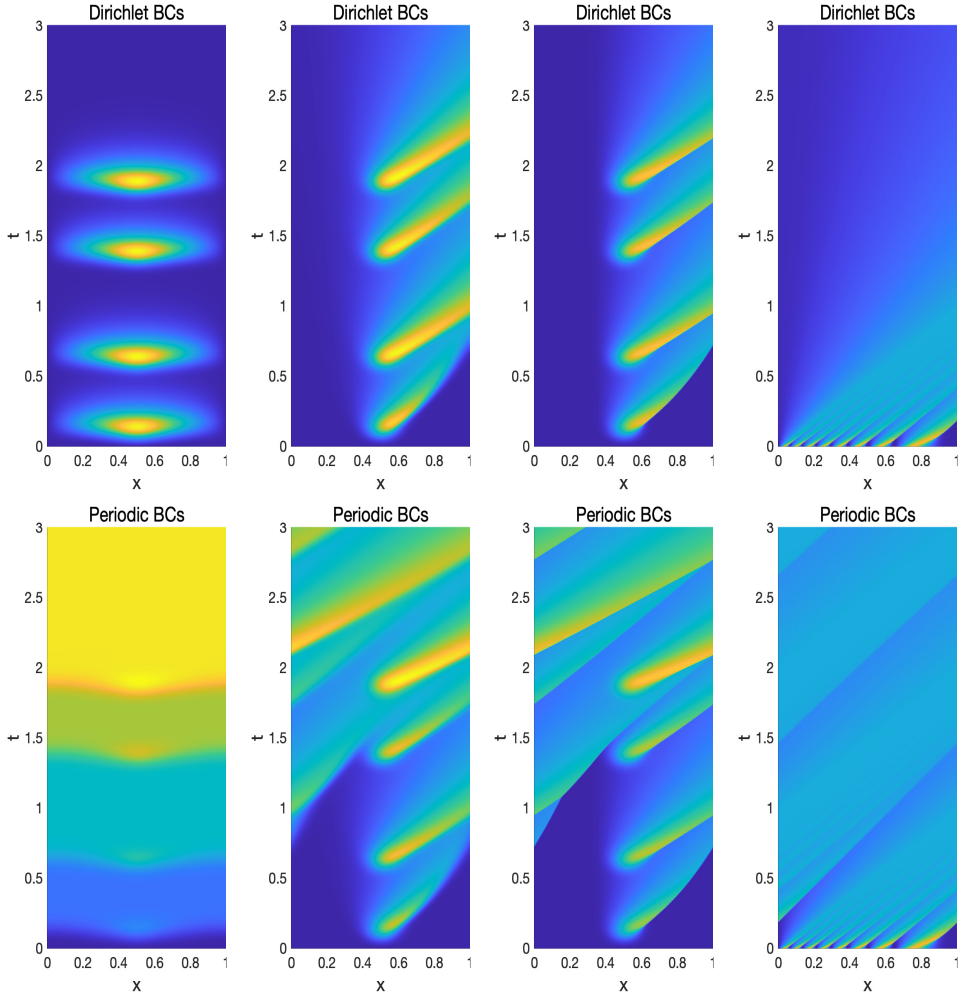

Figure 2.4. Burgers' equation with zero Dirichlet boundary conditions (top row) and periodic boundary conditions (bottom row). In the first three pannels, we use a zero initial condition, $u_0(x) = 0$, and the same source term as in Figure 2.2 for the heat equation, and a smaller and smaller diffusion parameter $\nu = 1, 10^{-2}$ and 5×10^{-4}. Last column: solution for zero source term and initial condition $u_0(x) = \sin^2(8\pi(1-x)^2)$ with small diffusion $\nu = 5 \times 10^{-4}$.

and transported very far in time, through shock waves that are forming. For successful PinT computation, such high frequency shock waves must be transported by a mechanism that propagate them effectively far in space and time, which is very difficult using a coarse grid for example. The last panel without source and just a non-zero initial condition shows the same effect for an initial condition transported over very long time: one cannot pre-compute the solution later in time any more without knowing the solution earlier in time when the diffusion parameter becomes small. It is therefore even harder to do PinT computations in such non-linear problems when $\nu \to 0$, in the hyperbolic limit, where shock waves are natural in the solutions and all frequency components in it travel very far in space and time. Note that again we need periodic boundary conditions and small diffusion to encounter these difficulties, an important point when testing the performance of PinT methods on such problems. In the next subsection we will see that for hyperbolic problems, these difficulties already appear no matter what boundary conditions are used.

2.4. Second-order wave equation

For hyperbolic problems, we will use the second-order wave equation as our model problem,

$$\begin{array}{rcll} \partial_{tt} u(x,t) & = & c^2 \partial_{xx} u(x,t) + g(x,t) & \text{in } (0,1) \times (0,T], \\ u(x,0) & = & u_0(x) & \text{in } (0,1), \\ \partial_t u(x,0) & = & 0 & \text{in } (0,1), \end{array} \tag{2.7}$$

with a constant wave speed $c > 0$. We show in Figure 2.5 in the first three panels the solution of the wave equation with the same source term (2.4) we used before, with Dirichlet, Neumann and periodic boundary conditions. In all cases, we observe that for the wave equation, the solution depends in a complex and detailed manner over a long time on the various source terms in space and time. In the last panel in Figure 2.5, we show the solution of the wave equation with zero source term, but using the initial condition $u_0(x) = \sin^2(8\pi(1-x)^2)$ and zero first derivative in time. We see that the solution of this hyperbolic problem depends in a very detailed manner on all frequency components present in the initial condition, and this would be the same also for the other boundary conditions. This is typical for hyperbolic problems and all boundary conditions; one only needs periodic boundary conditions in the advection-diffusion and Burgers' equation case to make this difficulty appear for small ν, because the advection term is first order and the transport has a direction. For the wave equation and other hyperbolic problems, this detailed and long time propagation in several directions with reflections is present already for Dirichlet and Neumann boundary conditions. It is this propagation of fine information over long

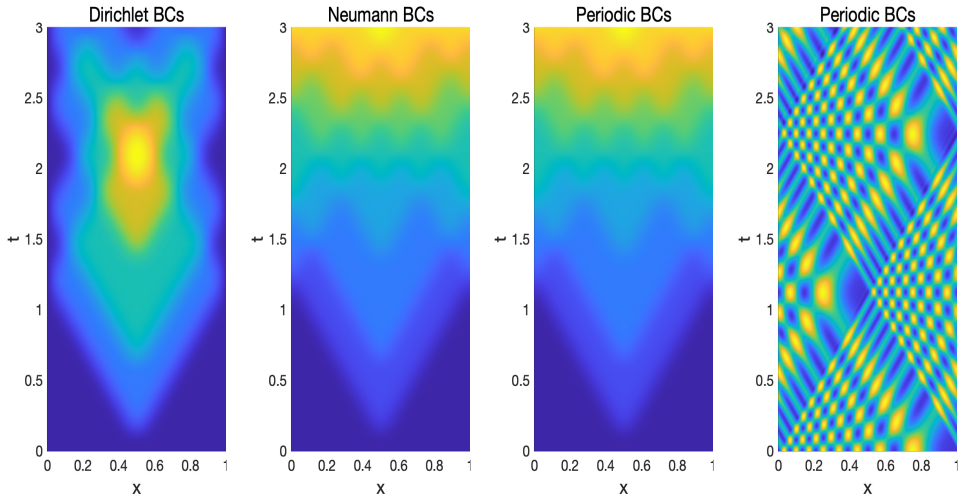

Figure 2.5. Solution of the second-order wave equation, with $c^2 = 0.2$, zero initial condition $u_0(x) = 0$ and the same source term as in Figure 2.2 for Dirichlet, Neumann and periodic boundary conditions (first three panels), and in the last panel the solution with zero source term and non-zero initial condition $u_0(x) = \sin^2(8\pi(1-x)^2)$.

time in hyperbolic problems that makes time parallelization more challenging than for parabolic problems, and requires different PinT techniques to address it.

3. Effective PinT methods for hyperbolic problems

We have seen in Section 2 that parabolic problems have solutions which are rather local in time, see Figure 2.2, where with Dirichlet conditions all information is forgotten very rapidly over time, and with Neumann and periodic boundary conditions only the lowest frequency component, namely the constant, remains over long time. This changes when transport terms are present and become dominant, see Figures 2.3, 2.4, and in the hyperbolic limit, and for hyperbolic problems in general, all frequency components can travel arbitrarily far in space and time, see Figure 2.5 for the second order wave equation. PinT methods must take this into account to be effective. It is interesting that many methods designed specifically for hyperbolic problems also work well (or even better) for parabolic problems. An exception are the mapped tent pitching methods introduced at the end of Subsection 3.2 which use the finite speed of propagation in hyperbolic problems in their construction. On the other hand, PinT methods designed for parabolic problems (see Section 4) do not perform well in general for hyperbolic problems.

3.1. Historical development

We present four PinT methods that have proven effective for hyperbolic problems. For each method, we show the main theoretical properties and demonstrate these properties using the four PDEs introduced in Section 2.

The first methods are rooted in solving overlapping or non-overlapping space-time continuous subproblems, an approach initially proposed for parabolic problems in (Gander 1999) and independently introduced in (Giladi and Keller 2002). This strategy incorporates elements of both Domain Decomposition (DD) methods, a long-established technique for parallel PDE solving going back to (Schwarz 1870), and Waveform Relaxation (WR) methods, which originated in circuit simulations (Lelarasmee, Ruehli and Sangiovanni-Vincentelli 1982*b*). These methods have been developed and analyzed both for parabolic and hyperbolic problems in (Gander 1997), and the name Schwarz Waveform Relaxation (SWR) methods was coined in (Gander et al. 1999). Further results for non-linear parabolic problems can be found in (Gander 1998, Gander and Rohde 2005*a*). Optimized Schwarz Waveform Relaxation (OSWR) methods using more effective transmission conditions were developed for parabolic problems in (Gander and Halpern 2007, Bennequin, Gander and Halpern 2009, Bennequin, Gander, Gouarin and Halpern 2016*a*), and for hyperbolic problems in (Gander, Halpern and Nataf 2003, Gander and Halpern 2004), see also (Gander, Lunowa and Rohde 2023*c*) for non-linear advection-diffusion equations. The recently developed Unmapped Tent Pitching (UTP) technique (Ciaramella, Gander and Mazzieri 2023) is based on SWR. There are also Dirichlet-Neumann and Neumann-Neumann Waveform relaxation variants, see (Gander, Kwok and Mandal 2016*b*, Gander, Kwok and Mandal 2021*b*).

The third method is based on the time parallelization of the Integral Deferred Correction (IDC) technique. IDC for evolution problems was first introduced in (Böhmer and Stetter 1984), and was later identified as a specialized time-integrator in (Dutt, Greengard and Rokhlin 2000), which theoretically has the capability to generate numerical solutions of arbitrarily high order by accurately treating the associated integral. Revisionist Integral Deferred Correction (RIDC) is one such technique (Christlieb et al. 2010) that can be used parallel in time, and there is another recent parallel version (PIDC) from (Guibert and Tromeur-Dervout 2007), which we will introduce in detail in Section 3.3.

The fourth time-parallel method that we will introduce in Section 3.4 is the ParaExp method, proposed a decade ago by (Gander and Güttel 2013), which relies on a new strategy of separately handling the initial value and source term, see also (Merkel, Niyonzima and Schöps 2017, Kooij, Botchev and Geurts 2017), and (Gander, Güttel and Petcu 2018*a*) for a non-linear variant.

Finally, in Section 3.5, we will present the ParaDiag family of methods. Time-parallel methods based on diagonalization were first proposed in (Maday and Rønquist 2008) as direct time parallel solvers, without iteration, and they were studied in more detail in (Gander, Halpern, Ryan and Tran 2016*a*) for parabolic problems, with a non-linear variant in (Gander and Halpern 2017), and in (Gander, Halpern, Rannou and Ryan 2019) for hyperbolic problems. Rapidly then iterative variants appeared, within WR methods (Gander and Wu 2019) or within Parareal (Gander and Wu 2020). Approximate ParaDiag methods were also applied as preconditioners for Krylov methods directly to the all-at once system in (McDonald et al. 2018) and (Liu and Wu 2020). A comprehensive study of ParaDiag methods appeared in (Gander et al. 2019). Since then, ParaDiag methods have gained widespread traction in the PinT field, with new techniques enhancing these methods, see for example (Kressner, Massei and Zhu 2022) for a direct ParaDiag technique using interpolation, and (Gander and Palitta 2024) for a new ParaDiag variant combining the Sherman-Morrison-Woodbury formula and Krylov techniques.

3.2. Schwarz waveform relaxation (SWR) methods

SWR combines the strengths of the classical Schwarz DD method and WR, while overcoming some of their inherent limitations. In the context of evolution PDEs, the Schwarz DD method typically involves a uniform implicit time discretization, followed by the application of the DD technique to solve the resulting elliptic problems at each time step sequentially; see e.g. (Cai 1991, Meurant 1991, Cai 1994). The DD iterations need to converge at each time step before proceeding to the next, across all subdomains, and one has to use the same time discretization across subdomains, which undermines a key advantage of DD methods, namely to tailor numerical treatments for each subdomain individually.

On the other hand, the classical WR method begins with a system of ODEs, often obtained from a spatial discretization of an evolution PDE, which is then solved using a dynamic iteration similar to the Picard iteration, but using an appropriate system partitioning. For instance, in the case of the linear system of ODEs (2.1), the WR iteration can be expressed as

$$\frac{d\boldsymbol{u}^k(t)}{dt} - M\boldsymbol{u}^k(t) = N\boldsymbol{u}^{k-1}(t) + f(t),\ t \in (0,T),$$

where $k \geq 1$ denotes the iteration index, $\boldsymbol{u}^k(0) = \boldsymbol{u}_0$ for all $k \geq 0$, and (M,N) represents a consistent splitting of A such that $A = M + N$. For Jacobi (diagonal) or Gauss-Seidel (triangular) type splittings, solving for $\boldsymbol{u}^k(t)$ boils down to solving a series of scalar ODEs. In the Jacobi case, all these ODEs can be solved in parallel, making this into a PinT method in

the sense that the future of all unknowns is approximated before the future of connected unknowns is already known. Similarly in the Gauss-Seidel case, one can obtain such parallelism using red-black or other colorings. Even more parallelism can be introduced using the cyclic reduction technique, see (Worley 1991, Horton, Vandewalle and Worley 1995, Simoens and Vandewalle 2000). However, a significant challenge of WR lies in finding an effective system splitting to ensure rapid convergence. As Nevanlinna remarked in (Nevanlinna 1989):

"*In practice, one is interested in knowing what subdivisions yield fast convergence for the iterations... The splitting into subsystems is assumed to be given. How to split in such a way that the coupling remains weak is an important question.*"

A bad splitting can lead to arbitrarily slow convergence or even divergence, rendering WR impractical.

SWR circumvents these limitations by initially decoupling the spatial domain (rather than performing a spatial discretization first) and then independently solving the space-time continuous PDEs on these subdomains, similar to the WR approach. This approach allows for the use of tailored space and time discretizations for each subdomain problem; but more importantly, knowing that the space-time subdomain problems are coupled for a particular PDE, one can design transmission conditions which decouple the problems such that the methods converge very rapidly, completely addressing the difficulty identified by Nevanlinna above. This leads to the class of Optimized Schwarz Waveform Relaxation (OSWR) methods, which have been studied for many different types of PDEs, see e.g. (Martin 2009) for the shallow water equations, (Courvoisier and Gander 2013) for the time domain Maxwell equations, (Halpern and Szeftel 2010, Besse and Xing 2017, Antoine and Lorin 2017) for Schrödinger equations, (Audusse, Dreyfuss and Merlet 2010) for the primitive equations of the ocean, (Antoine and Lorin 2016) for quantum wave problems, (Wu 2017) for fractional diffusion problems, (Thery, Pelletier, Lemarié and Blayo 2022) the coupled Ekman boundary layer problem, and also the many references therein. OSWR methods have distinct convergence characteristics for first-order parabolic problems (such as the advection-diffusion equation (2.5) and the nonlinear Burgers' equation (2.6)) compared to second-order hyperbolic problems like the wave equation (2.7). In the following, we present these two types of problems separately.

3.2.1. First-order parabolic problems

For the advection-diffusion equation (2.5) with homogeneous Dirichlet boundary conditions, $u(0,t) = u(L,t) = 0$, and an initial condition, $u(x,0) = u_0(x)$, the OSWR method with the two overlapping subdomains $\Omega_1 := (0, \beta L)$ and $\Omega_2 := (\alpha L, L)$, $\alpha < \beta$, and Robin transmission conditions is

given by

$$\begin{cases} \partial_t u_1^k(x,t) + \mathcal{L}u_1^k(x,t) = 0 \quad (x,t) \in \Omega_1 \times (0,T], \\ u_1^k(0,t) = 0, \\ \frac{1}{p}\partial_x u_1^k(\beta L,t) + u_1^k(\beta L,t) = \frac{1}{p}\partial_x u_2^{k-1}(\beta L,t) + u_2^{k-1}(\beta L,t), \end{cases} \\ \begin{cases} \partial_t u_2^k(x,t) + \mathcal{L}u_2^k(x,t) = 0 \quad (x,t) \in \Omega_2 \times (0,T], \\ \frac{1}{p}\partial_x u_2^k(\alpha L,t) - u_2^k(\alpha L,t) = \frac{1}{p}\partial_x u_1^{k-1}(\alpha L,t) - u_1^{k-1}(\alpha L,t), \\ u_2^k(L,t) = 0, \end{cases} \tag{3.1}$$

with $\mathcal{L} = \partial_x - \nu\partial_{xx}$, and initial conditions $u_1^k(x,0) = u_0(x)$ for $x \in \Omega_1$ and $u_2^k(x,0) = u_0(x)$ for $x \in \Omega_2$. Here, $k \geq 1$ represents the iteration index, $\{u_1^0(\alpha L,t), u^0(\beta L,t)\}$ are initial guesses for $t \in (0,T]$, and $0 < \alpha \leq \beta < 1$, and $(\beta - \alpha)L$ denotes the overlap size. We use Robin Transmission Conditions (TCs) in (3.1) with parameter $p > 0$ at $x = \alpha L$ and $x = \beta L$ to transmit information between the subdomains, and the classical Dirichlet TCs correspond to the limit $p \to \infty$, i.e., $u_1^k(\beta L,t) = u_2^{k-1}(\beta L,t)$ and $u_2^k(\alpha L,t) = u_1^{k-1}(\alpha L,t)$. Generalizing this two-subdomain case in (3.1) to a multi-subdomain scenario is straightforward, and the OSWR method for nonlinear problems is obtained by simply replacing the linear operator $\mathcal{L}$ by the corresponding non-linear one in (3.1).

For the OSWR iteration given in (3.1), the optimized choice of the parameter p and the corresponding convergence factor were analyzed by Gander and Halpern in 2007 under the simplified assumption that the space domain is unbounded.

Theorem 3.1. (Gander and Halpern 2007) *For OSWR* (3.1) *with* $\mathcal{L} = \partial_x - \nu\partial_{xx}$ *and an overlap size of* $l > 0$*, the optimized choice for the Robin parameter, denoted by* p^**, is given by* $p^* = \frac{\tilde{p}^*\nu}{s}$*, where* $\tilde{p}^*$ *is the unique solution of the nonlinear equation*

$$R_0(y_0, \tilde{p}^*, y_0) = R_0(\bar{y}(y_0, \tilde{p}^*), \tilde{p}^*, y_0), \tag{3.2a}$$

provided that $y_0 := \frac{l}{\nu} < y_c$*, where* y_c *is a constant equal to* 1.618386576...*, and*

$$R_0(y, \tilde{p}, y_0) = \frac{(y-\tilde{p})^2 + y^2 - y_0^2}{(y+\tilde{p})^2 + y^2 - y_0^2} e^{-y},$$

$$\bar{y}(y_0, \tilde{p}) = \sqrt{\frac{y_0^2 + 2\tilde{p} + \sqrt{\tilde{p}(-\tilde{p}^3 - 4\tilde{p}^2 + (4 + 2y_0^2)\tilde{p} + 8y_0^2)}}{2}}.$$

The function $R_0(y, \tilde{p}, y_0)$ *is the convergence factor of the OSWR iteration, obtained in Fourier space, where* y *corresponds to a single Fourier mode* $\omega \in [\frac{\pi}{T}, \frac{\pi}{\Delta t}]$*, i.e.,* $y = \frac{l}{\nu}\omega$.

If, on the other hand, $y_0 \geq y_c$, then $\tilde{p}^*$ is the unique solution of

$$y_0 = \tilde{p}^* \sqrt{\frac{\tilde{p}^*}{4 + \tilde{p}^*}}. \tag{3.2b}$$

With the optimized Robin parameter p^*, the convergence factor ρ over all relevant Fourier modes can be bounded as

$$\rho := \max_{y \in [y_{min}, y_{max}]} R_0(y, \tilde{p}^*, y_0) \leq R_0(\bar{y}(y_0, \tilde{p}^*), \tilde{p}^*, y_0), \tag{3.2c}$$

where $y_{min} = \frac{l\pi}{\nu T}$ and $y_{max} = \frac{l\pi}{\nu \Delta t}$.

As we mentioned above, classical SWR with Dirichlet TCs corresponds to the case $p = \infty$ in (3.1). By setting $p = \infty$ in the function R_0, we obtain

$$\rho \leq e^{-y_{\min}} = e^{-\frac{l\pi}{\nu T}}, \tag{3.3}$$

which was analyzed in (Gander and Halpern 2007, Section 3).

We now show a numerical experiment to illustrate OSWR. We discretize the advection-diffusion operator $\mathcal{L}$ using a centered finite difference method and Backward Euler in time. Let $L = 8.2$, $T = 5$, $\Delta t = 0.01$, $\Delta x = 0.02$, and $l = 2\Delta x$. The initial value for the advection-diffusion equation is $u_0(x) = e^{-10(x-L/2)^2}$. We show in Figure 3.6 (left) the theoretical convergence factor of the SWR method with Dirichlet and optimized Robin TCs as function of the diffusion parameter ν. We see that the convergence factor becomes small when the advection term becomes dominant, and the method converges faster and faster. To test this numerically, we decompose the spatial domain $(0, L)$ into 4 subdomains and solve the advection-diffusion equation using the OSWR method for several values of the parameter ν. The method starts from a random initial guess, and we stop the iteration when the error between the iterate and the converged solution is less than 10^{-8}. The iteration number for Dirichlet and optimized Robin TCs is shown in Figure 3.6 (right), and we see that indeed fewer iterations are required when ν is small, as predicted by the theoretical results in the left panel.

For a given ν, the discretized OSWR method with four subdomains may however not converge as rapidly as predicted by the theoretical convergence factor ρ obtained for a two subdomain decomposition at the continuous level on an unbounded domain. For instance, when $\nu = 0.1$, the iteration numbers shown in Figure 3.6 are 92 for Dirichlet TCs and 28 for Robin TCs, but iteration numbers predicted by ρ are 32 for Dirichlet TCs and 4 for Robin TCs, which are significantly smaller. The reason for this discrepancy is that the convergence factor ρ in Theorem 3.1 is analyzed for the 2-subdomain case at the space-time continuous level on an unbounded domain, and we tested the discretized SWR method in the 4-subdomain case on a bounded domain. Convergence analyses of SWR with Dirichlet TCs in the multi-subdomain case can be found in (Gander and Stuart 1998, Wu, Huang and

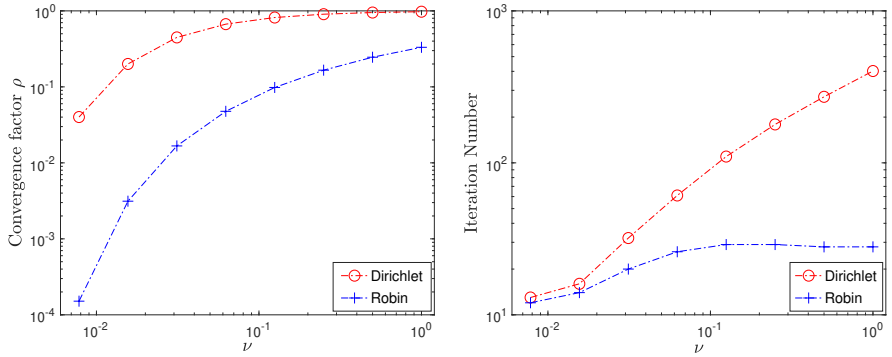

Figure 3.6. Left: the theoretical convergence factor of the OSWR method when applied to the advection-diffusion equation (2.5) decreases when the diffusive parameter ν decreases. Right: the iteration, measured in a 4-subdomain numerical experiment with tolerance of 10^{-8}, supports this prediction very well.

Huang 2012). However, a comprehensive convergence analysis for Robin TCs in the multi-subdomain case is still missing so far. For Robin TCs, the convergence of SWR in the two-subdomain case at the semi-discrete level can be found in (Wu and Al-Khaleel 2014), see also the detailed studies in (Gander, Halpern, Hubert and Krell 2018*b*, Gander, Halpern, Hubert and Krell 2021*a*) for the steady state case between continuous and discrete analyses on bounded and unbounded domains.

In addition to Dirichlet and Robin TCs, there are efforts to further accelerate OSWR using Ventcel TCs (Bennequin, Gander, Gouarin and Halpern 2016*b*). Essentially, these TCs serve as local approximations of the *optimal* TCs analyzed in Fourier (or Laplace) space in (Gander and Halpern 2007, Section 3), namely

$$\partial_x - \frac{1}{2\nu}\mathcal{F}^{-1}\left(1+\sqrt{1+\mathrm{i}4\nu\omega}\right),$$

where $\mathrm{i} = \sqrt{-1}$ and $\mathcal{F}^{-1}$ denotes the inverse Fourier transform with ω representing the Fourier mode. In an asymptotic sense, i.e., $l = C_1\Delta x$, $\Delta t = C_1\Delta x^{\beta}$, and with Δx being small, the convergence factor ρ satisfies $\rho = 1 - \mathcal{O}(\Delta x^{\gamma})$, where $\gamma > 0$ is a quantity that depends on β; see (Gander and Halpern 2007, Bennequin et al. 2016*b*, Bennequin et al. 2009). Convolution TCs analyzed in (Wu and Xu 2017) result in a mesh-independent constant convergence factor $\rho = 1 - C$, where $C \in (0, 1)$. This is particularly useful for handling evolution PDEs with nonlocal terms, such as Volterra partial integro-differential equations.

3.2.2. Second-order hyperbolic problems

Unlike for first-order parabolic problems, for second-order hyperbolic problems (e.g., the wave equation (2.7)), the SWR method converges in a finite number of iterations, even when using simple Dirichlet transmission conditions. Applying SWR to the wave equation for a two subdomain decomposition leads to the algorithm

$$\begin{cases} \partial_{tt} u_1^k(x,t) = c^2 \partial_{xx} u_1^k(x,t) + g(x,t) & (x,t) \in \Omega_1 \times (0,T], \\ u_1^k(x,0) = u_0(x),\ \partial_t u_1^k(x,0) = \tilde{u}_0(x) & x \in (0,\beta L), \\ u_1^k(0,t) = 0,\ u_1^k(\beta L,t) = u_2^{k-1}(\beta L,t) & t \in (0,T), \end{cases} \\ \begin{cases} \partial_{tt} u_2^k(x,t) = c^2 \partial_{xx} u_2^k(x,t) + g(x,t) & (x,t) \in \Omega_2 \times (0,T], \\ u_2^k(x,0) = u_0(x),\ \partial_t u_2^k(x,0) = \tilde{u}_0(x) & x \in (\alpha L, L), \\ u_2^k(\alpha L,t) = u_1^{k-1}(\alpha L,t),\ u_2^k(L,t) = 0 & t \in (0,T), \end{cases} \tag{3.4}$$

where $c > 0$ and $0 < \alpha < \beta < 1$.

Theorem 3.2. (Gander 1997, Theorem 6.3.3) For the SWR method (3.4), the errors at the interfaces $x = \alpha L$ and $x = \beta L$ become zero after k iterations, i.e., $u_1^k(\alpha L,t) - u(\alpha L,t) = 0$ and $u_2^k(\beta L,t) - u(\beta L,t) = 0$, provided that $k > \frac{Tc}{\beta - \alpha}$.

The reason for this convergence result lies in the finite speed of propagation inherent to hyperbolic problems. By exploiting this property, similar results can be achieved in the case of many subdomains, as well as for more general decompositions in higher dimensions (Gander and Halpern 2004) and for other hyperbolic equations, see for example (Gander and Rohde 2005*b*) for 1D nonlinear conservation laws.

Theorem 3.2 shows a very important property of SWR applied to hyperbolic problems, already indicated in (Gander et al. 2003, Figure 3.1): the subdomains are computing the exact solution in the cone within the subdomain which is only influenced by the initial condition, and not by the transmission conditions where possibly incorrect data is still coming from the neighboring subdomains. Using this property, one can choose the space-time subdomains in SWR applied to hyperbolic problems in order to avoid iterations and advance directly with space-time subdomain solves in parallel. To illustrate this, it is best to consider again the one-dimensional wave equation (2.7) and a *red-black* domain decomposition with generous overlap, as it was done in (Ciaramella et al. 2023), see Figure 3.7. In the first panel, we solve the wave equation in parallel within the three red subdomains in space-time, i.e. in $\Omega_j \times (0,T_1)$ for $j = 1,3,5$. We use arbitrary interface data at $x = x_2$ and $x = x_4$ because the solution there is not yet known. Due to the finite speed of propagation, we obtain the exact solution within triangular tents in space-time, bounded by the characteristic lines of the

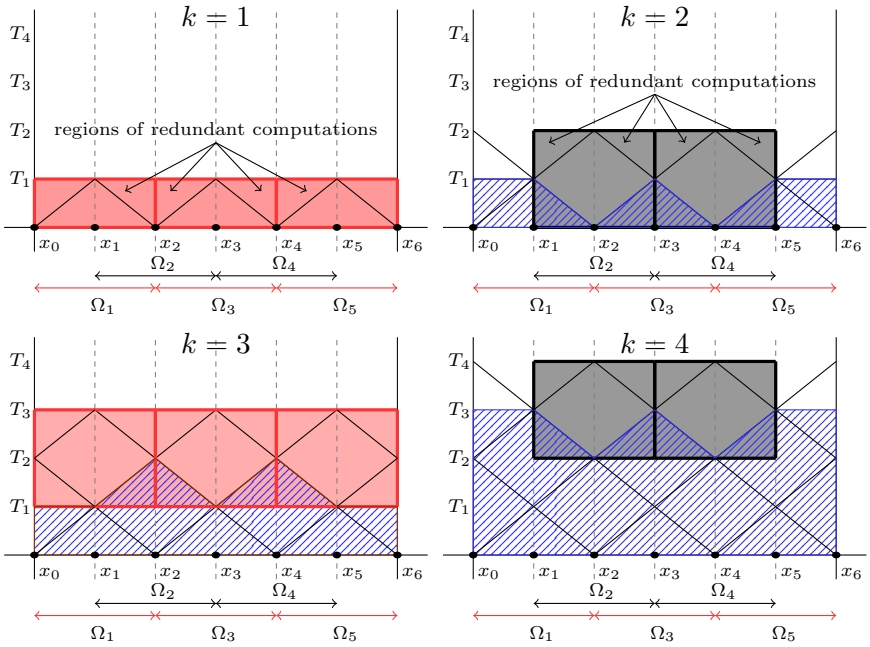

Figure 3.7. Illustration of Red-Black SWR with generous overlap.

wave equation. These tents are marked in blue in the second panel, and the solution is also correct in two additional small zones on the left and right, since the outer boundary conditions are known. In this first red solve, SWR also computes a not yet correct approximation in the regions above the correct tents, as indicated in the first panel: SWR performs redundant computations, as advocated by Nievergelt already in order to obtain more parallelism. In the next step of this red-black SWR, one computes wave equation solutions in the black subdomains $\Omega_j \times (0, T_2)$ for $j = 2, 4$ in space-time, as indicated in the second panel of Figure 3.7. Since we already have the correct solution in the blue region, the exact solution is now obtained in the two rhomboid blue tents indicated in the third panel, again at the cost of some redundant computations. Next, one solves again in the red subdomains, but now further in time in the interval (T_1, T_3). Continuing this red-black SWR algorithm, we obtain the exact solution further and further advanced in time, as indicated in the last panel in Figure 3.7.

This red-black SWR algorithm is an effective and simple way to implement one of the most powerful current space-time solvers for hyperbolic problems, namely the Mapped Tent Pitching (MTP) algorithm from (Gopalakrishnan, Schöberl and Wintersteiger 2017), see (Gopalakrishnan, Hochsteger, Schöberl and Wintersteiger 2020) for its application to time domain Maxwell equations. In MTP, one maps the tent shape which we have

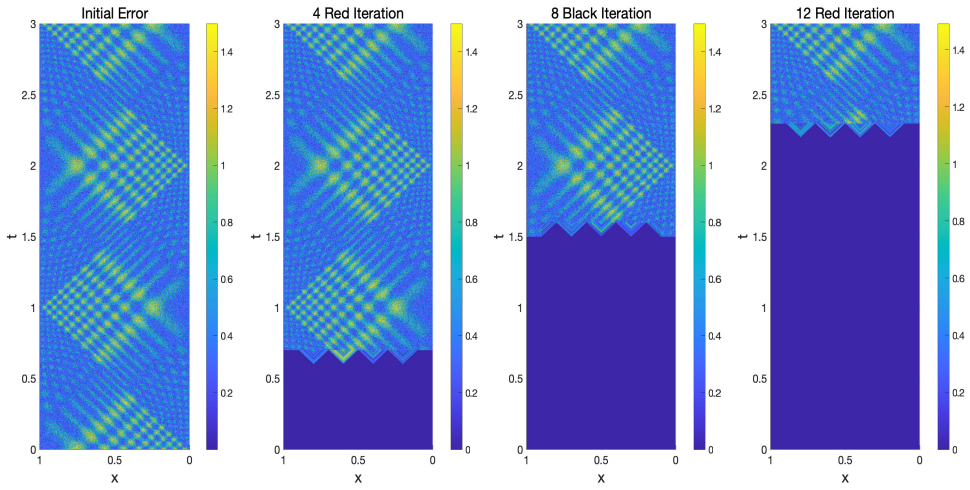

Figure 3.8. Red-black SWR or equivalently UTP applied to the second order wave equation. From left to right: initial error with random initial guess, and then 4th red iteration, 8th black iteration and 12th red iteration.

seen in red-black SWR to space-time cylinders in which the solution is then computed by classical time stepping, and then the solution is mapped back, thus avoiding redundant computations. However, one has the extra cost of computing the mapping, and also after the mapping the computational domains have the same size as the space-time subdomains in the red-black SWR above, and thus comparable computational cost. In addition, in MTP, order reduction was observed due to the mapping, and specialized time integrators were developed and need to be used to avoid this. In contrast, in red-black SWR, also called now Unmapped Tent Pitching (UTP), no order reduction occurs, and red-black SWR can be very easily implemented, also for higher spatial dimensions, using Restricted Additive Schwarz (RAS) techniques from DD directly applied to the all-at-once space-time system, see (Gander 2008) for an explanation. We show in Figure 3.8 an example of using red-black SWR or equivalently UTP to solve our wave equation model problem 2.7 whose solution is shown in Figure 2.5 on the right. We see that without knowing anything about the tent structure, UTP constructs the exact solution in the red and black tents, and advances exactly like MTP. Note, that UTP can also as easily be applied to non-linear hyperbolic problems, and if one does not know the tent height, it suffices to look at the residual in the computed solution which indicated naturally the tent height by how far the residual has become zero in time, and one can adapt the time domain length $T_i - T_{i-1}$ accordingly!

Clearly, the original MTP is not appropriate for parabolic problems, since

for such problems the speed of propagation is infinite, and hence there are no tents in which the solution would be correct. SWR and thus UTP however can be very effective also for parabolic problems, especially the optimized SWR variants, see e.g (Gander and Halpern 2007, Bennequin et al. 2009), and in case of slightly diffusive problems like our advection dominated diffusion model problem one could consider to apply UTP, maybe with one or two additional iterations in each time slab.

3.3. Time Parallel IDC

Integral Deferred Correction (IDC), introduced by (Dutt et al. 2000), serves as a technique to obtain high-order numerical solutions through an iterative correction procedure. While the original IDC is sequential in time, there are two more recent techniques to parallelize IDC in time: the Pipeline IDC (PIDC) method by (Guibert and Tromeur-Dervout 2007) and the Revisionist IDC (RIDC) method by (Christlieb et al. 2010). Both PIDC and RIDC fundamentally differ from the original IDC, but to understand them, it is necessary to explain first how IDC works. To this end, we rewrite the nonlinear ODE (2.2) as an integral equation,

$$\boldsymbol{u}(t) = \boldsymbol{u}_0 + \int_0^t f(\boldsymbol{u}(\tau), \tau) d\tau, \quad t \in (0, T]. \tag{3.5}$$

Suppose we already have a rough approximation $\tilde{\boldsymbol{u}}(t)$ of the desired solution $\boldsymbol{u}(t)$, for example by simply setting $\tilde{\boldsymbol{u}}(t) \equiv \boldsymbol{u}_0$ for $t \in [0, T]$ or by solving the ODE with lower accuracy. To improve the approximation $\tilde{\boldsymbol{u}}(t)$, we introduce the error $\boldsymbol{e}(t) := \boldsymbol{u}(t) - \tilde{\boldsymbol{u}}(t)$, and the residual

$$\boldsymbol{r}(t) \quad := \quad \boldsymbol{u}_0 + \textstyle\int_0^t f(\tilde{\boldsymbol{u}}(\tau), \tau) d\tau - \tilde{\boldsymbol{u}}(t), \quad t \in (0, T]. \tag{3.6}$$

By substituting $\boldsymbol{u}(t) = \boldsymbol{e}(t) + \tilde{\boldsymbol{u}}(t)$ into (3.5) and using (3.6), we can express the error $\boldsymbol{e}(t)$ in terms of the residual $\boldsymbol{r}(t)$,

$$\begin{aligned} \boldsymbol{e}(t) &= \boldsymbol{u}_0 + \int_0^t f(\tilde{\boldsymbol{u}}(\tau) + \boldsymbol{e}(\tau), \tau) d\tau - \tilde{\boldsymbol{u}}(t) \\ &= \boldsymbol{r}(t) + \int_0^t [f(\tilde{\boldsymbol{u}}(\tau) + \boldsymbol{e}(\tau), \tau) - f(\tilde{\boldsymbol{u}}(\tau), \tau)] d\tau, \quad t \in (0, T]. \end{aligned} \tag{3.7}$$

Taking a derivative, this is equivalent to the differential equation

$$\boldsymbol{e}'(t) - \boldsymbol{r}'(t) = f(\tilde{\boldsymbol{u}}(t) + \boldsymbol{e}(t), t) - f(\tilde{\boldsymbol{u}}(t), t), \quad t \in (0, T]. \tag{3.8}$$

Let the current approximate solution $\tilde{\boldsymbol{u}}$ be known at specific time points $0 = t_0 < t_1 < t_2 < \cdots < t_M = T$, $\{\boldsymbol{u}_m^k\} := \tilde{\boldsymbol{u}}(t_m)$. The procedure to obtain the next approximate solution $\{\boldsymbol{u}_m^{k+1}\}$ involves discretizing (3.8) and using a quadrature rule to approximate $\boldsymbol{r}(t)$ at the discrete time nodes. Applying

the linear-θ method (with $\theta \in [0,1]$) to (3.8) yields

$$\begin{aligned}\boldsymbol{e}_{m+1} - \boldsymbol{e}_m =& \boldsymbol{r}_{m+1} - \boldsymbol{r}_m + \Delta t_m(1-\theta)[f(\boldsymbol{u}_m^{k+1}, t_m) - f(\boldsymbol{u}_m^k, t_m)] + \\ & \Delta t_m \theta [f(\boldsymbol{u}_{m+1}^{k+1}, t_{m+1}) - f(\boldsymbol{u}_{m+1}^k, t_{m+1})],\end{aligned} \tag{3.9}$$

where $m = 0, 1, \ldots, M-1$ and $\Delta t_m = t_{m+1} - t_m$. From (3.6), $\boldsymbol{r}_{m+1} - \boldsymbol{r}_m = \int_{t_m}^{t_{m+1}} f(\boldsymbol{u}^k(\tau), \tau) d\tau - (\boldsymbol{u}_{m+1}^k - \boldsymbol{u}_m^k)$. Substituting this into (3.9) and then using $\boldsymbol{u}_m^{k+1} = \boldsymbol{u}_m^k + \boldsymbol{e}_m^k$, we obtain

$$\begin{aligned}\boldsymbol{u}_{m+1}^{k+1} =& \boldsymbol{u}_m^{k+1} + \Delta t_m(1-\theta)[f(\boldsymbol{u}_m^{k+1}, t_m) - f(\boldsymbol{u}_m^k, t_m)] + \\ & \Delta t_m \theta [f(\boldsymbol{u}_{m+1}^{k+1}, t_{m+1}) - f(\boldsymbol{u}_{m+1}^k, t_{m+1})] + \int_{t_m}^{t_{m+1}} f(\boldsymbol{u}^k(\tau), \tau) d\tau,\end{aligned}$$

where the integral is computed using a quadrature rule,

$$\int_{t_m}^{t_{m+1}} f(\boldsymbol{u}^k(\tau), \tau) d\tau \approx \sum\nolimits_{j=1}^{M} \omega_{m,j} f(\boldsymbol{u}_j^k, t_j). \tag{3.10a}$$

The quadrature weights are determined by integrating the Lagrange polynomials,

$$\omega_{m,j} = \int_{t_m}^{t_{m+1}} \left(\prod_{i=1, i\neq j}^{M} \frac{\tau - t_i}{t_s - t_i} \right) d\tau. \tag{3.10b}$$

In summary, with the discrete time points $\{t_m\}_{m=0}^M$ spanning the time interval $[0, T]$, IDC generates the approximate solution as

$$\begin{aligned}\boldsymbol{u}_{m+1}^{k+1} =& \boldsymbol{u}_m^{k+1} + \Delta t_m(1-\theta)[f(\boldsymbol{u}_m^{k+1}, t_m) - f(\boldsymbol{u}_m^k, t_m)] + \\ & \Delta t_m \theta [f(\boldsymbol{u}_{m+1}^{k+1}, t_{m+1}) - f(\boldsymbol{u}_{m+1}^k, t_{m+1})] + \sum_{j=1}^{M} \omega_{m,j} f(\boldsymbol{u}_j^k, t_j),\end{aligned} \tag{3.11}$$

where $k = 0, 1, \ldots, k_{\max} - 1$, and for each correction index k we sweep from left to right, i.e., $m = 0, 1, \ldots, M-1$. The choice of quadrature rule determines the maximum order of accuracy achievable in practice. If we use M uniformly distributed nodes with distance Δt, the maximal order of accuracy is of $\mathcal{O}(\Delta t^M)$ and, more specifically, we have

Theorem 3.1 ((Dutt et al. 2000)). *Suppose the time-integrator is of order p, such as $p = 1$ for $\theta = 1$ (Backward Euler) and $p = 2$ for $\theta = \frac{1}{2}$ (Trapezoidal Rule) in* (3.11)*. Then, the approximate solution $\{\boldsymbol{u}_m^k\}$ after k corrections is of order $\mathcal{O}(\Delta t^{\min\{M, (k+1)p\}})$.*

The original IDC method (Dutt et al. 2000) used Gauss nodes for the quadrature rule, resulting in a higher maximal order of accuracy. For instance, using Gauss-Lobatto nodes achieves an order of $2J - 1$. This IDC

variant is called *Spectral Deferred Correction* (SDC) and serves as the key component of the PFASST algorithm, introduced in Section 4.3 later.

For long-time computations, where T is large, creating a uniformly high-order numerical approximation across the entire time interval is challenging, because it is difficult to accurately approximate a function over a large interval with a single high-order polynomial. In such scenarios, it is natural to segment the time interval $[0, T]$ into multiple *windows*, denoted by $\{I_n := [T_{n-1}, T_n]\}_{n=1}^{N_t}$ with $T_0 = 0$ and $T_{N_t} = T$. Then, IDC can be applied to each window individually. Within each time window, a lower-order polynomial often provides precise quadrature, especially when the window size is small. However, this process is entirely sequential, since the computations for the $(n+1)$-st window I_{n+1} must await the completion of computations for I_n. This dependency arises because the initial value for I_{n+1} remains unknown until the preceding window's computations are finalized. Additionally, within each time window, we use a basic IDC and the computation proceeds step-by-step.

3.3.1. Pipeline IDC (PIDC)

The first parallel version of IDC, known as PIDC, was introduced in (Guibert and Tromeur-Dervout 2007). PIDC uses a *pipeline* parallelization approach for IDC. It is based on a simple concept applicable to any time evolution computation, already proposed by (Womble 1990). Specifically, the computation for the next window $I_{n+1} = [T_n, T_{n+1}]$ can start when a preliminary initial value from the current computation on $I_n = [T_{n-1}, T_n]$ at $t = T_n$ becomes available. For instance, following the first sweep on I_n, an approximation $\boldsymbol{u}_{n,M}^1$ at $t = T_n$, i.e. the rightmost solution on window I_n after one sweep, is obtained, and one can compute the first sweep on I_{n+1} using $\boldsymbol{u}_{n,M}^1$ as the initial value at the same time as performing the second sweep on I_n. After this computation, one can already start on I_{n+2} while continuing on I_{n+1} and I_n, computing three sweeps in parallel. In general, for N_t time windows, when conducting the k-th sweep on the n-th window I_n, we simultaneously perform the $(k-1)$-st sweep on I_{n+1}, the $(k-2)$-nd sweep on I_{n+2}, and so on, up to the first sweep on I_{n+k-1}. This procedure is illustrated in Figure 3.9 with $M = 6$ and $k_{\max} = 4$, where the first 4 sweeps represent the initial state. Following this stage, sweeps on I_n, I_{n+1}, I_{n+2}, and I_{n+3} are executed in parallel. On I_n (with $n \geq 1$), because each sweep starts from a rough and changing initial value, there is no guarantee that the accuracy of the generated solution increases as we proceed with the corrections. We illustrate this point by applying IDC and PIDC to the advection-diffusion equation (2.5) with two values of the diffusion parameter, $\nu = 1$ and $\nu = 10^{-3}$. We consider periodic boundary conditions and discretize with centered finite differences with a mesh size $\Delta x = \frac{1}{64}$, which leads to the linear system of ODEs (2.1), i.e., $\boldsymbol{u}'(t) = A\boldsymbol{u}(t)$ with $A = \frac{\nu}{\Delta x^2} A_{\mathrm{xx}} - \frac{1}{2\Delta x} A_{\mathrm{x}}$, where $\frac{\nu}{\Delta x^2} A_{\mathrm{xx}} \approx \nu \partial_{xx}$ and

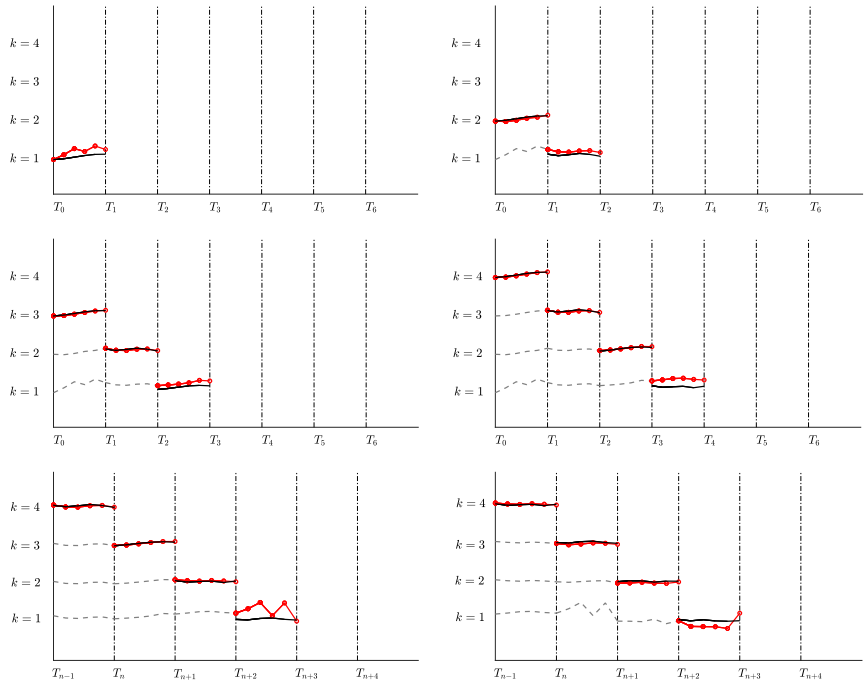

Figure 3.9. In PIDC with $k_{\max} = 4$, the sweeps on the 4 time windows can run simultaneously (bottom row), once the initialization phase in the first four windows (top and middle rows) is completed. The black dashed lines represent sweep histories, while the red solid lines marked with a red circle indicate current sweeps run in parallel. The black solid lines show the exact solution.

$\frac{1}{2\Delta x} A_{\mathrm{x}} \approx \partial_x$ are the discretization matrices given by

$$A_{\mathrm{xx}} = \begin{bmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{bmatrix}, \; A_{\mathrm{x}} = \begin{bmatrix} 0 & 1 & & & -1 \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ 1 & & & -1 & 0 \end{bmatrix}. \tag{3.12}$$

Let $T = 3$ and the window size be $\Delta T = \frac{1}{10}$. Then, using Backward Euler as time-integrator, we show in Figure 3.10 for IDC and PIDC with $M = 5$ the maximal relative error for each time window measured as

$$\mathrm{err}_n^k = \frac{\max_m \|\boldsymbol{u}_{n,m}^{\mathrm{ref}} - \boldsymbol{u}_{n,m}^k\|_\infty}{\max_{n,m} \|\boldsymbol{u}_{n,m}^{\mathrm{ref}}\|_\infty},$$

where the reference solution $\boldsymbol{u}_{n,m}^{\mathrm{ref}}$ is computed by the built-in solver `ODE45` in Matlab, using for both the relative and absolute tolerance $1e-13$. In each

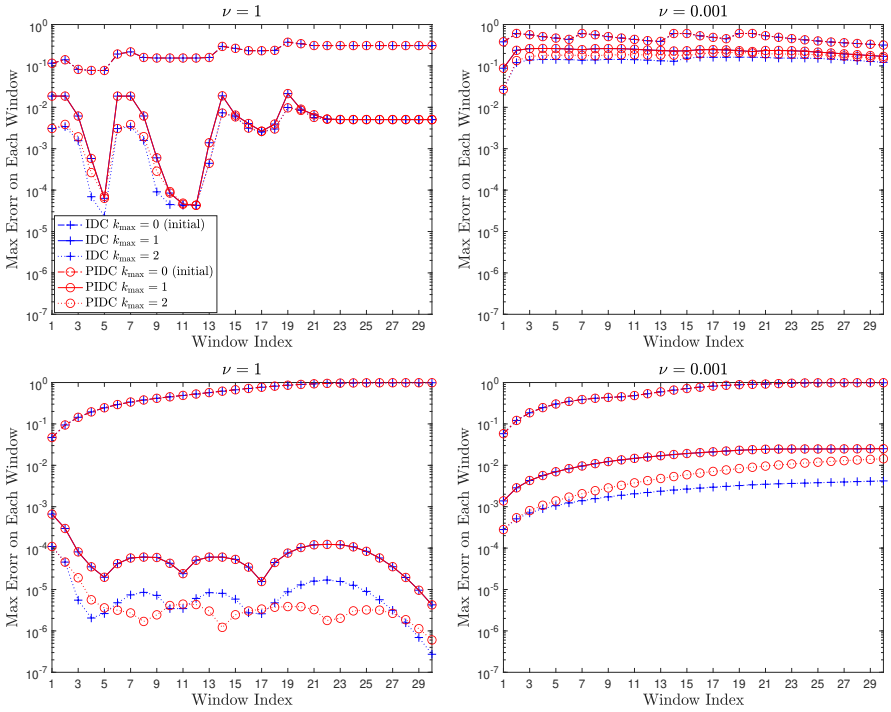

Figure 3.10. Maximum relative error on each time window for original IDC and its parallel version PIDC for the advection diffusion equation (2.5) with source function $g(x,t)$ in (2.4) with $\sigma = 1000$ (low regularity, top) and $\sigma = 5$ (higher regularity, bottom), and large diffusion parameter (left), and small diffusion parameter (right). The legend in the top left panel is also valid for the other panels.

panel, for both IDC and PIDC, we show the initial error and the errors after 1 and 2 sweeps. The initial guess on the $(n+1)$-th window $I_{n+1} = [T_n, T_{n+1}]$ is fixed simply as $\boldsymbol{u}^0_{n+1,m} \equiv \boldsymbol{u}^1_{n,M}$ for $j = 0, 1, \ldots, M$.

The results in Figure 3.10 show that for good performance of IDC and PIDC, the solution of the problem needs to be regular. In the first row, we used the source function $g(x,t)$ from (2.4) with parameter $\sigma = 1000$, which implies a δ-function type source, such that the solution is not regular enough. We see in the first panel that both IDC and PIDC perform similarly, and after the first correction the errors are not further reduced, the solution is not regular enough for a higher order approximation to perform well. In the second panel in the top row, we see that when the diffusion is becoming small, the improvement of the first IDC iteration is much worse than in the left panel, and a further iteration does also not help much, and similarly for PIDC. In the bottom row on the left, we see that if we use

a very regular source, (2.4) with parameter $\sigma = 5$, and thus the solution has enough regularity, both IDC and PIDC improve now for large diffusion in the second iteration as well, and PIDC is comparable to IDC. For small diffusion however at the bottom right, again performance is not as good, and PIDC performs clearly less well at the second iteration compared to IDC. These results indicate that for hyperbolic problems, if the solution is not regular enough, PIDC will not be very suitable for PinT computations.

3.3.2. Revisionist IDC (RIDC)

The RIDC method proposed by (Christlieb et al. 2010) is using a sliding IDC interval as a main new idea for more fine grained parallelization. To do so, consider a quadrature rule with M equidistant nodes. In RIDC, a first processor computes the initial approximation using a low order time stepper, like in IDC, but once it arrives at the end of the IDC interval after M steps, it does not stop, it just continues progressing in time computing step $M+1$, $M+2$ and so on. With the first M values of the first processor available, the second processor has now enough information to start the first IDC correction. Once it arrives at the end of the first IDC interval computing the correction for the M-th step, the third processor can start, but the second processor does not stop, it just continues by moving its IDC interval and associated quadrature formula one fine time step to the right, i.e. instead of using the approximations from the steps $1, 2, \ldots, M$ of the first processor, it considers the approximations from the steps $2, 3, \ldots, M+1$ of the first processor as its IDC interval and quadrature nodes, and computes with these its approximation for the $M+1$-st step. And then it considers the approximations from the steps $3, 4, \ldots, M+2$-nd of the first processor as its IDC interval, and computes with these as quadrature nodes the step $M+2$, and so on. Similarly, the third processor will also continue with a sliding IDC interval, and so on. Like PIDC, RIDC needs regularity to be effective, since it is a high order approximation technique, and thus for hyperbolic problems in the case of low regularity solutions, RIDC risks not to be very effective for PinT computations.

3.4. ParaExp

The fourth time parallel method we want to present is the ParaExp algorithm (Gander and Güttel 2013), which is a direct time parallel method that solves linear problems such as (2.1), which is the semi-discrete version of PDEs like the advection-diffusion equation (2.5) or the wave equation (2.7), see also (Merkel et al. 2017, Kooij et al. 2017). ParaExp uses special approximations of the matrix exponential function, and there are also other such techniques, like REXI (Schreiber, Peixoto, Haut and Wingate 2018), see also the early PinT methods based on Laplace transforms REXI (Schreiber et al. 2018).

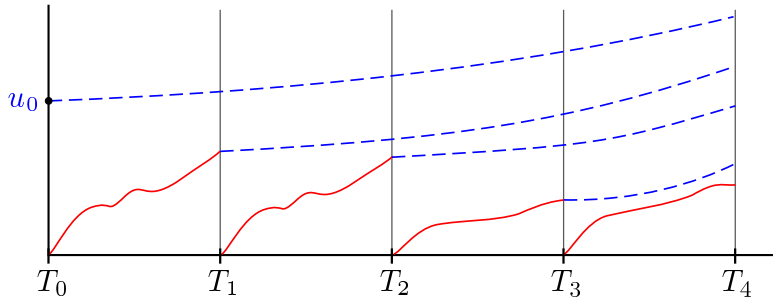

Figure 3.11. Two steps in the ParaExp solver.

ParaExp is based on a time decomposition, and performs two steps in order to construct the solution. First, on each time interval, the equation is solved in parallel with a source term but zero initial condition (red problems in Figure 3.11),

$$\boldsymbol{v}_n'(t) = A\boldsymbol{v}_n(t) + \boldsymbol{g}(t), \quad t \in (T_{n-1}, T_n], \quad \boldsymbol{v}_n(T_{n-1}) = 0, \tag{3.13}$$

where $n = 1, 2, \ldots, N_t$ and $T_{N_t} = T$. One then solves in parallel the linear equation (2.1) without source terms (blue problems in Figure 3.11), using as initial conditions the results from (3.13),

$$\boldsymbol{w}_n'(t) = A\boldsymbol{w}_n(t), \quad t \in (T_{n-1}, T], \quad \boldsymbol{w}_n(T_{n-1}) = \boldsymbol{v}_{n-1}(T_{n-1}), \tag{3.14}$$

where $n = 1, 2, \ldots, N_t$ and $\boldsymbol{v}_0(T_0) = \boldsymbol{u}_0$. The exact solution $\boldsymbol{u}(t)$ can then by linearity be constructed from the decoupled red and blue solutions,

$$\boldsymbol{u}(t) = \boldsymbol{v}_n(t) + \sum\nolimits_{j=1}^{n} \boldsymbol{w}_j(t), \quad t \in [T_{n-1}, T_n], \quad n = 1, 2, \ldots, N_t, \tag{3.15}$$

as one can see as follows: for $n = 1$, by adding (3.13) to (3.14) we have

$$(\boldsymbol{v}_1(t)+\boldsymbol{w}_1(t))' = A(\boldsymbol{v}_1(t)+\boldsymbol{w}_1(t))+\boldsymbol{g}(t),\ t \in (T_0, T_1],\ (\boldsymbol{v}_1(0)+\boldsymbol{w}_1(0)) = \boldsymbol{u}_0.$$

This proves (3.15) for $n = 1$. Now, suppose (3.15) holds for n, and we thus have

$$\boldsymbol{u}(T_n) = \boldsymbol{v}_n(T_n) + \sum\nolimits_{j=1}^{n} \boldsymbol{w}_j(T_n).$$

Then, in the next time interval $[T_n, T_{n+1}]$, since $\boldsymbol{w}_{n+1}(T_n) = \boldsymbol{v}_n(T_n)$ we have

$$\boldsymbol{u}(T_n) = \boldsymbol{w}_{n+1}(T_n) + \sum\nolimits_{j=1}^{n} \boldsymbol{w}_j(T_n) = \sum\nolimits_{j=1}^{n+1} \boldsymbol{w}_j(T_n).$$

Now, the function $\boldsymbol{w}(t)$ consisting of the first $n+1$ blue solutions, i.e., $\boldsymbol{w}(t) = \sum_{j=1}^{n+1} \boldsymbol{w}_j(t)$ satisfies $\boldsymbol{w}'(t) = A\boldsymbol{w}(t)$ for $t \in (T_n, T_{n+1}]$ and $\boldsymbol{w}(T_n) = \boldsymbol{u}(T_n)$, and hence $\boldsymbol{w}(t) + \boldsymbol{v}_{n+1}(t)$ satisfies the underlying problem (2.1) for $t \in [T_n, T_{n+1}]$, which proves (3.15) for $n+1$.

As illustrated in Figure 3.11, the computation of the red and blue solutions can be done in parallel for all time intervals. But the computation of the

blue problems (3.14) over longer and longer time intervals seems at first sight to be as expensive as the original problem (2.1). This is however not the case, since the blue problems are homogeneous, i.e., without a source term, and their solution is given by a matrix exponential,

$$\boldsymbol{w}_n(t) = \exp((t - T_{n-1})A)\boldsymbol{v}_{n-1}(T_{n-1}), \quad t \in [T_{n-1}, T], \tag{3.16}$$

where the computation time of the product between the matrix exponential and the vector $\boldsymbol{v}_{n-1}(T_{n-1})$ is independent of the value of t. There are many efficient and mature computational tools to approximate such solutions over long time (Higham 2008, Moler and Van Loan 2003), such as rational Krylov methods and Chebyshev expansions, and also the scaling and squaring algorithm with a Padé approximation (i.e., the built-in command '`expmv`' in MATLAB_R2023b or later versions). This latter approach is however more suitable for smaller matrices, for large sparse matrices the former approaches should be used. With efficient computations of the matrix exponential, using ParaExp can achieve high parallel efficiencies, up to 80% for the time parallelization of the wave equation (2.7), see (Gander and Güttel 2013). ParaExp is therefore an excellent time parallelization method for linear hyperbolic problems.

The ParaExp method described above is restricted to linear problems. An extension to nonlinear problems (2.2) was presented in (Gander et al. 2018*a*), assuming that there is a linear part of the nonlinear term such as

$$f(\boldsymbol{u}(t), t) = A\boldsymbol{u}(t) + B(\boldsymbol{u}(t)) + \boldsymbol{g}(t). \tag{3.17}$$

Following the idea in the linear case, we decouple the nonlinear problem (3.17) into a linear problem $\boldsymbol{w}'(t) = A\boldsymbol{w}(t)$ with $\boldsymbol{w}(0) = \boldsymbol{u}_0$ and a nonlinear problem $\boldsymbol{v}'(t) = B(\boldsymbol{v}(t) + \boldsymbol{w}(t)) + \boldsymbol{g}(t)$ with zero initial value $\boldsymbol{v}(0) = 0$. The sum $\boldsymbol{u}(t) = \boldsymbol{w}(t) + \boldsymbol{v}(t)$ then still solves (3.17), but the problems on the time intervals are now coupled: in $\{[T_{n-1}, T_n]\}_{n=1}^{N_t}$, the initial value of $\boldsymbol{w}(t)$ at $t = T_{n-1}$ depends on $\boldsymbol{v}(T_{n-1})$. To obtain parallelism in time, we need to iterate by first solving in parallel the linear problems

$$\begin{aligned} (\boldsymbol{w}_n^k)'(t) &= A\boldsymbol{w}_n^k(t), & t \in [T_{n-1}, T], \\ \boldsymbol{w}_n^k(T_{n-1}) &= \boldsymbol{v}_{n-1}^{k-1}(T_{n-1}), & \boldsymbol{w}_1^k(T_0) = \boldsymbol{u}_0, \end{aligned}$$

and then solving in parallel the nonlinear problems

$$\begin{aligned} (\boldsymbol{v}_n^k)'(t) &= A\boldsymbol{u}_n^k(t) + B(\boldsymbol{v}_n^k(t) + \textstyle\sum_{j=1}^n \boldsymbol{w}_j^k(t)) + \boldsymbol{g}(t), \quad t \in [T_{n-1}, T_n], \\ \boldsymbol{u}_n^k(T_{n-1}) &= 0, \end{aligned}$$

where $n = 1, 2, \ldots, N_t$. The k-th iterate solution is then defined by $\boldsymbol{u}_n^k(t) = \boldsymbol{v}_n^k(t) + \sum_{j=1}^n \boldsymbol{w}_j^k(t)$ for $t \in [T_{n-1}, T_n]$.

In the above nonlinear problems, the explicit dependence of B on $\sum_{j=1}^n \boldsymbol{w}_j^k(t)$ implies that we have to solve the linear problems on the entire interval $[T_{n-1}, T_n]$. This would be redundant and expensive if A is large. To avoid

this, we reformulate the iteration by rewriting $\boldsymbol{v}_n^k(t)$ as $\boldsymbol{v}_n^k(t) = \boldsymbol{u}_n^k(t) - \sum_{j=1}^n \boldsymbol{w}_j^k(t)$. In this new nonlinear version of the ParaExp algorithm we then solve for $n = 1, 2, \ldots, N_t$ sequentially

$$\begin{array}{rcll} (\boldsymbol{w}_n^k)'(t) & = & A\boldsymbol{w}_n^k(t), & t \in [T_{n-1}, T], \\ \boldsymbol{w}_n^k(T_{n-1}) & = & \boldsymbol{u}_{n-1}^{k-1}(T_{n-1}) - \sum_{j=1}^{n-1} \boldsymbol{w}_j^{k-1}(T_{n-1}) & \boldsymbol{w}_1^k(T_0) = \boldsymbol{u}_0, \end{array} \tag{3.18}$$

followed by solving in parallel the nonlinear problems

$$\begin{array}{rcl} (\boldsymbol{u}_n^k)'(t) & = & A\boldsymbol{u}_n^k(t) + B(\boldsymbol{u}_n^k(t)) + \boldsymbol{g}(t), \quad t \in [T_{n-1}, T_n], \\ \boldsymbol{u}_n^k(T_{n-1}) & = & \sum_{j=1}^n \boldsymbol{w}_j^k(T_{n-1}), \end{array} \tag{3.19}$$

and finally we form the approximate solution at the k-th iteration as

$$\boldsymbol{u}^k(t) = \boldsymbol{u}_n^k(t), \ t \in [T_{n-1}, T_n].$$

The non-linear ParaExp algorithm (3.18)-(3.19) has a finite step convergence property and a very interesting relation to the Parareal algorithm:

Theorem 3.3 ((Gander et al. 2018*a*)). *The iterate $\boldsymbol{u}^k(t)$ at the k-th iteration coincides with the exact solution $\boldsymbol{u}(t)$ for $t \in [0, T_k]$, i.e., the iterative ParaExp method converges in a finite number of steps. Moreover, at each time point T_n, the iterate $\boldsymbol{u}^k(t)$ also coincides with the solution generated by the Parareal algorithm*

$$\boldsymbol{U}_n^k = \mathcal{G}(T_{n-1}, T_n, \boldsymbol{U}_{n-1}^k) + \mathcal{F}(T_{n-1}, T_n, \boldsymbol{U}_{n-1}^{k-1}) - \mathcal{G}(T_{n-1}, T_n, \boldsymbol{U}_{n-1}^{k-1}), \tag{3.20a}$$

i.e., $\boldsymbol{u}_n^k = \boldsymbol{U}_n^k$ for $n = 0, 1, \ldots, N_t$, where the coarse propagator $\mathcal{G}(T_{n-1}, T_n, \boldsymbol{U})$ solves the linear problem

$$\boldsymbol{u}'(t) = A\boldsymbol{u}(t), \ \boldsymbol{u}(T_{n-1}) = \boldsymbol{U}, \ t \in [T_{n-1}, T_n], \tag{3.20b}$$

and the fine propagator $\mathcal{F}(T_{n-1}, T_n, \boldsymbol{U})$ solves the nonlinear problem

$$\boldsymbol{u}'(t) = A\boldsymbol{u}(t) + B(\boldsymbol{u}(t)) + \boldsymbol{g}(t), \ \boldsymbol{u}(T_{n-1}) = \boldsymbol{U}, \ t \in [T_{n-1}, T_n]. \tag{3.20c}$$

This is the first time that we see the *Parareal* algorithm, which we will discuss in detail in Section 4. The Parareal algorithm (3.20a)-(3.20c) is a simplified version since, for the standard version, the coarse propagator $\mathcal{G}$ also solves (3.20c). As will be discussed in Section 4, also the standard Parareal algorithm does not perform well for hyperbolic problems, and thus we cannot expect the simplified version to work well in this case. An illustration of this aspect is shown in Figure 3.12 for (2.2) with

$$f(\boldsymbol{u}(t), t) = A\boldsymbol{u}(t) + B\boldsymbol{u}^2(t), \ t \in (0, 2), \tag{3.21}$$

arising from semi-discretizing the 1D Burgers' equation with periodic boundary conditions using centered finite differences with a mesh size $\Delta x = \frac{1}{100}$, where $A = A_{\text{xx}}$ and $B = -\frac{1}{2}A_{\text{x}}$ with A_{xx} and A_{x} given in (3.12). For both ParaExp and Parareal, we use for the fine solver $\mathcal{F}$ Backward Euler with

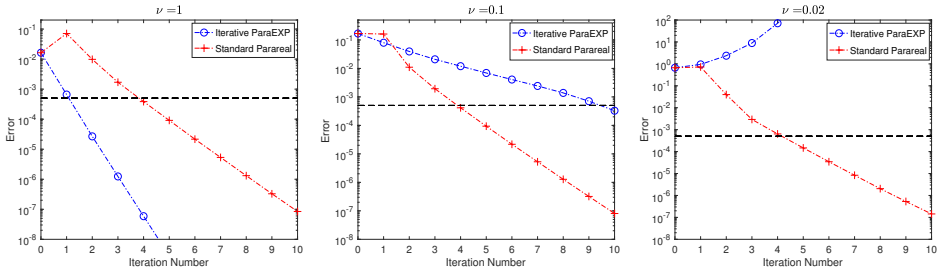

Figure 3.12. Convergence behavior of ParaExp and standard Parareal for Burgers' equation with diffusion parameter ν changing from large to small. In each panel, the transverse line indicates the order of the truncation error, $\max\{\Delta t, \Delta x^2\}$, where in practice one would stop the iteration.

a small step size $\Delta t = \frac{0.01}{20}$. For Parareal, we use for the coarse solver $\mathcal{G}$ Backward Euler as well but with a larger step size $\Delta T = 0.01$. For ParaExp, we use the built-in solver `expmv` in Matlab for the coarse propagator.

Clearly, for strongly diffusive problems, i.e., when ν is large, ParaExp converges very fast and the convergence rate is better than for standard Parareal. As ν decreases, standard Parareal converges faster than ParaExp, and particularly for $\nu = 0.02$, the latter diverges as shown in Figure 3.12 on the right. If we decrease ν further, then also standard Parareal will eventually diverge, and we will delve into this more in Section 4.

3.5. ParaDiag

The last technique we would like to explain is the ParaDiag family of methods, which is based on diagonalizing the time stepping matrix (or its approximation). There are two variants of ParaDiag depending on how we treat the time stepping matrix.

In the ParaDiag I family, which also represents a direct time parallel solver like ParaExp, one diagonalizes the time stepping matrix and then can solve each time step in parallel after diagonalization (Maday and Rønquist 2008). The diagonalization in ParaDiag I is however only possible when either using variable time step sizes, or by using a different time-integrator for the last step compared to the other ones, like in boundary value methods. For the case of variable step sizes, a detailed error analysis (Gander et al. 2019) shows that one can only use a limited number of time steps to parallelize, in double precision about 20, since one has to balance roundoff error with truncation error. Another shortcoming is that this direct ParaDiag method has only been explored for a few low-order time-integrators, such as Backward Euler and the trapezoidal rule. ParaDiag I with variable time step sizes is not easy to generalize to higher-order time-integrators, such as Runge-Kutta methods. When using a boundary value method type discretization,

the number of time steps one can parallelize in a single time window is greatly improved, but again only Backward Euler and the trapezoidal rule are applicable (Liu, Wang, Wu and Zhou 2022).

Both the limitations on the number of time steps and the time-integrator are overcome by the ParaDiag II family (Gander et al. 2021*c*). The key idea is to design a suitable approximation of the time stepping matrix and then to use it in a stationary iteration or as a preconditioner for a Krylov method, so one has to pay with iterations. The design principles for constructing such a preconditioner are twofold: its diagonalization should be well-conditioned in contrast to the ParaDiag I family of methods (i.e., the condition number of its eigenvector matrix should be small), and the iterations should converge fast, i.e. have small spectral radius, or equivalently the spectrum of the preconditioned matrix should be tightly clustered around 1 for Krylov acceleration. The first design principle ensures that round-off error arising from solving the preconditioning step via diagonalization is well controlled. The second design principle guarantees fast convergence of the preconditioned iteration. This iterative ParaDiag method was proposed in (McDonald et al. 2018) and independently in (Gander and Wu 2019). ParaDiag II techniques have been used as important components in new variants of Parareal and MGRiT which we will see in Section 4.2 and 4.4, and which in their original form work only well for parabolic problems. ParaDiag II techniques enhance the new Parareal and MGRiT variants in two directions: they improve the speedup by making the coarse grid correction parallel (Wu 2018, Wu and Zhou 2019), and they can also make Parareal and MGRiT work well for hyperbolic problems (Gander and Wu 2020) by allowing the coarse and fine propagators using the same grids, as we will show in Section 4.5. The application of ParaDiag II to solve the forward-backward system arising in PDE constrained optimization can be found in (Wu, Wang and Zhou 2023, Wu and Liu 2020, Heinzelreiter and Pearson 2024), where ParaDiag II produces a parallel version of the matching Schur complement (MSC) preconditioner (Pearson, Stoll and Wathen 2012). Modifications and improvements of ParaDiag II can be found in (Gander and Lunet 2024) and (Liu and Wu 2022*a*).

ParaDiag methods are applicable to both parabolic and hyperbolic problems, but the mechanisms for the direct and iterative versions are completely different. In the following, we introduce the main theory for these two versions and illustrate them with numerical results for the advection-diffusion equation (2.5), Burgers' equation (2.6), and the wave equation (2.7).

3.5.1. Direct ParaDiag Methods (ParaDiag I)

Parallelization by diagonalization of the time stepping matrix, originally introduced in (Maday and Rønquist 2008), is based on a very simple idea: consider solving the initial value problem (2.1), i.e. $\boldsymbol{u}' = A\boldsymbol{u} + \boldsymbol{g}(t)$ with

$A \in \mathbb{R}^{N_x \times N_x}$, by Backward Euler with variable step sizes,

$$\frac{\boldsymbol{u}_n - \boldsymbol{u}_{n-1}}{\Delta t_n} = A\boldsymbol{u}_n + \boldsymbol{g}_n, \quad n = 1, 2, \ldots, N_t, \tag{3.22}$$

with $\sum_{n=1}^{N_t} \Delta t_n = T$. Instead of solving these N_t difference equations one by one, we reformulate them as an *all-at-once* system, i.e., we solve all the solution vectors collected in $\boldsymbol{U} := (\boldsymbol{u}_1^\top, \boldsymbol{u}_2^\top, \ldots, \boldsymbol{u}_{N_t}^\top)^\top$ in one shot,

$$\mathcal{K}\boldsymbol{U} = \boldsymbol{b}, \quad \mathcal{K} := B \otimes I_x - I_t \otimes A, \tag{3.23a}$$

where $\otimes$ is the Kronecker product, $I_x \in \mathbb{R}^{N_x \times N_x}$ and $I_t \in \mathbb{R}^{N_t \times N_t}$ are identity matrices, and B is the time stepping matrix,

$$B = \begin{bmatrix} \frac{1}{\Delta t_1} & & & \\ -\frac{1}{\Delta t_2} & \frac{1}{\Delta t_2} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta t_{N_t}} & \frac{1}{\Delta t_{N_t}} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} \frac{1}{\Delta t_1}\boldsymbol{u}_0 + \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_{N_t} \end{bmatrix}. \tag{3.23b}$$

Since the time steps $\{\Delta t_n\}$ are all different from each other, we can diagonalize B,

$$B = VDV^{-1}, \quad D = \operatorname{diag}\left(\frac{1}{\Delta t_1}, \frac{1}{\Delta t_2}, \ldots, \frac{1}{\Delta t_{N_t}}\right). \tag{3.24}$$

Then, we can factor $\mathcal{K}$ in a block-wise manner as

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

This allows us to solve $\boldsymbol{U}$ from (3.23a) by the following three steps:

$$\begin{cases} \boldsymbol{U}^a = (V^{-1} \otimes I_x)\boldsymbol{b}, & \text{(step-a)} \\ \left(\frac{1}{\Delta t_n} I_x - A\right)\boldsymbol{u}_n^b = \boldsymbol{u}_n^a, \quad n = 1, 2, \ldots, N_t, & \text{(step-b)} \\ \boldsymbol{U} = (V \otimes I_x)\boldsymbol{u}^b, & \text{(step-c)} \end{cases} \tag{3.25}$$

where $\boldsymbol{U}^a := ((\boldsymbol{u}_1^a)^\top, \ldots, (\boldsymbol{u}_{N_t}^a)^\top)^\top$ and $\boldsymbol{U}^b := ((\boldsymbol{u}_1^b)^\top, \ldots, (\boldsymbol{u}_{N_t}^b)^\top)^\top$. Note that the first and last steps only involve matrix-vector multiplications and thus the computation is cheap. The major computation is step-b, but interestingly, all the N_t linear systems stemming from the time steps are independent and can be solved in parallel.

For an arbitrary choice of the step sizes $\{\Delta t_n\}$, we have to rely on numerical methods (e.g., `eig` in Matlab) to obtain the eigenvector matrix V. This does not bring significant computational burden since N_t does not need to be very large in practice, but it prevents us from performing a complete analysis of the method, such as studying the roundoff error and the selection of the parameters involved. The time mesh used in (Maday and Rønquist 2008) is a geometric mesh $\Delta t_n = \mu^{n-1}\Delta t_1$ with $\mu > 1$. They tested this direct

time-parallel method with $\mu = 1.2$ for the heat equation in 1D and obtained close to perfect speedup.

For prescribed T, the constraint $\sum_{n=1}^{N_t} \Delta t_n = \sum_{n=1}^{N_t} \mu^{n-1} \Delta t_1 = T$ specifies the first step size Δt_1 as $\Delta t_1 = \frac{T}{\sum_{n=1}^{N_t} \mu^{n-1}}$. This gives

$$\Delta t_n = \frac{\mu^{n-1}}{\sum_{n=1}^{N_t} \mu^{n-1}} T. \tag{3.26}$$

A large μ will produce large step sizes and thus large discretization error, while a small μ close to 1 produces large roundoff error when diagonalizing the time stepping matrix B, which can be understood by noticing that as μ approaches 1, the time stepping matrix B is close to a Jordan block, and diagonalizing such matrices results in large roundoff error. Therefore, it is important to know how to fix μ by balancing the discretization and roundoff error. This was carefully studied in (Gander et al. 2016*a*) for first-order parabolic problems, and in (Gander et al. 2019) for the second-order wave equation. In what follows, we let $\mu = 1 + \varrho$ with $\varrho > 0$ being a small value and we revisit the main existing results for fixing ϱ.

Theorem 3.4 (first-order problem). *For the system of ODEs $\boldsymbol{u}' = A\boldsymbol{u} + \boldsymbol{g}$ with initial value $\boldsymbol{u}(0) = \boldsymbol{u}_0$ and $t \in [0, T]$, suppose $\sigma(A) \subset \mathbb{R}^-$ with $|\lambda(A)| \leq {\lambda_{\max}}^2$. Let $\boldsymbol{u}_{N_t}(\varrho)$ and $\boldsymbol{u}_{N_t}(0)$ be the numerical solutions at $t = T$ obtained by using Backward Euler with geometric step sizes and the uniform step size, respectively. Let $\{\tilde{\boldsymbol{u}}_n(\varrho)\}$ be the numerical solution computed by the diagonalization method* (3.25)*. Then, it holds that*

$$\begin{aligned} &\|\boldsymbol{u}_{N_t}(\varrho) - \boldsymbol{u}_{N_t}(0)\| \lesssim C(\lambda_* T, N_t)\varrho^2, \\ &\|\tilde{\boldsymbol{u}}_n(\varrho) - \boldsymbol{u}_n(\varrho)\| \lesssim \epsilon \frac{N_t^2(2N_t+1)(N_t+\lambda_{\max}T)}{\phi(N_t)} \varrho^{-(N_t-1)}, \end{aligned} \tag{3.27}$$

where $C(x, N_t) := \frac{N_t(N_t^2-1)}{24} r(x/N_t, N_t)$ with $r(\tilde{x}, N_t) := \left(\frac{\tilde{x}}{1+\tilde{x}}\right)^2 (1+\tilde{x})^{-N_t}$. Here ϵ is the machine precision[1] *and*

$$\phi(N_t) := \begin{cases} \frac{N_t}{2}! \left(\frac{N_t}{2} - 1\right)!, & \text{if } N_t \text{ is even}, \\ \left(\frac{N_t-1}{2}!\right)^2, & \text{if } N_t \text{ is odd}. \end{cases}$$

The quantity $\lambda_* := \frac{N_t \tilde{x}_*}{T}$ with $\tilde{x}_*$ being the maximizer of the function $r(\tilde{x}, N_t)$ for $\tilde{x} \in [0, \infty)$. The best choice of ϱ, denoted by ϱ_{opt}, is the quantity balanc-

[2] Here and hereafter, $\lambda(\cdot)$ and $\sigma(\cdot)$ denote an arbitrary eigenvalue and the spectrum of the involved matrix.

[1] In the ISO C Standard, $\epsilon = 1.19 \times 10^{-7}$ for single precision and $\epsilon = 2.22 \times 10^{-16}$ for double precision.

ing the two error bounds in (3.27)*, i.e.,*

$$\varrho_{\text{opt}} = \left(\epsilon \frac{N_t^2 (2N_t + 1)(N_t + \lambda_{\max} T)}{\phi(N_t) C(\lambda_* T, N_t)} \right)^{\frac{1}{N_t+1}}. \tag{3.28}$$

Proof. Let $\lambda \in \sigma(A)$ be an arbitrary eigenvalue of A and consider the Dahlquist test equation $y' = \lambda y$. Then, the first estimate in (3.27) follows from the analysis in (Gander et al. 2016*a*, Theorem 2). For this test equation, the error due to diagonalization follows from the analysis in (Gander et al. 2016*a*, Theorem 6) and the bound of the error reaches its maximum when $|\lambda| = \lambda_{\max}$. □

The first estimate in (3.27) presents the truncation error between the use of a geometric time mesh and a uniform time mesh. From this, we can estimate the truncation error between $\boldsymbol{u}_{N_t}(\varrho)$ and the exact solution $\boldsymbol{u}(T)$ as $\|\boldsymbol{u}_{N_t}(\varrho) - \boldsymbol{u}(T)\| \le \|\boldsymbol{u}_{N_t}(\varrho) - \boldsymbol{u}_{N_t}(0)\| + \|\boldsymbol{u}_{N_t}(0) - \boldsymbol{u}(T)\|$, where the estimate of the last term is well understood and does not play a dominant role. The second estimate in (3.27) is the roundoff error due to diagonalization of the time stepping matrix B. The analysis of this error is closely related to the condition number of the eigenvector matrix V. With the geometric step sizes in (3.26), V and V^{-1} are lower triangular Toeplitz matrices, see (Gander et al. 2016*a*),

$$\begin{aligned} V &= \mathbb{T}(p_1, p_2, \ldots, p_{N_t-1}),\ p_n := \frac{1}{\prod_{j=1}^{n}(1-\varrho^j)}, \\ V^{-1} &= \mathbb{T}(q_1, q_2, \ldots, q_{N_t-1}),\ q_n = (-1)^n \varrho^{\frac{n(n-1)}{2}} p_n, \end{aligned} \tag{3.29a}$$

where $\mathbb{T}$ is the lower triangular Toeplitz operator

$$\mathbb{T}(a_1, a_2, \ldots, a_{N_t}) = \begin{bmatrix} 1 & & & \\ a_1 & 1 & & \\ \vdots & \ddots & \ddots & \\ a_{N_t-1} & \ldots & a_1 & 1 \end{bmatrix}. \tag{3.29b}$$

The closed form formula for V and V^{-1} in (3.29a) is useful to estimate $\text{Cond}(V)$, and then the roundoff error in (3.27). However, in practice, we do not use these formulas for V and V^{-1} in (3.25). Instead, we use the command `eig` in MATLAB to get V and V^{-1}, since it automatically optimizes the condition number by scaling the eigenvectors.

We now study the error of the ParaDiag I method for two PDEs with homogeneous Dirichlet boundary conditions and the initial value $u(x, 0) = \sin(2\pi x)$ for $x \in (0, 1)$, the heat equation (2.3) and the advection-diffusion equation (2.5) with $\nu = 10^{-2}$. Both PDEs are discretized by centered finite differences with mesh size $\Delta x = \frac{1}{50}$. With $T = 0.2$ and five values of N_t, we show in Figure 3.13 the error of ParaDiag I for $\varrho \in [10^{-2}, 1]$. The

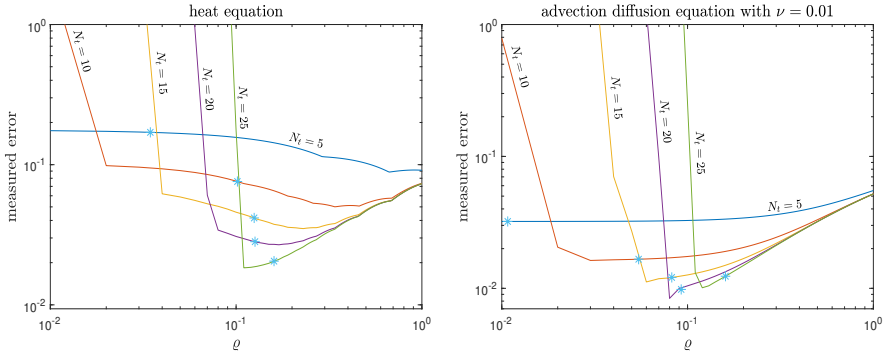

Figure 3.13. Measured error of ParaDiag I for five values of N_t using geometric step sizes like in (3.26) with $\mu = 1 + \varrho$ and $\varrho \in [10^{-2}, 1]$. The star denotes the theoretically estimated ϱ_{opt} from (3.28).

error is measured as $\max_{n=1,2,\ldots,N_t} \|\tilde{\boldsymbol{u}}_n(\varrho) - \boldsymbol{u}(t_n)\|_\infty$, where $\boldsymbol{u}(t_n)$ is the reference solution computed by the exponential integrator, i.e., $\boldsymbol{u}(t_n) = e^{-At_n}\boldsymbol{u}_0$ and $\tilde{\boldsymbol{u}}_n(\varrho)$ is the solution at $t = t_n$ by ParaDiag I. Clearly, there exists an optimal choice of ϱ that minimizes the error. We also show by a star for each N_t the theoretically estimated ϱ_{opt} from (3.28). For the advection-diffusion equation, this ϱ_{opt} predicts the optimal choice very well, and also quite well for the heat equation, except when N_t is small, even though the theoretical estimate was just obtained by balancing roundoff and truncation error estimates asymptotically. Let ϱ_{num} be the minimizer determined numerically as shown in Figure 3.13. We show in Figure 3.14 the error for Backward Euler with uniform step size $\Delta t = \frac{T}{N_t}$ and the ParaDiag I method using variable step sizes Δt_n, i.e., (3.26) with $\mu = 1 + \varrho_{num}$. Here $T = 0.5$ and $N_t = 2^{4:10}$. As N_t grows, the error for ParaDiag I decreases first and then increases rapidly when N_t exceeds some threshold less than 100.

We next consider a second-order problem, e.g., the wave equation (2.7) after space discretization,

$$\boldsymbol{u}''(t) = A\boldsymbol{u}(t) \text{ for } t \in (0, T],\ \boldsymbol{u}(0) = \boldsymbol{u}_0,\ \boldsymbol{u}'(0) = \tilde{\boldsymbol{u}}_0, \tag{3.30}$$

where $A \in \mathbb{R}^{N_x \times N_x}$. For the wave equation (2.7), A is a discretized Laplacian. To use ParaDiag I, we transform this equation into a first order system,

$$\boldsymbol{w}'(t) = \boldsymbol{A}\boldsymbol{w}(t) \text{ for } t \in (0, T], \boldsymbol{w}(0) = (\boldsymbol{u}_0^\top, \tilde{\boldsymbol{u}}_0^\top)^\top, \tag{3.31}$$

where $\boldsymbol{w}(t) = (\boldsymbol{u}^\top(t), (\boldsymbol{u}'(t))^\top)^\top$ and

$$\boldsymbol{A} = \begin{bmatrix} & I_x \\ A & \end{bmatrix}.$$

To avoid numerical dispersion, we use the Trapezoidal Rule as the time-

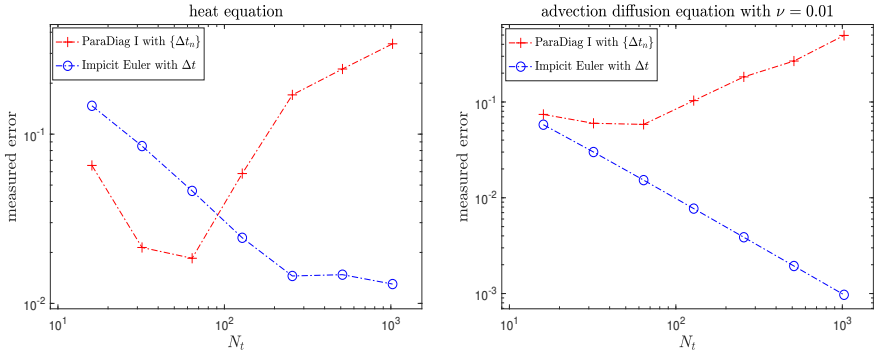

Figure 3.14. The error of ParaDiag I increases rapidly for N_t exceeding a threshold when using $\mu = 1 + \varrho_{num}$.

integrator,

$$\frac{\boldsymbol{w}_n - \boldsymbol{w}_{n-1}}{\Delta t_n} = \frac{\boldsymbol{A}}{2}(\boldsymbol{w}_n + \boldsymbol{w}_{n-1}),\ n = 1, 2, \ldots, N_t, \tag{3.32}$$

which is energy preserving in the sense that $\|\boldsymbol{w}_n\|_2 = \|\boldsymbol{w}_0\|_2$. The step sizes $\{\Delta t_n\}$ are again geometric as in (3.26) with some parameter $\mu = 1 + \varrho > 1$. Similar to (3.23a), we can represent (3.32) as an all-at-once system,

$$\mathcal{K}\boldsymbol{W} = \boldsymbol{b},\ \mathcal{K} = B \otimes I_x - \tilde{B} \otimes A, \tag{3.33a}$$

with some suitable vector $\boldsymbol{b}$, where B is the matrix in (3.23b) and

$$\tilde{B} = \frac{1}{2}\begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix}. \tag{3.33b}$$

To apply the diagonalization technique, we rewrite (3.33a) as

$$\mathcal{K}\boldsymbol{W} = \tilde{\boldsymbol{b}},\ \mathcal{K} = \tilde{B}^{-1}B \otimes I_x - I_t \otimes A,\ \tilde{\boldsymbol{b}} = (\tilde{B}^{-1} \otimes I_x)\boldsymbol{b}. \tag{3.34}$$

The matrix $\tilde{B}^{-1}B$ can be diagonalized, see (Gander et al. 2019),

$$\tilde{B}^{-1}B = V \operatorname{diag}\left(\frac{2}{\Delta t_1}, \ldots, \frac{2}{\Delta t_{N_t}}\right) V^{-1}, \tag{3.35a}$$

where V and V^{-1} are given by

$$\begin{aligned} V &= \mathbb{T}(p_1, p_2, \ldots, p_{N_t-1}),\ p_n := \prod_{j=1}^{n} \frac{1+\mu^j}{1-\mu^j}, \\ V^{-1} &= \mathbb{T}(q_1, q_2, \ldots, q_{N_t-1}),\ q_n := \mu^{-n} \prod_{j=1}^{n} \frac{1+\mu^{-j+2}}{1-\mu^{-j}}. \end{aligned} \tag{3.35b}$$

Then, the all-at-once system (3.34) can be solved via ParaDiag I (cf. (3.25)) as well.

Similar to the first-order equation studied above, by balancing the truncation error (between the geometric mesh and the uniform mesh) and the roundoff error, the best mesh parameter $\mu = 1 + \varrho_{\text{opt}}$ is obtained as follows.

Theorem 3.5 (second-order problem). *For the 2nd-order system of ODEs $\boldsymbol{u}'' = A\boldsymbol{u}$ with $\lambda(A) \le 0$, let $\boldsymbol{u}_{N_t}(\varrho)$ and $\boldsymbol{u}_{N_t}(0)$ denote the numerical solutions at $t = T$ obtained by using the Trapezoidal Rule with geometric time step sizes and the uniform time step size. Let $\{\tilde{\boldsymbol{u}}_n(\varrho)\}$ denote the numerical solution computed by the diagonalization method* (3.25)*. Then, it holds that*

$$\begin{aligned} \|\boldsymbol{u}_{N_t}(\varrho) - \boldsymbol{u}_{N_t}(0)\| &\lesssim \frac{N_t(N_t^2-1)}{15}\varrho^2, \\ \|\tilde{\boldsymbol{u}}_n(\varrho) - \boldsymbol{u}_n(\varrho)\| &\lesssim \epsilon \frac{2^{2N_t-\frac{1}{2}}N_t}{(N_t-1)!}\varrho^{-(N_t-1)}. \end{aligned} \tag{3.36}$$

The best choice of ϱ, denoted by ϱ_{opt}, is the quantity balancing the two error bounds in (3.36)*, i.e.,*

$$\varrho_{\text{opt}} = \left(\epsilon \frac{15 \times 2^{2N_t-\frac{1}{2}}}{(N_t^2-1)(N_t-1)!}\right)^{\frac{1}{N_t+1}}. \tag{3.37}$$

Proof. Let $\lambda > 0$ be an arbitrary eigenvalue of $-A$ and consider the scalar equation $u'' + \lambda u = 0$. Then, for small ϱ, according to (Gander et al. 2019, Theorem 2.1), the truncation error between using the geometric mesh and the uniform mesh is of order $\mathcal{O}(\frac{N_t(N_t^2-1)}{6} r_1(\frac{\lambda T}{2N_t})\varrho^2)$ with $r_1(s) = \frac{s^3}{(1+s^2)^2}$. For $s \ge 0$, it holds that $r_1(s) \le \frac{2}{5}$ and this gives the first estimate in (3.36). The second estimate follows from the roundoff error analysis in (Gander et al. 2019, Theorem 2.11), which is $\frac{2^{2N_t-\frac{1}{2}}N_t}{(N_t-1)!} r_2(\frac{\lambda T}{2N_t})$ with $r_2(s) = \frac{1}{1+s^2}$. For $s \ge 0$ it holds that $r_2(s) \le 1$. □

We show in Figure 3.15 (left) the error of ParaDiag I for 5 values of N_t when applied to the wave equation (2.7) with homogeneous Dirichlet boundary conditions. For each N_t, the parameter ϱ varies from 10^{-2} to 1. Here, $\Delta x = \frac{1}{20}$ and $T = 0.2$. Similar to the first-order parabolic problems (cf. Figure 3.13), there exists an optimal choice of ϱ which minimizes the error, and the theoretical estimate ϱ_{opt} is close to this choice. With ϱ_{num} denoting the best working parameter determined numerically for each N_t, we show in Figure 3.15 (right) the error for the geometric time mesh and the uniform time mesh. For the former, the error grows rapidly when $N_t > 32$.

For ParaDiag I, the increase in the error shown in Figures 3.14 and 3.15 (right) can be attributed to the poor condition number of the eigenvector matrix V of the time step matrices B and $\tilde{B}^{-1}B$. In Table 3.1, we show this

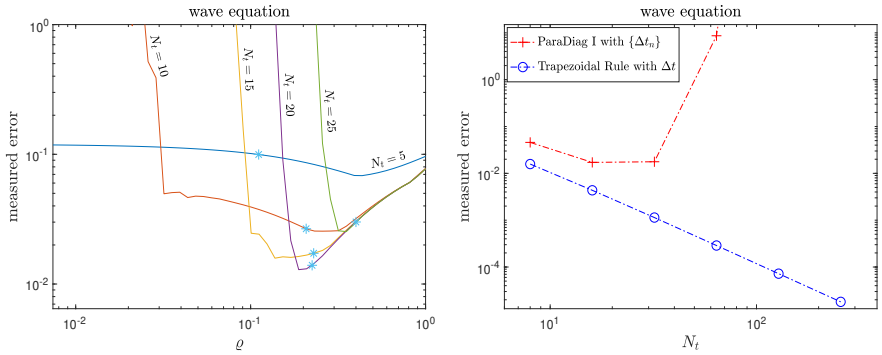

Figure 3.15. Measured error of ParaDiag I and the Trapezoidal Rule using uniform step sizes for the wave equation (2.7). The star in the left panel is the parameter ϱ_{opt} obtained in theory (cf. (3.37)).

Table 3.1. Cond(V) for B (Backward Euler) and $\tilde{B}^{-1}B$ (Trapezoidal Rule)

N_t	5	10	20	30	60	100
B	1.7×10^3	8.4×10^4	1.3×10^6	2.8×10^6	4.4×10^6	4.8×10^6
$\tilde{B}^{-1}B$	4.7×10^3	7.9×10^5	6.9×10^7	3.8×10^8	1.9×10^9	4.1×10^9

condition number for several values of N_t, where $\mu = 1 + \varrho_{num}$, and ϱ_{num} is determined numerically for each N_t by minimizing the error shown in Figures 3.13 and 3.15 on the left. As N_t increases, the condition number rises rapidly, which confirms our analysis of roundoff error very well (cf. (3.27) and (3.36)), but then reaches a plateau when using the numerically optimized parameter, and not the theoretically determined one, an observation that merits further study.

We now introduce another ParaDiag I method from (Liu et al. 2022), which addresses the limitation associated with N_t. Instead of using Backward Euler or the Trapezoidal Rule with geometric time step sizes, we use the same time step size Δt, but different methods, an idea which goes back to the boundary value technique (Axelsson and Verwer 1985). In this approach, we take for example centered finite differences for the first $(N_t - 1)$ time steps, followed by a final Backward Euler step. For the system of ODEs (2.1), this gives

$$\begin{cases} \frac{\boldsymbol{u}_{n+1} - \boldsymbol{u}_{n-1}}{2\Delta t} = A\boldsymbol{u}_n + \boldsymbol{g}_n, n = 1, 2, \ldots, N_t - 1, \\ \frac{\boldsymbol{u}_{N_t} - \boldsymbol{u}_{N_t-1}}{\Delta t} = A\boldsymbol{u}_{N_t} + \boldsymbol{g}_{N_t}. \end{cases} \tag{3.38}$$

It is important to note that this implicit boundary value technique time

discretization has quite different stability properties from traditional time discretizations; see e.g. (Gander 2015, Section 5.2). For the boundary value technique discretization (3.38), the all-at-once system is

$$\mathcal{K}\boldsymbol{U} = \boldsymbol{b}, \quad \mathcal{K} = B \otimes I_x - I_t \otimes A, \tag{3.39a}$$

where

$$B = \frac{1}{\Delta t}\begin{bmatrix} 0 & \frac{1}{2} & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & -1 & 1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} \frac{\boldsymbol{u}_0}{2\Delta t} + \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_{N_t} \end{bmatrix}. \tag{3.39b}$$

For the all-at-once system given by equation (3.39a), only the initial value $\boldsymbol{u}_0$ is required, and all time steps are solved simultaneously in one-shot.

The authors in (Axelsson and Verwer 1985) explored boundary value techniques to circumvent the well-known Dahlquist barriers between convergence and stability, which arise when using (3.38) in a time-stepping mode. In a general nonlinear case, they proved that the numerical solutions obtained simultaneously are of uniform second-order accuracy (Axelsson and Verwer 1985, Theorem 4), even though the last step is a first-order scheme. Even earlier, in (Fox 1954) and also (Fox and Mitchell 1957), such boundary value technique discretizations appeared already: instead of Backward Euler, the authors used the BDF2 method for the last step in (3.38),

$$\frac{3\boldsymbol{u}_{N_t} - 4\boldsymbol{u}_{N_t-1} + \boldsymbol{u}_{N_t-2}}{2\Delta t} = A\boldsymbol{u}_{N_t} + \boldsymbol{g}_{N_t}.$$

The method (3.38) is a prime example of the so-called *boundary value methods* (BVMs) developed a bit later, and the all-at-once system (3.39a) was carefully justified in (Brugnano, Mazzia and Trigiante 1993), see also (Brugnano and Trigiante 2003). In BVMs, the resulting all-at-once system is typically solved iteratively by constructing effective preconditioners.

A mathematical analysis of ParaDiag I based on BVM discretization like (3.38) can be found in (Liu et al. 2022):

Theorem 3.6. *The time stepping matrix B given by* (3.39b) *can be factored as $B = VDV^{-1}$ with* $\mathrm{Cond}(V) = \mathcal{O}(N_t^2)$[1].

ParaDiag I with BVM discretization can also be applied to second-order problems of the form $\boldsymbol{u}'' = A\boldsymbol{u}$, with initial values $\boldsymbol{u}(0) = \boldsymbol{u}_0$ and $\boldsymbol{u}'(0) = \tilde{\boldsymbol{u}}_0$. By setting $\boldsymbol{v}(t) := \boldsymbol{u}'(t)$ and $\boldsymbol{w}(t) := (\boldsymbol{u}^\top(t), \boldsymbol{v}^\top(t))^\top$, we can rewrite this

[1] Closed form formulas for V, V^{-1}, and D are provided in (Liu et al. 2022, Section 3).

equation as

$$\boldsymbol{w}'(t) = \boldsymbol{A}\boldsymbol{w}(t),\ \boldsymbol{A} := \begin{bmatrix} & I_x \\ A & \end{bmatrix},\ \boldsymbol{w}(0) := \begin{bmatrix} \boldsymbol{u}_0 \\ \tilde{\boldsymbol{u}}_0 \end{bmatrix}.$$

Then, similar to (3.38) the same time discretization scheme leads to

$$\begin{cases} \frac{\boldsymbol{w}_{n+1}-\boldsymbol{w}_{n-1}}{2\Delta t} = \boldsymbol{A}\boldsymbol{w}_n,\ n = 1, 2, \ldots, N_t - 1, \\ \frac{\boldsymbol{w}_{N_t}-\boldsymbol{w}_{N_t-1}}{\Delta t} = \boldsymbol{A}\boldsymbol{w}_{N_t}. \end{cases} \tag{3.40}$$

Rewriting the second order problem as a first order system doubles the storage requirement for the space variables at each time point, which is not desirable, especially if the second-order problem arises from a semi-discretization of a PDE in high dimensions or with small mesh sizes. To avoid this, one can write the all-at-once system for (3.40) using only $\boldsymbol{U} := (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{N_t})^\top$, which leads to

$$(B^2 \otimes I_x - I_t \otimes A)\boldsymbol{U} = \boldsymbol{b}, \tag{3.41}$$

where B is the matrix defined in (3.39b), and $\boldsymbol{b} := \left(\frac{\tilde{\boldsymbol{u}}_0^\top}{2\Delta t}, -\frac{\boldsymbol{u}_0^\top}{4\Delta t^2}, 0, \ldots, 0\right)^\top$. To see this, we trace the steps back at the discrete level which led to the first order system at the continuous level: from (3.40) we represent $\{\boldsymbol{u}_n\}$ and $\{\boldsymbol{v}_n\}$ separately as

$$\begin{cases} \frac{\boldsymbol{u}_{n+1}-\boldsymbol{u}_{n-1}}{2\Delta t} = \boldsymbol{v}_n,\ n = 1, 2, \ldots, N_t - 1, \\ \frac{\boldsymbol{u}_{N_t}-\boldsymbol{u}_{N_t-1}}{\Delta t} = \boldsymbol{v}_{N_t}, \end{cases}$$
$$\begin{cases} \frac{\boldsymbol{v}_{n+1}-\boldsymbol{v}_{n-1}}{2\Delta t} = A\boldsymbol{u}_n,\ n = 1, 2, \ldots, N_t - 1, \\ \frac{\boldsymbol{v}_{N_t}-\boldsymbol{v}_{N_t-1}}{\Delta t} = A\boldsymbol{u}_{N_t}. \end{cases}$$

Hence, with the matrix B in (3.39b), we have

$$(B \otimes I_x)\boldsymbol{U} - \boldsymbol{V} = \boldsymbol{b}_1, \quad (B \otimes I_x)\boldsymbol{V} - A\boldsymbol{U} = \boldsymbol{b}_2,$$

where $\boldsymbol{V} := (\boldsymbol{v}_1^\top, \ldots, \boldsymbol{v}_{N_t}^\top)^\top$, $\boldsymbol{b}_1 := (\frac{\boldsymbol{u}_0^\top}{2\Delta t}, 0, \ldots, 0)^\top$ and $\boldsymbol{b}_2 := (\frac{\tilde{\boldsymbol{u}}_0^\top}{2\Delta t}, 0, \ldots, 0)^\top$. From the first equation, we have $\boldsymbol{V} = (B \otimes I_x)\boldsymbol{U} - \boldsymbol{b}_1$. Substituting this into the second equation gives $(B \otimes I_x)^2\boldsymbol{U} - A\boldsymbol{U} = \boldsymbol{b}_2 + (B \otimes I_x)\boldsymbol{b}_1$. A routine calculation then yields $\boldsymbol{b}_2 + (B \otimes I_x)\boldsymbol{b}_1 = \boldsymbol{b}$, and combining this with $(B \otimes I_x)^2 = B^2 \otimes I_x$ gives the all-at-once system (3.41).

We now compare the ParaDiag I method with geometric time stepping to the one with BVM discretization applied to the wave equation (2.7) with homogeneous Dirichlet boundary conditions and $T = 0.5$, discretized in space using centered finite differences with $\Delta x = \frac{1}{40}$. The errors for $N_t = 2^2$ to 2^8 are shown in Figure 3.16 on the left. We see that the error of ParaDiag I with geometric time stepping shows the typical deterioration due to roundoff around $N_t = 32$, whereas ParaDiag I with BVM discretization is of order

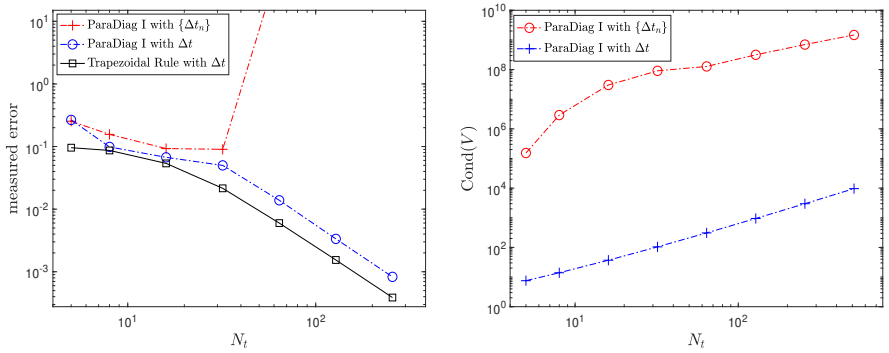

Figure 3.16. Left: measured error for the wave equation (2.7) using ParaDiag I with the Trapezoidal Rule and geometric step sizes $\{\Delta t_n\}$ and the BVM discretization (3.38) with uniform time step $\Delta t = T/N_t$. Right: condition number of the eigenvector matrix of the time-stepping matrix.

$\mathcal{O}(\Delta t^2)$ without any deterioration, like the Trapezoidal Rule. In Figure 3.16 on the right, we show the corresponding condition number of the eigenvector matrix of the time-step matrix, which shows that ParaDiag I with BVM discretization has a much lower condition number and explains why we do not observe any deterioration.

To conclude this section, we show how to apply ParaDiag I to nonlinear problems. We consider the first-order nonlinear system of ODEs (2.2), i.e., $\boldsymbol{u}'(t) = f(\boldsymbol{u}(t), t)$ with initial value $\boldsymbol{u}(0) = \boldsymbol{u}_0$, nonlinear second-order problems can be treated similarly. As in the linear case, the all-at-once system for this nonlinear problem is

$$(B \otimes I_x)\boldsymbol{U} - F(\boldsymbol{U}) = \boldsymbol{b}, \tag{3.42}$$

where $F(\boldsymbol{U}) := (f^\top(\boldsymbol{u}_1, t_1), f^\top(\boldsymbol{u}_2, t_2), \ldots, f^\top(\boldsymbol{u}_{N_t}, t_{N_t}))^\top$, and $\boldsymbol{b}$ is a suitable right hand side vector containing the initial condition and possible terms not depending on the solution. The time step matrix B is either the one given in (3.23b) using variable time steps, or the one given in (3.39b) corresponding to the BVM discretization (3.38).

Since the problem (3.42) is non-linear, we apply Newton's method,

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

which can be simplified to

$$(B \otimes I_x - \nabla F(\boldsymbol{U}^k))\boldsymbol{U}^{k+1} = \boldsymbol{b} - \left(\nabla F(\boldsymbol{U}^k)\boldsymbol{U}^k - F(\boldsymbol{U}^k)\right), \tag{3.43a}$$

where $k \geq 0$ is the Newton iteration index, and

$$\nabla F(\boldsymbol{U}^k) = \text{blkdiag}(\nabla f(\boldsymbol{u}_1^k, t_1), \ldots, \nabla f(\boldsymbol{u}_{N_t}^k, t_{N_t})), \tag{3.43b}$$

with $\nabla f(\boldsymbol{u}_n^k, t_n)$ being the Jacobian matrix of $f(\boldsymbol{u}, t_n)$ with respect to the

first variable $\boldsymbol{u}$. To make the diagonalization technique still applicable, we have to replace (or approximate) all the blocks $\{\nabla f(\boldsymbol{u}_n^k, t_n)\}$ by a single matrix A_k. Inspired by the idea in (Gander and Halpern 2017), we consider an averaged Jacobian matrix,

$$A_k := \frac{1}{N_t}\sum\nolimits_{n=1}^{N_t} \nabla f(\boldsymbol{u}_n^k, t_n) \quad \text{or} \quad A_k := \nabla f\left(\frac{1}{N_t}\sum\nolimits_{n=1}^{N_t} \boldsymbol{u}_n^k, \frac{T}{N_t}\right). \quad (3.44)$$

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

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

By substituting this into (3.43a), we obtain the quasi Newton method

$$(B \otimes I_x - I_t \otimes A_k)\boldsymbol{U}^{k+1} = \boldsymbol{b} - \Big((I_t \otimes A_k)\boldsymbol{U}^k - F(\boldsymbol{U}^k)\Big). \quad (3.45)$$

Convergence of such quasi Newton methods is well-understood; see, e.g., (Deuflhard 2004, Theorem 2.5) and (Ortega and Rheinboldt 2000).

In this quasi Newton method (3.45), the Jacobian system can also be solved parallel in time: with the diagonalization $B = VDV^{-1}$, we solve $\boldsymbol{U}^{k+1}$ in (3.45) again in three steps,

$$\begin{cases} \boldsymbol{U}^a = (V^{-1} \otimes I_x)\boldsymbol{r}^k, & \text{(step-a)} \\ (\lambda_n I_x - A_k)\boldsymbol{u}_n^b = \boldsymbol{u}_n^a, \ n = 1, 2, \ldots, N_t, & \text{(step-b)} \\ \boldsymbol{U}^{k+1} = (V \otimes I_x)\boldsymbol{U}^b, & \text{(step-c)} \end{cases} \quad (3.46)$$

where $\boldsymbol{r}^k := \boldsymbol{b} - ((I_t \otimes A_k)\boldsymbol{U}^k - F(\boldsymbol{U}^k)$. In the linear case, i.e., $f(\boldsymbol{u}, t) = A\boldsymbol{u} + \boldsymbol{g}(t)$, we have $A_k = A$ and $\boldsymbol{r}^k = \boldsymbol{b}$ and (3.46) reduces to (3.25).

The convergence rate of the quasi Newton method depends on the accuracy of the approximation of the average matrix A_k to all N_t Jacobian blocks $\nabla f(\boldsymbol{u}_n^k, t_n)$. One can imagine that if $\nabla f(u_n^k, t_n)$ changes dramatically for $n = 1, 2, \ldots, N_t$, any single matrix cannot be a good approximation. In this case, we can divide the time interval $[0, T]$ into multiple smaller windows and apply ParaDiag I to these time windows sequentially. We tested this approach by combining ParaDiag I with the BVM discretization (3.38) for Burgers' equation (2.6) with periodic boundary conditions. We discretized in space using centered finite differences with mesh size $\Delta x = 0.01$. In time, both the time step size Δt and the length of the time interval T were varied simultaneously, maintaining a fixed number of time steps, $N_t = \frac{T}{\Delta t} = 200$. In Figure 3.17, we present the convergence histories for two values of the diffusion parameter ν and several values of T. Note the dependence of the convergence rate on T, especially when ν is small. For $\nu = 0.1$, ParaDiag I has similar convergence rates when T increases, indicating that the Jacobian matrix $\nabla f(u, t)$ has smaller variations for $(t, u) \in \{(t_1, u_1), (t_2, u_n), \ldots, (t_{N_t}, u_{N_t})\}$.

The major computation in ParaDiag I is solving the N_t independent Jacobian systems in step-b of (3.46). Assuming the method reaches the stopping

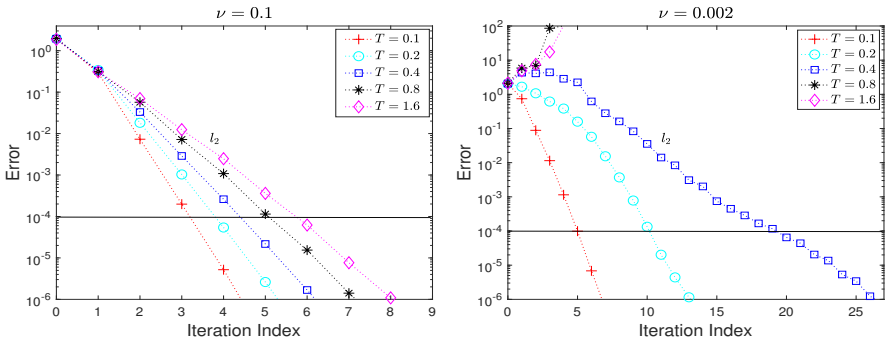

Figure 3.17. Error of ParaDiag I with BVM discretization (3.38) for Burgers' equation (2.6) with two values of the diffusion parameter ν. For each value of T, the number of time steps is fixed to $N_t = \frac{T}{\Delta t} = 200$. The horizontal line denotes the approximate space-time discretization error $\max\{\Delta t^2, \Delta x^2\} = 10^{-4}$.

Table 3.2. Number of total Jacobian solves for the sequential Trapezoidal Rule and ParaDiag I with BVM discretization in parallel

	T	0.1	0.2	0.4	0.8	1.6
$\nu = 0.1$	Trapezoidal Rule	401	401	403	419	443
	ParaDiag I	5	5	6	7	7
	T	0.1	0.2	0.4	0.8	1.6
$\nu = 0.002$	Trapezoidal Rule	400	446	476	460	526
	ParaDiag I	7	12	22	×	×

criterion after k iterations, the total number of Jacobian solves is k, given that we have access to N_t processors and each of them handles one Jacobian system in step-b. On the other hand, in a time-stepping mode, one would need to solve $\sum_{n=1}^{N_t} \mathrm{It}_n$ Jacobian systems, where It_n represents the number of Newton iterations performed at the n-th time step. Table 3.2 shows a comparison of the total number of parallel Jacobian solves in ParaDiag I with BVM discretization, and when using the trapezoidal rule sequentially. We see the clear computational advantage of ParaDiag I with BVM discretization, especially when convergence is rapid.

A recently proposed idea in (Liu and Wu 2022*b*, Section 3.3) to accelerate nonlinear ParaDiag II which we will see in the next section can also be used to accelerate nonlinear ParaDiag I: instead of using a single matrix A_k to approximate all the blocks $\{\nabla f(\boldsymbol{u}_n^k, t_n)\}$ (cf. (3.44)), we approximate $\nabla F(\boldsymbol{U}^k)$ by using a tensor structure matrix $\Phi_k \otimes A_k$ with a diagonal matrix

Φ_k determined by minimizing

$$\min_{\Phi_k=\mathrm{diag}(\phi_1,\phi_2,\ldots,\phi_{N_t})} \|\nabla F(\boldsymbol{U}^k) - \Phi_k \otimes A_k\|, \tag{3.47}$$

where A_k is the averaging matrix given in (3.44). For the Frobenius norm $\|\cdot\|_F$, the solution of this minimization problem is known as the *Nearest Kronecker product Approximation* (NKA), given by (Van Loan and Pitsianis 1993, Theorem 3)

$$\phi_n = \frac{\mathrm{trace}((\nabla f(\boldsymbol{u}_n^k, t_n))^\top A_k)}{\mathrm{trace}(A_k^\top A_k)}, \quad n = 1, 2, \ldots, N_t, \tag{3.48}$$

under the assumption that $\mathrm{trace}(A_k^\top A_k) > 0$. This leads to the quasi Newton iteration

$$(B \otimes I_x - \Phi_k \otimes A_k)\boldsymbol{U}^{k+1} = \boldsymbol{b} - \Big((\Phi_k \otimes A_k)\boldsymbol{U}^k - F(\boldsymbol{U}^k)\Big),$$

which, after multiplying both sides by the matrix $B^{-1} \otimes I_x$, can be represented as

$$(I_t \otimes I_x - B^{-1}\Phi_k \otimes A_k)\boldsymbol{U}^{k+1} = (B^{-1} \otimes I_x)(\boldsymbol{b} + F(\boldsymbol{U}^k)) - (B^{-1}\Phi_k \otimes A_k)\boldsymbol{U}^k.$$

By diagonalizing $B^{-1}\Phi_k$ as $V\mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{N_t})V^{-1}$ we can solve $\boldsymbol{U}^{k+1}$ via the 3-step diagonalization procedure (3.46) as well, where for step-b we now have to solve the linear systems

$$(I_x - \lambda_n A_k)\boldsymbol{u}_n^b = \boldsymbol{u}_n^a, \ n = 1, 2, \ldots, N_t.$$

So far there is no theory for the diagonalization of $B^{-1}\Phi_k$, but in practice this matrix is often diagonalizable and V is generally well conditioned.

In practice, it is better not to compute the scaling factors $\{\phi_n\}$ for each Newton iteration, which can be rather expensive due to the matrix-matrix multiplications in (3.48). It suffices to compute these quantities only once before starting the Newton iteration (i.e., as an offline task) by using a *reduced* model. For time-dependent PDEs, such a model could be a semi-discretized system of ODEs obtained by using a coarse space grid. We now show this idea for Burgers' equation (2.6) with periodic boundary conditions, discretized by a centered finite difference scheme with mesh size $\Delta x = \frac{1}{200}$. The scaling factors $\{\phi_n\}$ were determined by the Trapezoidal Rule with a coarse mesh size $\Delta X = \frac{1}{20}$. In Figure 3.18 we show for two values of the diffusion parameter ν how the error decays for the two quasi Newton versions of ParaDiag I with BVM discretization (3.38). For each ν we consider two time interval lengths, $T = 0.7$ and $T = 1.3$. Clearly, the quasi Newton version with NKA technique improves the convergence rate, especially when $T = 1.3$.

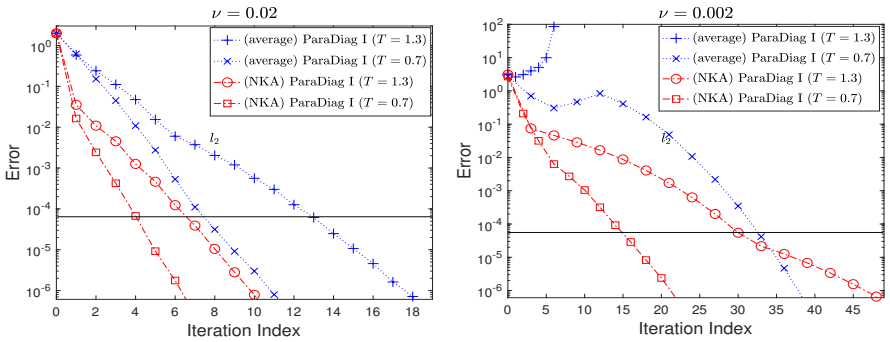

Figure 3.18. Error of the two quasi Newton versions of ParaDiag I with BVM discretization (3.38) for Burgers' equation (2.6) with two values of the diffusion parameter ν.

3.5.2. Iterative ParaDiag Methods (ParaDiag II)

It is difficult to generalize ParaDiag I to high-order time-integrators like multistage Runge-Kutta methods. This is not the case for the class of ParaDiag II methods, which use approximations of the time stepping matrices in order to make them diagonalizable, and then solve the all-at-once system by iteration.

A first member in the ParaDiag II class of methods was proposed in (McDonald et al. 2018) at the discrete level, and independently in (Gander and Wu 2019) at the continuous level. Although these two papers essentially describe the same method, the descriptions themselves are quite different. In (McDonald et al. 2018), the authors consider the approximate solution of the first-order linear system of ODEs (2.1) using a linear multistep method with m-steps,

$$\sum_{l=0}^{m} a_l \boldsymbol{u}_{n-l} = \Delta t \sum_{l=0}^{m} b_l (A\boldsymbol{u}_{n-l}) + \bar{\boldsymbol{g}}_n, \;\; n = 1, \ldots, N_t,$$

where one assumes that the first m initial values $\{\boldsymbol{u}_{-(m-1)}, \boldsymbol{u}_{-(m-2)}, \ldots, \boldsymbol{u}_0\}$ are given. The all-at-once system of these N_t difference equations is $\mathcal{K}\boldsymbol{U} = \boldsymbol{b}$ with $\boldsymbol{U} := (\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_{N_t}^\top)^\top$ and $\mathcal{K} := B_1 \otimes I_x - B_2 \otimes (\Delta t A)$, where $\boldsymbol{b}$ is a vector depending on the initial values and the source term $\boldsymbol{g}(t)$, and

$$B_1 := \begin{bmatrix} a_0 & & & & & \\ a_1 & a_0 & & & & \\ \vdots & \ddots & \ddots & & & \\ a_m & & \ddots & \ddots & & \\ & \ddots & & & a_1 & a_0 \\ & & a_m & \ldots & a_1 & a_0 \end{bmatrix}, \; B_2 := \begin{bmatrix} b_0 & & & & & \\ b_1 & b_0 & & & & \\ \vdots & \ddots & \ddots & & & \\ b_m & & \ddots & \ddots & & \\ & \ddots & & & b_1 & b_0 \\ & & a_m & \ldots & b_1 & b_0 \end{bmatrix}.$$

In (McDonald et al. 2018), this all-at-once system is solved with GMRES

using a preconditioner $\mathcal{P}$ for $\mathcal{K}$ obtained by replacing the two time stepping matrices B_1 and B_2 with two circulant matrices of Strang type, i.e.,

$$\mathcal{P} := C_1 \otimes I_x - C_2 \otimes (\Delta t A),$$

where

$$C_1 := \begin{bmatrix} a_0 & & a_m & \dots & a_1 & a_0 \\ a_1 & a_0 & & & & a_1 \\ \vdots & \ddots & \ddots & & \ddots & \vdots \\ a_m & & \ddots & \ddots & & a_m \\ & \ddots & & a_1 & a_0 & \\ & & a_m & \dots & a_1 & a_0 \end{bmatrix}, \; C_2 := \begin{bmatrix} b_0 & & b_m & \dots & b_1 & b_0 \\ b_1 & b_0 & & & & b_1 \\ \vdots & \ddots & \ddots & & \ddots & \vdots \\ b_m & & \ddots & \ddots & & b_m \\ & \ddots & & b_1 & b_0 & \\ & & b_m & \dots & b_1 & b_0 \end{bmatrix}.$$

For a theoretical understanding, or if the preconditioner $\mathcal{P}$ is very good (like multigrid for Poisson problems), one can use it directly in the stationary iteration

$$\mathcal{P}\Delta \boldsymbol{U}^k = \boldsymbol{r}^k := \boldsymbol{b} - \mathcal{K}\boldsymbol{U}^k, \; \boldsymbol{U}^{k+1} = \boldsymbol{U}^k + \Delta \boldsymbol{U}^k, \; k = 0, 1, \dots, \tag{3.49}$$

and the asymptotic convergence is fast if $\rho(\mathcal{P}^{-1}\mathcal{K}) \ll 1$. If convergence is not fast, this process can be accelerated using the preconditioner $\mathcal{P}$ within a Krylov method, i.e. solving the preconditioned linear system $\mathcal{P}^{-1}\mathcal{K}\boldsymbol{U} = \mathcal{P}^{-1}\boldsymbol{b}$ with a Krylov method, see (Ciaramella and Gander 2022, Section 4.1) for a simple introduction. This can even work when $\rho(\mathcal{P}^{-1}\mathcal{K}) \geq 1$, and is advantageous when the spectrum $\sigma(\mathcal{P}^{-1}\mathcal{K})$ is clustered.

The first advantage of using the block-circulant matrix $\mathcal{P}$ as preconditioner is that, similar to ParaDiag I, for each iteration, the preconditioning step $\mathcal{P}^{-1}\boldsymbol{r}^k$ can be solved via the diagonalization procedure, because any two circulant matrices C_1 and C_2 are commutative and therefore can be diagonalized simultaneously (Ng 2004, Chapter 4), i.e.

$$C_l = \mathrm{F}^* D_l \mathrm{F}, \; l = 1, 2,$$

where F is the discrete Fourier matrix defined as (F^* is the conjugate transform of F)

$$\mathrm{F} := \frac{1}{\sqrt{N_t}} \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \dots & \omega^{N_t-1} \\ \vdots & \vdots & \dots & \vdots \\ 1 & \omega^{N_t-1} & \dots & \omega^{(N_t-1)^2} \end{bmatrix}, \; \omega := \exp\left(\frac{2\pi \mathrm{i}}{N_t}\right), \tag{3.50}$$

and $D_l := \mathrm{diag}(\lambda_{l,1}, \lambda_{l,2}, \dots, \lambda_{l,N_t})$ contains the eigenvalues of C_l, i.e.,

$$D_l = \mathrm{diag}\left(\sqrt{N_t}\mathrm{F}C_l(:,1)\right), \; l = 1, 2. \tag{3.51}$$

Then, according to the property of the Kronecker product, we can factor

$\mathcal{P} = (\mathrm{F}^* \otimes I_x)(D_1 \otimes I_x - D_2 \otimes (\Delta t A))(\mathrm{F} \otimes I_x)$ and thus we can compute $\mathcal{P}^{-1}\boldsymbol{r}^k$ by performing again three steps:

$$\begin{cases} \boldsymbol{U}^a = (\mathrm{F} \otimes I_x)\boldsymbol{r}^k, & \text{(step-a)} \\ (\lambda_{1,n} I_x - \lambda_{2,n}\Delta t A)\boldsymbol{u}_n^b = \boldsymbol{u}_n^a,\ n = 1, 2, \ldots, N_t, & \text{(step-b)} \\ \boldsymbol{U} = (\mathrm{F}^* \otimes I_x)\boldsymbol{U}^b. & \text{(step-c)} \end{cases} \tag{3.52}$$

Here the first and last steps can be computed efficiently using the Fast Fourier Transform (FFT), with $\mathcal{O}(N_x N_t \log N_t)$ operations, and as in all ParaDiag methods, step-b can be computed in parallel, since all linear systems are completely independent of each other at different time points.

In (McDonald et al. 2018, Section 3), an important result about the clustering of eigenvalues of the preconditioned matrix $\mathcal{P}^{-1}\mathcal{K}$ was obtained when this ParaDiag II techniques is used to precondition a system for its solve by a Krylov method.

Theorem 3.7. *When $A \in \mathbb{R}^{N_x \times N_x}$ is symmetric negative definite, the preconditioned matrix $\mathcal{P}^{-1}\mathcal{K}$ has at most mN_x eigenvalues not equal to 1.*

This implies that GMRES converges in at most $mN_x + 1$ steps for the all-at-once system $\mathcal{K}\boldsymbol{U} = \boldsymbol{b}$ using $\mathcal{P}$ as preconditioner. Note however that when N_x is large, this result does not guarantee fast convergence of GMRES, and moreover, if A is not symmetric, the clustering of $\sigma(\mathcal{P}^{-1}\mathcal{K})$ becomes worse. To illustrate this, we consider three examples: the heat equation (2.3), the advection-diffusion equation (2.5) with two values of the diffusion parameter ν, and the second-order wave equation (2.7). We use homogeneous Dirichlet boundary conditions and the initial condition $u(x, 0) = \sin(2\pi x)$ for all PDEs, and for the wave equation, we set $\partial_t u(x, 0) = 0$.

The semi-discrete system of ODEs using centered finite differences is of the form (2.1) for the first order parabolic problems, and for the second-order wave equation we get

$$\boldsymbol{u}''(t) = A\boldsymbol{u}(t),\ \boldsymbol{u}(0) = \boldsymbol{u}_0,\ \boldsymbol{u}'(0) = 0,\ t \in (0, T], \tag{3.53}$$

where $A = \mathrm{Tri}[1 \quad -2 \quad 1]/\Delta x^2$. We solve the first-order system of ODEs (2.1) using the Trapezoidal Rule, and the second order system of ODEs (3.53) using a parametrized Numerov-type method (Chawla 1983),

$$\begin{cases} \tilde{\boldsymbol{u}}_n - \boldsymbol{u}_n + \gamma\Delta t^2 A(\boldsymbol{u}_{n+1} - 2\boldsymbol{u}_n + \boldsymbol{u}_{n-1}) = 0, \\ \boldsymbol{u}_{n+1} - 2\boldsymbol{u}_n + \boldsymbol{u}_{n-1} - \frac{\Delta t^2 A}{12}(\boldsymbol{u}_{n+1} + 10\tilde{\boldsymbol{u}}_n + \boldsymbol{u}_n) = 0, \end{cases} \tag{3.54}$$

where $\gamma > 0$ is a parameter. For $\gamma = 0$, (3.54) reduces to the classical Numerov method, which is a fourth-order method but only conditionally stable. With $\gamma \geq \frac{1}{120}$, this method is unconditionally stable and still fourth order.

Let $T = 2$, $\Delta t = \frac{1}{50}$, $\Delta x = \frac{1}{100}$, and $\gamma = \frac{1}{100}$. We show in Figure 3.19

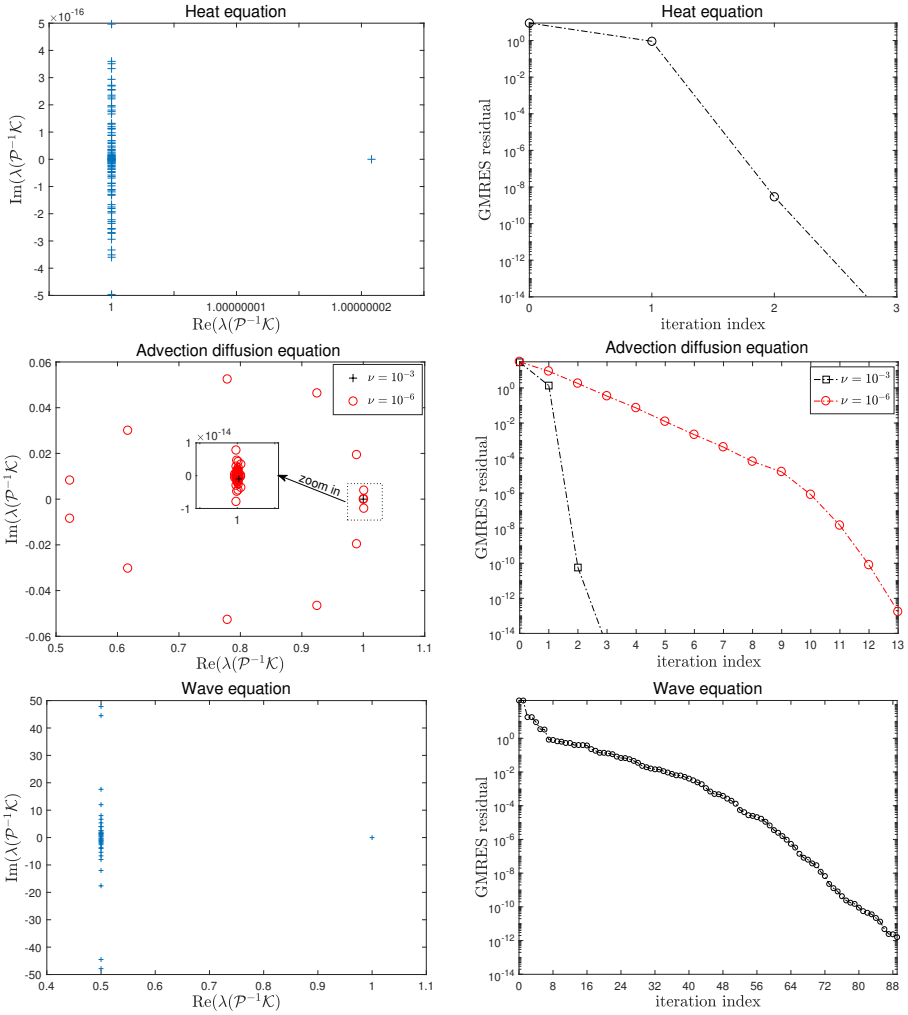

Figure 3.19. Spectra of the preconditioned matrix $\mathcal{P}^{-1}\mathcal{K}$ (left) and the measured convergence of preconditioned GMRES (right) for the three representative PDEs.

the eigenvalues of the preconditioned matrix $\mathcal{P}^{-1}\mathcal{K}$ for these three PDEs, and the decay of the residual as function of the iteration number of the preconditioned GMRES solver. We see that for a wide range of problems, the block-circulant matrix $\mathcal{P}$ is a good preconditioner, even for the advection-dominated diffusion equation with a small diffusion parameter $\nu = 10^{-3}$. If we continue however to reduce ν, the middle row of Figure 3.19 shows that the preconditioner $\mathcal{P}$ becomes worse, and ultimately, it loses its power when we switch to the hyperbolic problem represented by the wave equation.

During the same time, and independently of the work in (McDonald et al. 2018), another diagonalization-based time parallel method was proposed in (Gander and Wu 2019) within the framework of *waveform relaxation*. It is constructed at the continuous level by using a *head-tail* coupled condition,

$$\boldsymbol{u}_t^k(t) = A\boldsymbol{u}^k(t) + \boldsymbol{g}(t),\ \boldsymbol{u}^k(0) = \alpha[\boldsymbol{u}^k(T) - \boldsymbol{u}^{k-1}(T)] + \boldsymbol{u}_0, \tag{3.55}$$

where $\alpha \in \mathbb{C}$ is a free parameter. Upon convergence, we recover the solution to the initial value problem (2.1), i.e., $\boldsymbol{u}_t(t) = A\boldsymbol{u}(t) + \boldsymbol{g}(t)$ with $\boldsymbol{u}(0) = \boldsymbol{u}_0$. For the second-order problem (2.7), the iteration (3.55) can be defined similarly by reducing the problem to a first order system. The advantage of this iteration is that we can solve each iterate $\boldsymbol{u}^k(t)$ independently for all the time steps. To see this, we need to take a closer look at the structure of the discrete system stemming from (3.55). Suppose we discretize (3.55) using a one-step time-integrator specified by two matrices $r_1(\Delta tA)$ and $r_2(\Delta tA)$,

$$\begin{cases} r_1(\Delta tA)\boldsymbol{u}_n^k = r_2(\Delta tA)\boldsymbol{u}_{n-1}^k + \tilde{\boldsymbol{g}}_n,\ n = 1, \ldots, N_t, \\ \boldsymbol{u}_0^k = \alpha(\boldsymbol{u}_{N_t}^k - \boldsymbol{u}_{N_t}^{k-1}) + \boldsymbol{u}_0, \end{cases} \tag{3.56}$$

where $N_t = T/\Delta t$, $\boldsymbol{u}_0$ is a given initial value and $\tilde{\boldsymbol{g}}_n \in \mathbb{R}^{N_x}$ is a vector coming from the source term $\boldsymbol{g}(t)$. Examples for $r_1(\Delta tA)$ and $r_2(\Delta tA)$ are

$$\begin{cases} r_1 = I_x - \Delta tA,\ r_2 = I_x, & \text{Backward Euler}, \\ r_1 = I_x - \frac{1}{2}\Delta tA,\ r_2 = I_x + \frac{1}{2}\Delta tA, & \text{Trapezoidal Rule}. \end{cases} \tag{3.57}$$

By replacing $\boldsymbol{u}_0^k$ with $\alpha(\boldsymbol{u}_{N_t}^k - \boldsymbol{u}_{N_t}^{k-1}) + \boldsymbol{u}_0$ for $n = 1$, we can unfold (3.56) as

$$\begin{cases} r_1(\Delta tA)\boldsymbol{u}_1^k - \alpha r_2(\Delta tA)\boldsymbol{u}_{N_t}^k = \alpha r_2(\Delta tA)\boldsymbol{u}_{N_t}^{k-1} + r_2(\Delta tA)\boldsymbol{u}_0 + \tilde{\boldsymbol{g}}_1, \\ r_1(\Delta tA)\boldsymbol{u}_2^k - r_2(\Delta tA)\boldsymbol{u}_1^k = \tilde{\boldsymbol{g}}_2, \\ r_1(\Delta tA)\boldsymbol{u}_3^k - r_2(\Delta tA)\boldsymbol{u}_2^k = \tilde{\boldsymbol{g}}_3, \\ \vdots \\ r_1(\Delta tA)\boldsymbol{u}_{N_t}^k - r_2(\Delta tA)\boldsymbol{u}_{N_t-1}^k = \tilde{\boldsymbol{g}}_{N_t}. \end{cases}$$

We see that all the discrete unknowns $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{N_t}$ are coupled together and therefore we have to solve them in one shot. To this end, we represent these N_t equations as

$$\mathcal{P}_\alpha \boldsymbol{U}^k = \boldsymbol{b}^k, \tag{3.58a}$$

where $\boldsymbol{U}^k := ((\boldsymbol{u}_1^k)^\top, \ldots, (\boldsymbol{u}_{N_t}^k)^\top)^\top$ and $\boldsymbol{b}^k$ is a vector consisting of

$$\boldsymbol{b}^k := \boldsymbol{b} - \alpha \begin{bmatrix} r_2(\Delta tA)\boldsymbol{u}_{N_t}^{k-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix},\ \boldsymbol{b} := \begin{bmatrix} r_2(\Delta tA)\boldsymbol{u}_0 + \tilde{\boldsymbol{g}}_1 \\ \tilde{\boldsymbol{g}}_2 \\ \vdots \\ \tilde{\boldsymbol{g}}_{N_t} \end{bmatrix}. \tag{3.58b}$$

The matrix $\mathcal{P}_\alpha$ is given by

$$\begin{aligned}\mathcal{P}_\alpha &:= \begin{bmatrix} r_1(\Delta t A) & & & -\alpha r_2(\Delta t A) \\ -r_2(\Delta t A) & r_1(\Delta t A) & & \\ & \ddots & \ddots & \\ & & -r_2(\Delta t A) & r_1(\Delta t A) \end{bmatrix} \\ &= I_t \otimes r_1(\Delta t A) - C_\alpha \otimes r_2(\Delta t A), \end{aligned} \tag{3.58c}$$

$$C_\alpha := \begin{bmatrix} 0 & & & \alpha \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{N_t \times N_t}.$$

The matrix C_α is known as an α-circulant matrix, which, similarly to the standard circulant matrix where $\alpha = 1$, can be diagonalized by an eigenvector matrix V_α that depends on α only. Specifically, according to (Bini, Latouche and Meini 2005, Theorem 2.10), for an arbitrary α-circulant matrix C_α of Strang type, we have the spectral decomposition

$$C_\alpha = V_\alpha D_\alpha V_\alpha^{-1}, \tag{3.59a}$$

where the diagonal eigenvalue matrix and the eigenvector matrix V_α are given by

$$\begin{aligned} &D_\alpha = \operatorname{diag}(\sqrt{N_t}\mathrm{F}\Lambda_\alpha C_\alpha(:,1)), \\ &V_\alpha = \Lambda_\alpha \mathrm{F}^*,\ \Lambda_\alpha := \operatorname{diag}\left(1, \alpha^{-\frac{1}{N_t}}, \ldots, \alpha^{-\frac{N_t-1}{N_t}}\right), \end{aligned} \tag{3.59b}$$

and $C_\alpha(:,1)$ represents the first column of C_α. Using the property of the Kronecker product, we can factor $\mathcal{P}_\alpha$ as

$$\mathcal{P}_\alpha = (V_\alpha \otimes I_x)(I_t \otimes r_1(\Delta t A) - D_\alpha \otimes r_2(\Delta t A))(V_\alpha^{-1} \otimes I_x),$$

and hence again solve like in all ParaDiag methods for $\boldsymbol{U}^k$ in (3.58a) using three steps:

$$\begin{cases} \boldsymbol{U}^a = (V_\alpha^{-1} \otimes I_x)\boldsymbol{b}^k, & \text{(step-a)} \\ (r_1(\Delta t A) - \lambda_n r_2(\Delta t A))\boldsymbol{u}_n^b = \boldsymbol{u}_n^a,\ n = 1, 2, \ldots, N_t, & \text{(step-b)} \\ \boldsymbol{U}^k = (V_\alpha \otimes I_x)\boldsymbol{U}^b. & \text{(step-c)} \end{cases} \tag{3.60}$$

When $\alpha = 1$, the eigenvector matrix becomes the Fourier matrix, $V_\alpha = \mathrm{F}^*$, and hence this ParaDiag II method obtained from the discretization of the continuous formulation (3.55) coincides with (3.52) from (McDonald et al. 2018), and since $V_\alpha \otimes I_x = (\Lambda_\alpha \otimes I_x)(\mathrm{F}^* \otimes I_x)$ and $V_\alpha^{-1} \otimes I_x = (\mathrm{F} \otimes I_x)(\Lambda_\alpha^{-1} \otimes I_x)$, one can still use FFT techniques for the first and last step, also when $\alpha \neq 1$.

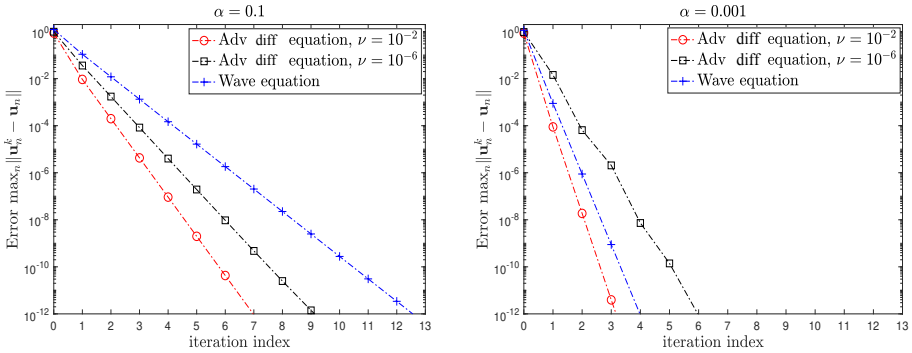

Figure 3.20. Error as a function of iteration for the head-tail coupled waveform relaxation method (3.55) using two values of the parameter α and the Trapezoidal Rule.

In (Gander and Wu 2019), the convergence of the waveform relaxation iterations (3.55) were examined at the continuous level, and it was shown that the error $\boldsymbol{u}^k(t) - \boldsymbol{u}(t)$ decays rapidly for both first-order and second-order problems, with a rate depending on α. We illustrate this in Figure 3.20, where the data for the two PDEs are the same as those used in Figure 3.19, but with periodic boundary conditions used here For this type of boundary conditions, the preconditioner $\mathcal{P}$ with the special choice $\alpha = 1$ as proposed in (McDonald et al. 2018) is singular and cannot be used, so we need to use $\alpha < 1$. We see that the introduction of the parameter α makes this ParaDiag II method a very powerful solver, which works also very well for highly advection dominated problems, and also the hyperbolic wave equation, and this without Krylov acceleration!

In (Gander and Wu 2019) two time-integrators were studied, Backward Euler and the Trapezoidal Rule, and it was shown that the discrete algorithm (3.56) preserves the convergence rate obtained from the analysis at the continuous level. The proof is technical and relies on a special representation of $r_1^{-1}(\Delta t A) r_2(\Delta t A)$ that appears to hold only for these two specific time-integrators (see (3.57) for the formulas of r_1 and r_2).

The head-tail coupled waveform relaxation method at the discrete level (3.56) can be represented as the *preconditioned* stationary iteration

$$\mathcal{P}_\alpha \Delta \boldsymbol{U}^{k-1} = \boldsymbol{r}^{k-1} := \boldsymbol{b} - \mathcal{K}\boldsymbol{U}^{k-1},\ \boldsymbol{U}^k = \boldsymbol{U}^{k-1} + \Delta \boldsymbol{U}^{k-1},\ k = 1, 2, \ldots, \quad (3.61)$$

where

$$\begin{aligned}\mathcal{K} &:= \begin{bmatrix} r_1(\Delta t A) & & & \\ -r_2(\Delta t A) & r_1(\Delta t A) & & \\ & \ddots & \ddots & \\ & & -r_2(\Delta t A) & r_1(\Delta t A) \end{bmatrix} \\ &= I_t \otimes r_1(\Delta t A) - B \otimes r_2(\Delta t A),\end{aligned} \tag{3.62a}$$

and $B \in \mathbb{R}^{N_t \times N_t}$ is a Toeplitz matrix,

$$B := \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix}. \tag{3.62b}$$

We thus see that the preconditioned iteration (3.61) with $\alpha = 1$ is precisely the method (3.49) of (McDonald et al. 2018). To see why the preconditioned iteration (3.61) equals the head-tail coupled waveform relaxation method (3.56), we notice that the vector $\boldsymbol{b}^k$ in (3.62b) can be represented as $\boldsymbol{b}^k = (\mathcal{P}_\alpha - \mathcal{K})\boldsymbol{U}^{k-1} + \boldsymbol{b}$, and substituting this into (3.62a) gives $\mathcal{P}_\alpha \boldsymbol{U}^k = (\mathcal{P}_\alpha - \mathcal{K})\boldsymbol{U}^{k-1} + \boldsymbol{b}$ which leads to (3.61).

Applying a one-step time-integrator specified by $r_1(\Delta t A)$ and $r_2(\Delta t A)$ to the initial value problem (2.1), i.e., $\boldsymbol{u}'(t) = A\boldsymbol{u}(t) + \boldsymbol{g}(t)$ with $\boldsymbol{u}(0) = \boldsymbol{u}_0$, leads to

$$r_1(\Delta t A)\boldsymbol{u}_n = r_2(\Delta t A)\boldsymbol{u}_{n-1} + \tilde{\boldsymbol{g}}_n,\ n = 1, \ldots, N_t. \tag{3.63}$$

Therefore, the matrix $\mathcal{K}$ is an all-at-once representation of these N_t difference equations, i.e., $\mathcal{K}\boldsymbol{U} = \boldsymbol{b}$. From this point of view, $\mathcal{P}_\alpha$ is a generalized block circulant preconditioner for $\mathcal{K}$.

For second-order problems of the form $\boldsymbol{u}''(t) = A\boldsymbol{u}(t) + \boldsymbol{g}(t)$ with initial values $\boldsymbol{u}(0) = u_0$ and $\tilde{\boldsymbol{u}}(0) = \tilde{\boldsymbol{u}}_0$, we can introduce $\boldsymbol{v}(t) = \boldsymbol{u}'(t)$ to transform them into a larger first-order system of ODEs, and then apply ParaDiag II. However, this approach doubles the memory requirements at each time step, which can be problematic in cases of very fine spatial mesh sizes or for high-dimensional problems. In that case, it can be preferable to discretize directly the second order problem, and we consider the symmetric two-step method

$$r_1(\Delta t^2 A)\boldsymbol{u}_{n+1} - r_2(\Delta t^2 A)\boldsymbol{u}_n + r_1(\Delta t^2 A)\boldsymbol{u}_{n-1} = \tilde{\boldsymbol{g}}_n,\ n = 1, \ldots, N_t - 1, \tag{3.64}$$

assuming that the second initial value $\boldsymbol{u}_1$ is given. Examples of the matrices

r_1 and r_2 are

$$r_1(\Delta t^2 A) = I_x - \frac{\Delta t^2 A}{12} + \frac{10\gamma(\Delta t^2 A)^2}{12},$$
$$r_2(\Delta t^2 A) = 2I_x + \frac{10\Delta t^2 A}{12} + \frac{20\gamma(\Delta t^2 A)^2}{12},$$

if we use the Numerov-type method from (3.54) as the time-integrator. For (3.64), the all-at-once matrix and the corresponding preconditioner are

$$\begin{aligned} \mathcal{K} &= \tilde{B} \otimes r_1(\Delta t^2 A) - B \otimes r_2(\Delta t^2 A), \\ \mathcal{P}_\alpha &= \tilde{C}_\alpha \otimes r_1(\Delta t^2 A) - C_\alpha \otimes r_2(\Delta t^2 A), \end{aligned} \tag{3.65a}$$

where B is the Toeplitz matrix from (3.62b), and C_α is the α-circulant matrix of B (see (3.58c)). The matrices $\tilde{B}$ and $\tilde{C}_\alpha$ are defined as

$$\tilde{B} := \begin{bmatrix} 1 & & & & \\ 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \end{bmatrix}, \quad \tilde{C}_\alpha := \begin{bmatrix} 1 & & & \alpha & \\ 0 & 1 & & & \alpha \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \end{bmatrix}. \tag{3.65b}$$

According to (3.59a)-(3.59b), we can simultaneously diagonalize C_α and $\tilde{C}_\alpha$. Thus, for the stationary iteration (3.61), we can solve the preconditioning step $\mathcal{P}_\alpha^{-1}\boldsymbol{r}^k$ using the diagonalization procedure (cf. (3.60)) as well.

The preconditioner $\mathcal{P}_\alpha$ used in the ParaDiag II method involves substituting the Toeplitz matrix within the all-at-once matrix $\mathcal{K}$ with a circulant (or α-circulant) matrix, while keeping the space matrices unchanged. This substitution, which approximates a pointwise Toeplitz matrix B by a circulant (or α-circulant) matrix C, is a natural approach that dates back to (Strang 1986). The spectrum of the preconditioned matrix $C^{-1}B$ has been extensively examined by researchers over the past three decades, yielding fruitful results; see the survey paper (Chan and Ng 1996) and the monographs (Ng 2004, Bini et al. 2005) for more details.

For blockwise Toeplitz matrices, where all blocks are Toeplitz (referred to as BTTB matrices), the circulant preconditioner is obtained by approximating each block by a circulant matrix, analogous to the approach used in ParaDiag II. Spectral analyses of such preconditioned matrices can be found in (Chan and Ng 1996) and (Ng 2004). However, in the context of ParaDiag II, the blocks (e.g., $r_1(\Delta t A)$ and $r_2(\Delta t A)$ in (3.57)) are not Toeplitz. In this scenario, there is a lack of systematic results regarding the eigenvalues of $\mathcal{P}_\alpha^{-1}\mathcal{K}$, and the work in (McDonald et al. 2018) explores this for $\alpha = 1$.

Since (McDonald et al. 2018) and (Gander and Wu 2019), a lot of efforts have been put into analyzing the spectrum of $\mathcal{P}_\alpha^{-1}\mathcal{K}$. Examples include (Gu and Wu 2020, Lin and Ng 2021, Wu and Zhou 2021*a*, Wu and Zhou 2021*b*,

Danieli, Southworth and Wathen 2022, Bouillon, Samaey and Meerbergen 2023, Heinzelreiter and Pearson 2024) for parabolic problems and (Danieli and Wathen 2021, Liu and Wu 2020) for hyperbolic problems. The analyses are intricate and rely heavily on special properties of the time-integrator, such as sparsity, Toeplitz structure, and diagonal dominance of the time stepping matrix.

A comprehensive spectral analysis of the preconditioned matrix $\mathcal{P}_\alpha^{-1}\mathcal{K}$ for both first-order and second-order problems can be found in (Wu, Zhou and Zhou 2022), with results that hold for any stable one-step time-integrator for first order systems of ODEs, and two-step symmetric time-integrator for second order systems of ODEs.

Theorem 3.8. *For the first-order system of ODEs $\boldsymbol{u}'(t) = A\boldsymbol{u}(t) + \boldsymbol{g}(t)$, if the one-step time-integrator* (3.63) *is stable, i.e., $|r_1^{-1}(z)r_2(z)| \leq 1$ for $z \in \sigma(\Delta t A) \subset \mathbb{C}^-$, then the eigenvalues of the preconditioned matrix satisfy*

$$\frac{1}{1-\alpha} \leq |\lambda(\mathcal{P}_\alpha^{-1}\mathcal{K})| \leq \frac{1}{1+\alpha}, \tag{3.66}$$

where $\mathcal{K}$ is the all-at-once matrix of the time-integrator (3.63)*, and $\mathcal{P}_\alpha$ is the block α-circulant matrix given by* (3.58c) *with $\alpha \in (0, 1)$. Similarly, for the second-order system of ODEs $\boldsymbol{u}''(t) = A\boldsymbol{u}(t) + \boldsymbol{g}(t)$, if the two-step method* (3.64) *is stable, i.e., $|r_1^{-1}(z)r_2(z)| \leq 2$ ($\forall z \in \sigma(\Delta t^2 A) \subset \mathbb{R}^-$ and $|r_1^{-1}(z)r_2(z)| = 2$ only if $z = 0$), then the eigenvalues of the preconditioned matrix $\mathcal{P}_\alpha^{-1}\mathcal{K}$ (with $\mathcal{P}$ and $\mathcal{K}$ given by* (3.65a)-(3.65b)*) also satisfy the bounds in* (3.66)*.*

For the stationary iteration (3.61), the iteration matrix $\mathcal{M}$ is given by

$$\mathcal{M} = \mathcal{I} - \mathcal{P}_\alpha^{-1}\mathcal{K}, \tag{3.67}$$

and based on (3.66), we get $\rho(\mathcal{M}) \leq \frac{\alpha}{1-\alpha}$. This explains the faster convergence of the ParaDiag II head-tail waveform relaxation method (3.55) when α is small, as we have seen in Figure 3.20. The stability of the underlying time-integrator serves as a sufficient condition for the eigenvalue bounds of the preconditioned matrix $\mathcal{P}_\alpha^{-1}\mathcal{K}$ in Theorem 3.8, or equivalently, the iteration matrix $\mathcal{M}$. Numerically, we find that stability is also a necessary condition. We illustrate this now for the Numerov-type method (3.54) applied to a second-order problem with A being a centered finite difference discretization of the Laplacian with Dirichlet boundary conditions, i.e., $\frac{1}{\Delta x^2}A \approx \partial_{xx}$. Setting $\Delta t = \frac{1}{16}$, $\Delta x = \frac{1}{128}$, and $\alpha = 0.02$, we show in Figure 3.21 the eigenvalues of the iteration matrix $\mathcal{M}$ for $T = 0.5, 10$, and 20, using the two values $\gamma = \frac{1}{120}$ and $\gamma = \frac{1}{120.01}$ for the Numerov-type method, where according to (Chawla 1983), $\gamma = \frac{1}{120}$ represents a stability threshold for the Numerov-type method. For this threshold value (top row of Figure 3.21), all eigenvalues of $\mathcal{M}$ lie within the theoretically analyzed circle. However, with

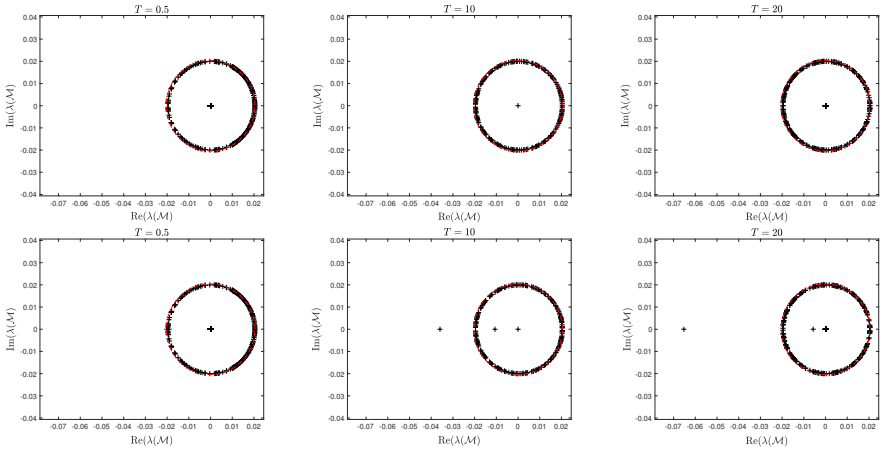

Figure 3.21. Eigenvalues of the iteration matrix $\mathcal{M}$ (cf. (3.67)) for the wave equation (2.7). Top row: $\gamma = \frac{1}{120}$. Bottom row: $\gamma = \frac{1}{120.01}$. In each panel, the dashed line represents the circle with radius $\frac{\alpha}{1-\alpha}$.

$\gamma = \frac{1}{120.01}$ (slightly below the threshold), the Numerov-type method looses unconditional stability, and the results in the bottom row clearly indicate that the eigenvalue bounds (3.66) no longer hold for large T.

The eigenvalue bounds (3.66) indicate that the ParaDiag II method converges faster when α decreases. This is true within a certain range of α, such as $\alpha \in [10^{-3}, 10^{-1}]$, as shown earlier (cf. Figure 3.20). However, α cannot be arbitrarily small due to roundoff errors arising from diagonalizing the α-circulant matrix. Specifically, for any diagonalizable square matrix P, floating point operations limit the precision of its factorization $P \approx VDV^{-1}$. For the α-circulant matrix C_α (cf. (3.58c)), even though its eigenvalues and eigenvectors have closed-form expressions, the difference between C_α and $V_\alpha D_\alpha V_\alpha^{-1}$ grows linearly as α decreases, the roundoff error $\mathrm{err}_{\mathrm{ro}}$ behaves like

$$\mathrm{err}_{\mathrm{ro}} = \mathcal{O}(\epsilon \mathrm{Cond}_2(V_\alpha)) = \mathcal{O}\left(\frac{\epsilon}{\alpha}\right),$$

where ϵ is the machine precision (e.g., $\epsilon = 2.2204 \times 10^{-16}$ for double precision), and the equality follows from (Gander and Wu 2019) and the fact that $V_\alpha = \Lambda_\alpha \mathrm{F}^*$ (cf. (3.59b)) with $\mathrm{Cond}_2(\mathrm{F}^*) = 1$, implying $\mathrm{Cond}_2(V_\alpha) = \frac{1}{\alpha}$. Interestingly, this does not necessarily imply a similar growth in the roundoff error of the ParaDiag II method. The error behavior depends on the implementation: directly solving for $\boldsymbol{U}^k$ as in (3.58a)-(3.58b) may lead to the mentioned growth, while first solving the error equation for $\Delta \boldsymbol{U}^{k-1}$ and then updating $\boldsymbol{U}^k$ (cf. (3.61)) can significantly mitigate the roundoff error; see Figure 3.22 for illustration. A comprehensive study of the roundoff error for ParaDiag II will appear in (Wu, Yang and Zhou 2024).

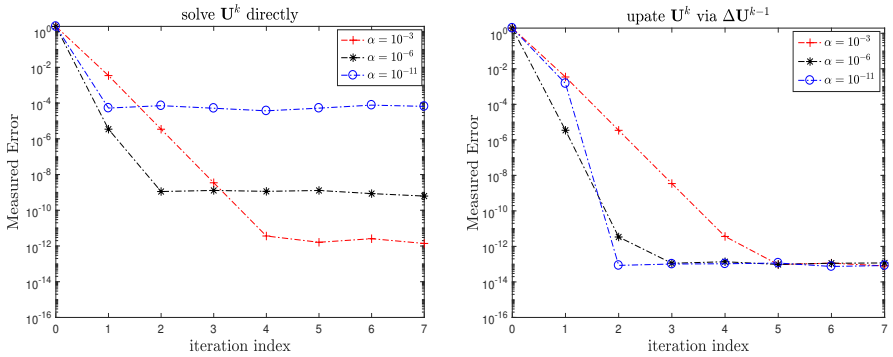

Figure 3.22. Measured error of ParaDiag II for the wave equation (2.7) implemented in two modes for three values of α.

So far, we have only considered one-step and symmetric two-step time-integrators. For general multistep methods, the eigenvalues of the preconditioned matrix $\mathcal{P}_\alpha^{-1}\mathcal{K}$ do not necessarily satisfy (3.66), yet we can demonstrate analogous results. For example the case where $\mathcal{K} = B \otimes I_x - I_t \otimes A$, with B being a *dense* lower triangular Toeplitz matrix was studied in (Gu and Wu 2020). This matrix arises in solving Volterra partial integro-differential equations, and its first column $\boldsymbol{\omega} = (\omega_0, \omega_1, \ldots, \omega_{N_t})$ is determined by the quadrature used to handle the integral term. The authors established a bound for the eigenvalues of the form $|\lambda(\mathcal{P}_\alpha^{-1}\mathcal{K})| = 1 + \mathcal{O}(\alpha)$, provided that the quantities $\{\omega_n\}$ satisfy certain conditions, such as positivity and monotonicity.

Turning to nonlinear problems, the application of ParaDiag II closely resembles that of ParaDiag I. We illustrate this for the first-order problem $\boldsymbol{u}'(t) = f(t, \boldsymbol{u}(t))$ discretized using Backward Euler with a step size Δt. Initially, we apply Newton's iteration to the nonlinear all-at-once system,

$$\mathcal{J}\Delta\boldsymbol{U}^l = \boldsymbol{b} - F(\boldsymbol{U}^l), \quad \boldsymbol{U}^{l+1} = \boldsymbol{U}^l + \Delta\boldsymbol{U}^l, \tag{3.68}$$

where the Jacobian matrix $\mathcal{J} := B \otimes I_x - \nabla F_l$ (cf. (3.43a)-(3.43b)), with

$$\nabla F_l = \mathrm{blkdiag}(\nabla f(\boldsymbol{u}_1^l, t_1), \ldots, \nabla f(\boldsymbol{u}_{N_t}^l, t_{N_t})),$$

$$B = \frac{1}{\Delta t}\begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}.$$

Then, we solve (3.68) with GMRES using $\mathcal{P}_\alpha = C_\alpha \otimes I_x - I_t \otimes A_l$ as preconditioner, where C_α is the α-circulant matrix of B and A_l is the average matrix of $\{\nabla f(\boldsymbol{u}_n^l, t_n)\}$. In general, we cannot use the stationary iteration (3.61) to solve the Jacobian system (3.68) since $\rho(\mathcal{P}_\alpha^{-1}\mathcal{J}) > 1$. However, the eigen-

values of $\mathcal{P}_\alpha^{-1}\mathcal{J}$ are clustered, which is good for GMRES. The eigenvalue distribution of $\mathcal{P}_\alpha^{-1}\mathcal{J}$ is influenced by the length of the time interval T, with a shorter T leading to more clustered eigenvalues and thus faster GMRES convergence. Numerical evidence supporting this aspect can be found in (Gander and Wu 2019) and (Wu et al. 2022). Additionally, we can leverage the nearest Kronecker product approximation introduced in Section 3.5.1 to accelerate convergence; see (Liu and Wu 2022*b*).

4. PinT methods designed for parabolic problems

We have shown in Section 2 intuitively why realizing PinT computations for hyperbolic problems is more challenging than for parabolic problems: parabolic problems tend to have local solutions in time, except for very low frequency components, whereas hyperbolic problems have highly non-local solutions in time, and this over all frequency components, from the lowest to the highest ones. Nevertheless, we have shown in Section 3 PinT methods that are effective for hyperbolic problems, and thus tackle all frequency components in a non-local way in time. Naturally, these methods perform then often even better when applied to parabolic problems, since they tackle all frequency components over long time, which includes the few very low frequency components that are highly non-local in time in parabolic problems. The methods we have seen so far were however often designed for linear problems, where they are most effective, whereas for nonlinear problems, they all suffer from certain drawbacks. For example, for OSWR it is not easy to determine the optimized Robin parameters, and without a reasonable parameter, the convergence rate can be quite poor. For ParaExp and ParaDiag I and II, non-linearity also affects the convergence rate of the Newton iteration used as an outer solver, and in particular, the Newton iteration may converge slowly or even diverge when the time interval is large. We show in this section now PinT methods that were designed for parabolic problems and take advantage of their properties to be local in time as we have seen in Section 2, and they work equally well for linear and non-linear problems. They have entirely different convergence mechanisms and properties from the methods in Section 3, and a direct application of these methods to hyperbolic problems often leads to slow convergence or even divergence.

4.1. Historical development

The first method we want to introduce is the Parareal algorithm from (Lions et al. 2001), which we have mentioned already in Section 3.4 to describe the nonlinear ParaExp variant. Even though Parareal was invented independently, it has its roots in earlier work on multiple shooting techniques for evolution problems, see (Bellen and Zennaro 1989, Chartier and

Philippe 1993), and the algorithm was presented already in (Saha et al. 1997) with a coarse model instead of a coarse grid in the context of solar system simulations, mentioning a relation to Waveform Relaxation. A very early precursor is even (Nievergelt 1964), although the method there is not iterative. Parareal, proposed 20 years ago, has attracted considerable attention in scientific and engineering computations. The convergence of Parareal is very well understood, see e.g. (Gander and Vandewalle 2007, Gander and Hairer 2008, Gander and Hairer 2014, Gander and Lunet 2024). In a sense, Parareal can be regarded as a template for developing more efficient PinT methods. There are numerous modifications of Parareal in the literature to make it applicable to different problems or for different purposes. Interesting examples are the *Parallel Implicit Time integration Algorithm* (PITA), see (Farhat and Chandesris 2003, Farhat, Cortial, Dastillung and Bavestrello 2006, Cortial and Farhat 2009), the *Parallel Full Approximation Scheme in Space-Time* (PFASST), see (Minion 2011, Emmett and Minion 2012, Minion, Speck, Bolten, Emmett and Ruprecht 2015*a*), *Multigrid Reduction in Time* (MGRiT), see (Falgout, Friedhoff, Kolev, MacLachlan and Schroder 2014, Dobrev, Kolev, Petersson and Schroder 2017*a*, Hessenthaler, Southworth, Nordsletten, Röhrle, Falgout and Schroder 2020), and also combinations of Parareal with ParaDiag (Wu 2018, Gander and Wu 2020). We present the convergence mechanisms and convergence properties of Parareal and its variants in this section. The basic feature of PinT methods based on Parareal are that they use two grids (or more) for the time discretization, while for space discretization they use just one grid. The idea of using multigrid in both space and time is going back to the *parabolic multigrid method* in (Hackbusch 1984) with an elegant analysis in the form of *multigrid waveform relaxation* in (Lubich and Ostermann 1987). Coarsening in time was not effectively possible in this approach, and important improvements using multigrid techniques for highly advective problems were proposed in (Vandewalle and Van de Velde 1994, Horton and Vandewalle 1995, Janssen and Vandewalle 1996, Van Lent and Vandewalle 2002). A new *Space-Time MultiGrid* (STMG) method using just standard components but as a new main ingredient a block Jacobi smoother in time was introduced and analyzed in (Gander and Neumüller 2016), and this is currently one of the most powerful PinT algorithms for parabolic problems, with excellent strong and weak scalability properties, see also (Neumüller and Smears 2019). We will introduce STMG at the end of this section, and show its effectiveness also for non-linear parabolic problems.

4.2. Parareal

The Parareal algorithm proposed in (Lions et al. 2001) is a non-intrusive time-parallel solver that is based on multiple shooting, although it was not

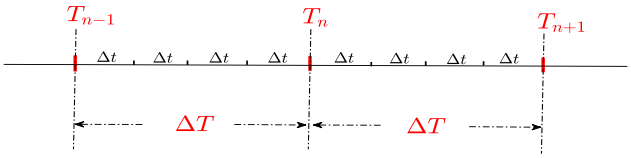

Figure 4.23. Parareal uses two time grids, where each large time step size ΔT contains J small time step sizes Δt.

invented in this context, but in the context of virtual control. In Parareal, the Jacobian in Newton's method used to solve the shooting equations is approximated by a finite difference across two iterates on a coarser grid or model (Gander and Vandewalle 2007). Similar to ParaExp, it is based on a time decomposition of the interval $(0, T)$ into several smaller time intervals $0 = T_0 < T_1 < \ldots < T_N = T$, with for example $T_n = T_0 + n\Delta t$. However, in contrast to ParaExp, it uses an iteration, starting with an initial guess $\boldsymbol{U}_n^0$ at T_n. For iteration index $k = 0, 1, \ldots$, Parareal computes improved approximations using the update formula

$$\boldsymbol{u}_{n+1}^{k+1} = \mathcal{F}(T_n, T_{n+1}, \boldsymbol{u}_n^k) + \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^{k+1}) - \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^k). \tag{4.1}$$

Here, $\mathcal{F}(T_n, T_{n+1}, \boldsymbol{u}_n^k)$ represents an accurate solver that uses a smaller step size Δt for the underlying evolution problem, with initial condition $\boldsymbol{u}_n^k$ at time $t = T_n$, yielding an approximate solution at time $t = T_{n+1}$. Similarly, $\mathcal{G}$ is a less expensive and less accurate solver that uses a larger step size, for example ΔT, or a simpler model, and the difference of the two $\mathcal{G}$ terms in (4.1) represents precisely the approximation of the Jacobian, see (Gander and Vandewalle 2007). Note that in (4.1), all the computationally expensive $\mathcal{F}$ solves can be performed in parallel, since at iteration k, the approximations $\boldsymbol{u}_n^k$ are all known. For simplicity, we consider here uniform fine and coarse time grids, assuming that each large step size contains J small steps, i.e., $\Delta T/\Delta t = J \geq 2$; see Figure 4.23. However, in principle, it is straightforward to apply non-uniform time grids in Parareal, see (Gander 2017, Maday and Mula 2020, Wu and Zhou 2024)).

The convergence of Parareal is well understood, both for linear problems (Gander and Vandewalle 2007) and non-linear problems (Gander and Hairer 2008): it converges super-linearly on bounded time intervals and linearly for parabolic problems on arbitrarily long time intervals. Specifically, for the linear problem (2.1), i.e., $\boldsymbol{u}'(t) = A\boldsymbol{u}(t) + \boldsymbol{g}(t)$ with initial value $\boldsymbol{u}(0) = \boldsymbol{u}_0$, and assuming that the fine and coarse solvers $\mathcal{G}$ and $\mathcal{F}$ are one-step time integrators with stability functions $\mathrm{R}_g(z)$ and $\mathrm{R}_f(z)$, we have the following convergence results:

Theorem 4.1. *Let $\{\boldsymbol{u}_n\}_{n=1}^{N_t}$ be the solutions computed sequentially by the*

$\mathcal{F}$ solver, $\boldsymbol{u}_{n+1} = \mathcal{F}(T_n, T_{n+1}, \boldsymbol{u}_n)$. Suppose the matrix A of the linear system of ODEs (2.1) is diagonalizable, $A = V_A D V_A^{-1}$, and the coarse solver $\mathcal{G}$ is stable, i.e., $|\mathrm{R}_g(z)| \le 1$ for $z \in \sigma(\Delta T A)$. Then Parareal satisfies the convergence estimate

$$\max_{1\le n\le N_t} \|V_A(\boldsymbol{u}_n^k - \boldsymbol{u}_n)\|_\infty \le \max_{z\in\sigma(\Delta T A)} \|M^k(z)\|_\infty \max_{1\le n\le N_t} \|V_A(\boldsymbol{u}_n^0 - \boldsymbol{u}_n)\|_\infty, \quad (4.2)$$

where $M(z)$ is a Toeplitz matrix given by two matrices $M_g(z)$ and $M_f(z)$,

$$\begin{aligned} M(z) &:= M_g^{-1}(z)[M_g(z) - M_f(z)], \\ M_g(z) &:= \begin{bmatrix} 1 & & & \\ -\mathrm{R}_g(z) & 1 & & \\ & \ddots & \ddots & \\ & & -\mathrm{R}_g(z) & 1 \end{bmatrix}, \\ M_f(z) &:= \begin{bmatrix} 1 & & & \\ -\mathrm{R}_f^J(z/J) & 1 & & \\ & \ddots & \ddots & \\ & & -\mathrm{R}_f^J(z/J) & 1 \end{bmatrix}. \end{aligned} \quad (4.3)$$

Proof. Applying the Parareal iteration (4.1) to the system of ODEs yields

$$\boldsymbol{u}_{n+1}^{k+1} = \mathrm{R}_f^J(\Delta T A/J)\boldsymbol{u}_n^k + \mathrm{R}_g(\Delta T A)\boldsymbol{u}_n^{k+1} - \mathrm{R}_g(\Delta T A)\boldsymbol{u}_n^k,\ n = 0, 1, \ldots, N_t - 1.$$

Since the overall fine solution computed sequentially by the $\mathcal{F}$ solver satisfies $\boldsymbol{u}_{n+1} = \mathcal{F}(T_n, T_{n+1}, \boldsymbol{u}_n)$, it also satisfies by adding and subtracting the same term

$$\boldsymbol{u}_{n+1} = \mathrm{R}_f^J(\Delta T A/J)\boldsymbol{u}_n + \mathrm{R}_g(\Delta T A)\boldsymbol{u}_n - \mathrm{R}_g(\Delta T A)\boldsymbol{u}_n.$$

The error $\boldsymbol{e}_n^k := \boldsymbol{u}_n - \boldsymbol{u}_n^k$ thus satisfies for $k \ge 0$ the error equation

$$\boldsymbol{e}_{n+1}^{k+1} = \mathrm{R}_g(\Delta T A)\boldsymbol{e}_n^{k+1} + [\mathrm{R}_f^J(\Delta T A/J) - \mathrm{R}_g(\Delta T A)]\boldsymbol{e}_n^k, \quad n = 0, 1, \ldots, N_t - 1.$$

For $n = 0$, $\boldsymbol{e}_0^k = 0$, since the initial value is known. Since $A = V_A D V_A^{-1}$, we have

$$\mathrm{R}_g(\Delta T A) = V_A \mathrm{R}_g(\Delta T D) V_A^{-1},\ \mathrm{R}_f^J(\Delta T A/J) = V_A \mathrm{R}_f^J(\Delta T D/J) V_A^{-1}.$$

Hence, we obtain the error equations in scalar form,

$$\xi_{n+1}^{k+1}(z) = \mathrm{R}_g(z)\xi_n^{k+1}(z) + [\mathrm{R}_f^J(z/J) - \mathrm{R}_g(z)]\xi_n^k(z), \quad n = 0, 1, \ldots, N_t - 1,$$

where $z = \Delta T\lambda$ with λ being an arbitrary eigenvalue of A and $\xi_n^k(z)$ is the element of $V_A \boldsymbol{e}_n^k$ corresponding to λ. Clearly, it holds that

$$\|V_A \boldsymbol{e}_n^k\|_\infty = \max_{z\in\sigma(\Delta T A)} |\xi_n^k(z)|. \quad (4.4)$$

Since $\xi_0^k(z) = 0$ for $k \ge 0$, we have $M_g(z)\boldsymbol{\xi}^{k+1}(z) = [M_g(z) - M_f(z)]\boldsymbol{\xi}^k(z)$,

which gives

$$\boldsymbol{\xi}^{k+1}(z) = M_g^{-1}(z)[M_g(z) - M_f(z)]\boldsymbol{\xi}^k(z),$$

where $\boldsymbol{\xi}^k(z) = (\xi_1^k(z), \xi_2^k(z), \ldots, \xi_{N_t}^k(z))^\top$ for $k \geq 0$. From (4.4) we have

$$\max_{1\leq n\leq N_t} \|V_A \boldsymbol{e}_n^k\|_\infty = \max_{z\in\sigma(\Delta T A)} \max_{1\leq n\leq N_t} |\xi_n^k(z)| = \max_{z\in\sigma(\Delta T A)} \|\boldsymbol{\xi}^k(z)\|_\infty,$$

which completes the proof of (4.2). □

From (4.2), we see that the norm $\|M^k(z)\|_\infty$ represents the convergence factor of the Parareal algorithm when applied to the Dahlquist test equation $u'(t) = \lambda u(t) + g(t)$, where λ is an arbitrary eigenvalue of A.

Remark 4.1. *From* (4.3) *we can interpret the Parareal algorithm from the perspective of a preconditioner by observing that*

$$M(z) = I_t - M_g^{-1}(z)M_f(z).$$

For the Dahlquist test equation $u'(t) = \lambda u(t) + g(t)$, the matrix $M_f(z)$ corresponds to the all-at-once matrix of the fine solver $\mathcal{F}$,

$$M_f(z)U = b,$$

where $U = (u_1, u_2, \ldots, u_{N_t})^\top$ and b is an appropriate vector. Parareal can thus be written as

$$M_g(z)\Delta U^k = r^k := b - M_f(z)U^k,\ U^{k+1} = U^k + \Delta U^k,$$

and the parallelization stems from computing the residual r^k: given U^k from the previous iteration, all components of r^k can be computed simultaneously as $r_n^k = b_n - (u_n^k - \mathcal{F}(T_{n-1}, T_n, u_{n-1}^k)) = b_n - (u_n^k - \mathrm{R}_f^J(z/J)u_{n-1}^k)$. This understanding is valuable for designing new variants of Parareal, and we will revisit this in Section 4.5.

Using $\|M^k(z)\|_\infty$ to predict the convergence behavior of Parareal is not convenient, so we introduce the results given in (Gander and Vandewalle 2007), which provide a very useful estimate of the convergence rate. This involves examining the structure of the matrix $M(z)$. Since

$$M_g^{-1}(z) = \begin{bmatrix} 1 & & & \\ \mathrm{R}_g(z) & 1 & & \\ \vdots & \ddots & \ddots & \\ \mathrm{R}_g^{N_t-1}(z) & \ldots & \mathrm{R}_g(z) & 1 \end{bmatrix},$$

we have

$$M(z) = [\mathrm{R}_f^J(z/J) - \mathrm{R}_g(z)]\tilde{M}(\mathrm{R}_g(z)),$$
$$\tilde{M}(\beta) := \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \\ \beta & 1 & 0 & & \\ \vdots & \ddots & \ddots & \ddots & \\ \beta^{N_t-2} & \dots & \beta & 1 & 0 \end{bmatrix}.$$

This implies that

$$\|M^k(z)\|_\infty = |\mathrm{R}_f^J(z/J) - \mathrm{R}_g(z)|^k \|\tilde{M}^k(\mathrm{R}_g(z))\|_\infty.$$

The infinity norm of the matrix $\tilde{M}^k$ was studied in (Gander and Vandewalle 2007, Lemma 4.4), and the main result is

$$\|\tilde{M}^k(\mathrm{R}_g(z))\|_\infty \le \begin{cases} \min\left\{ \left(\frac{1-|\mathrm{R}_g(z)|^{N_t-1}}{1-|\mathrm{R}_g(z)|}\right)^k, \binom{N_t-1}{k} \right\}, & \text{if } |\mathrm{R}_g(z)| < 1, \\ \binom{N_t-1}{k}, & \text{if } |\mathrm{R}_g(z)| = 1. \end{cases}$$

Substituting this into (4.2) leads to two different estimates of the convergence rate of the Parareal algorithm.

Theorem 4.2. *With the same notation and assumptions used in Theorem 4.1, the error of the k-th Parareal iteration satisfies*

$$\begin{aligned} \max_{1\le n\le N_t} \|\boldsymbol{e}_n^k\|_\infty &\le \max_{z\in\sigma(\Delta T A)} \varrho_s(J,z,N_t,k) \max_{1\le n\le N_t} \|\boldsymbol{e}_n^0\|_\infty, \\ \varrho_s(J,z,N_t,k) &:= \frac{|\mathrm{R}_g(z) - \mathrm{R}_f^J(z/J)|^k}{k!} \prod_{j=1}^{k} (N_t - j), \end{aligned} \tag{4.5a}$$

where $\boldsymbol{e}_n^k = V_A(\boldsymbol{u}_n^k - \boldsymbol{u}_n)$. *If* $|\mathrm{R}_g(z)| < 1 (\forall z \in \sigma(\Delta T A))$, *it holds that*

$$\begin{aligned} \max_{1\le n\le N_t} \|\boldsymbol{e}_n^k\|_\infty &\le \max_{z\in\sigma(\Delta T A)} \varrho_l^k(J,z) \max_{1\le n\le N_t} \|\boldsymbol{e}_n^0\|_\infty, \\ \varrho_l(J,z) &:= \frac{|\mathrm{R}_g(z) - \mathrm{R}_f^J(z/J)|}{1 - |\mathrm{R}_g(z)|}. \end{aligned} \tag{4.5b}$$

The estimate presented in (4.5a) indicates that Parareal converges superlinearly and completes iterations in at most N_t steps, since $\rho_s = 0$ when $k = N_t$. This estimate is particularly suitable for *short* time intervals where N_t is small. For larger N_t, ρ_s may not provide accurate predictions: initially, ρ_s increases, but the error actually decreases uniformly. This is illustrated in Figure 4.24 for the heat equation (2.3) with periodic boundary conditions, $g(x,t) = 0$, and initial value $u(x,0) = \sin^2(2\pi x)$ for $x \in (0,1)$. We use

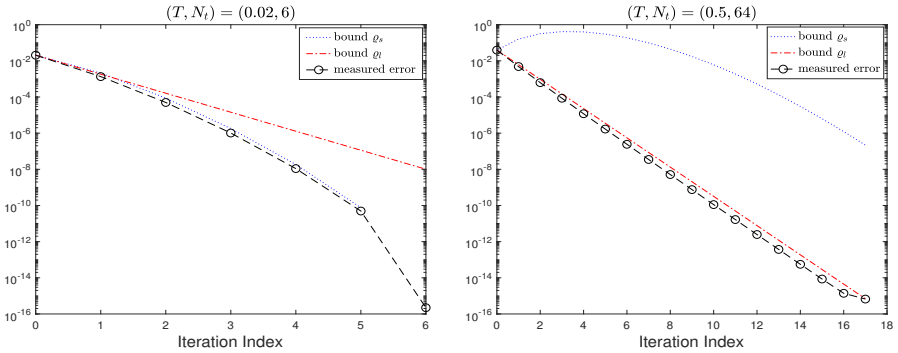

Figure 4.24. Parareal convergence for the heat equation showing its typical two convergence regimes: superlinear convergence over short time intervals and linear convergence over long time intervals.

in space a very large mesh size $\Delta x = \frac{1}{5}$, and both $\mathcal{F}$ and $\mathcal{G}$ use Backward Euler, with a coarsening factor $J = 10$. For $T = 0.02$ and $N_t = 6$, the error decreases at a superlinear rate, and ϱ_s accurately predicts this decrease. However, for a larger T and N_t, the error decreases linearly, and the prediction by ϱ_s is inaccurate. Note that for finer meshes in space, (e.g., $\Delta x = \frac{1}{8}$), Parareal converges linearly.

A convergence analysis of Parareal for nonlinear systems of ordinary differential equations using generating functions can be found in (Gander and Hairer 2008, Theorem 1), see also (Gander and Lunet 2024, Theorem 2.6).

Theorem 4.3. *Let $\mathcal{F}$ be the exact propagator and $\mathcal{G}$ be a time-integrator of order p with its local truncation error bounded by $C_3\Delta T^{p+1}$. Assume that $\mathcal{G}$ satisfies the Lipschitz condition*

$$\|\mathcal{G}(T_n, T_n + \Delta T, \boldsymbol{v}) - \mathcal{G}(T_n, T_n + \Delta T, \boldsymbol{w})\| \leq (1 + C_2\Delta T)\|\boldsymbol{v} - \boldsymbol{w}\|,$$

and the difference between $\mathcal{G}$ and $\mathcal{F}$ can be expressed, for small ΔT, as

$$\mathcal{F}(T_n, T_{n+1}, \boldsymbol{v}) - \mathcal{G}(T_n, T_{n+1}, \boldsymbol{v}) = c_{p+1}(\boldsymbol{v})\Delta T^{p+1} + c_{p+2}(\boldsymbol{v})\Delta T^{p+2} + \cdots,$$

where the coefficients $c_{p+1}, c_{p+2}, \ldots$ are continuously differentiable functions of $\boldsymbol{v}$. Then, the error of Parareal at iteration k is bounded by

$$\|\boldsymbol{u}(T_n) - \boldsymbol{u}_n^k\| \leq \frac{C_3\Delta T^{p+1}(C_1\Delta T^{p+1})^{k+1}}{(k+1)!}(1 + C_2\Delta T)^{n-k-1}\prod_{j=0}^{k}(n-j), \quad (4.6)$$

where $n = 1, 2, \ldots, N_t$ and C_1 is a constant related to the difference between $\mathcal{F}$ and $\mathcal{G}$.

The error estimate (4.6) has a similar consequence as the linear error estimate for short time intervals and small N_t (cf. (4.5a)): the product term

includes a factor of zero, resulting in an error bound of zero, indicating convergence in at most N_t steps. A detailed convergence analysis for Parareal applied to Hamiltonian systems using backward error analysis can be found in (Gander and Hairer 2014).

Parareal is highly effective for *diffusive* problems, such as the heat equation shown in Figure 4.24. In particular, for linear systems of ODEs of the form $\boldsymbol{u}'(t) = A\boldsymbol{u}(t) + \boldsymbol{g}(t)$, where A is a negative semi-definite matrix, it can be shown that Parareal has a constant convergence factor around 0.3 for arbitrarily large T and N_t, provided one uses Backward Euler[1] for $\mathcal{G}$ and $\mathcal{F}$ is an L-stable Runge-Kutta method.

Theorem 4.4. *If $\mathcal{G}$ is Backward Euler and $\mathcal{F}$ is an L-stable Runge-Kutta method, then*

$$\max_{z\in\mathbb{R}^-} \varrho_l(J,z) \approx 0.3, \quad \forall J \geq J_{\min}, \tag{4.7}$$

where $J_{\min} = \mathcal{O}(1)$.

Proof. For the case where $\mathcal{F}$ is Backward Euler, this result was established in (Mathew, Sarkis and Schaerer 2010). When $\mathcal{F}$ is the Trapezoidal Rule or BDF2 (i.e., Matlab's `ode23s` solver) or two singly diagonal implicit Runge-Kutta (SDIRK) methods, proofs can be found in (Wu 2015) and (Wu and Zhou 2015). For a general L-stable $\mathcal{F}$, the proof can be found in (Yang, Yuan and Zhou 2023). □

The origins of this result go back to a result already shown in (Gander and Vandewalle 2007, Table 5.1) at the continuous level, also for other coarse propagators, and contraction can even be much better, e.g. ≈ 0.068 for Radau IIA.

If $\mathcal{F}$ is only A-stable (not L-stable), for instance, the Trapezoidal Rule, Parareal does not always have a constant convergence factor. However, for large coarsening factors J, a similar result holds, namely

$$\max_{z\in[0,z_{\max}]} \varrho_l(J,z) \approx 0.3, \quad \forall J \geq J_{\min} = \mathcal{O}(\log^2(z_{\max})), \tag{4.8}$$

which was proved for the Trapezoidal Rule and a 4th-order Gauss Runge-Kutta method in (Wu and Zhou 2015). This differs significantly from the scenario where $\mathcal{F}$ is assumed to be the exact solution propagator (i.e., $\mathcal{F} = \exp(\Delta T A)$), where Parareal converges with a rate around 0.3 for $J \geq 2$.

We now illustrate this constant convergence factor by applying Parareal to the heat equation with periodic boundary conditions and discretization and problem parameters $\Delta x = \frac{1}{256}$, $\Delta T = 0.1$, $T = 4$, and $\nu = 0.1$ (the

[1] Note that using Backward Euler for $\mathcal{G}$ in Parareal is justified due to the need for a cheap and stable coarse grid correction.

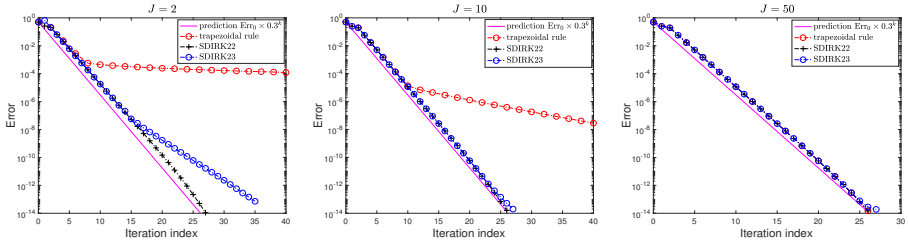

Figure 4.25. Different choices of the fine solver $\mathcal{F}$ lead to different convergence rates for Parareal.

diffusion coefficient). We use for $\mathcal{F}$ the Trapezoidal Rule and two SDIRK methods which are given by the Butcher tableau

$$\underbrace{\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-\gamma & \gamma \\ \hline & 1-\gamma & 1-\gamma \end{array}}_{\text{SDIRK22, } \gamma=\frac{2-\sqrt{2}}{2}}, \qquad \underbrace{\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & \frac{-1}{\sqrt{3}} & \gamma \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}}_{\text{SDIRK23, } \gamma=\frac{3+\sqrt{3}}{6}}. \tag{4.9}$$

Here, 'SDIRKsp' denotes an s-stage SDIRK method of order p. For SDIRK22, (4.7) holds for $J_{\min} = 2$ (Wu 2015), and for SDIRK23, $J_{\min} = 4$ (Wu and Zhou 2015). In Figure 4.25, we show the measured error at each iteration for three values of the coarsening factor J. We observe that for small J, these three time-integrators indeed lead to different convergence rates, especially for the Trapezoidal Rule and the SDIRK23 method, where Parareal converges more slowly. When J is large, say $J = 50$, Parareal converges at the similar rate closed to 0.3 for all three time-integrators. This can be intuitively understood by the fact that for small J the fine integrator which is not L-stable is not accurately enough resolving the physics for high frequencies, and Parareal tries to converge to this incorrect solution with Backward Euler as the coarse propagator which represents the correct physics for high frequencies.

While Parareal converges very well for the heat equation and more generally diffusive problems, Parareal is not well-suited for problems that are only weakly diffusive, since its convergence rate continuously deteriorates as the diffusion weakens. We illustrate this for the advection-diffusion equation (2.5) and Burgers' equation (2.6) with periodic boundary conditions, $g(x,t) = 0$, and initial condition $u(x,0) = \sin(2\pi x)$. We use $T = 4$, $\Delta T = 0.1$, $\Delta x = \frac{1}{128}$, and $J = 32$ for the problem and discretization parameters. The coarse solver is Backward Euler, and the fine solver is SDIRK22. For three values of the diffusion parameter ν, we show in Figure 4.26 the quantity $\varrho_l(J,z)$ for each $z = \Delta T\lambda(A)$. As ν decreases, the maximum of ϱ_l

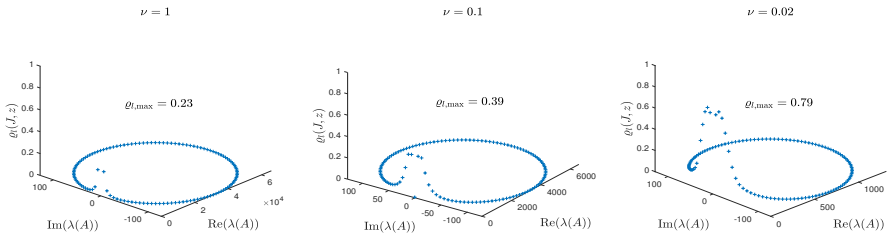

Figure 4.26. The quantity $\varrho_l(J, z)$ for each $z = \Delta T\lambda(A)$ for the advection-diffusion equation with three values of the diffusion parameter ν. As ν decreases, the maximum of ϱ_l approaches 1.

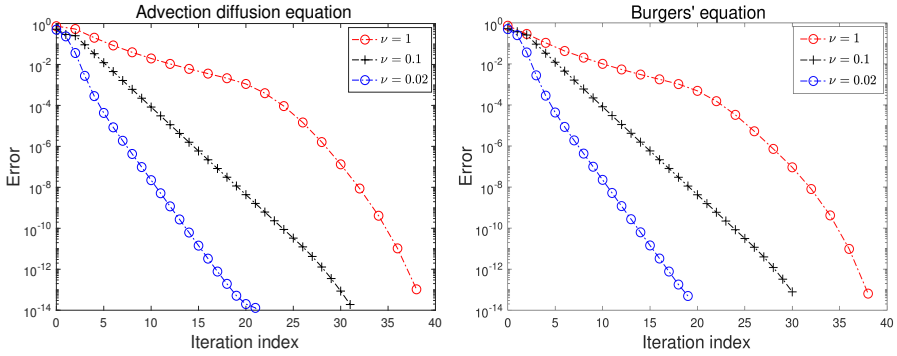

Figure 4.27. Convergence of Parareal applied to the advection-diffusion equation (left) and Burgers' equation (right) with three values of the diffusion parameter ν.

grows, indicating that Parareal converges more slowly. This is confirmed in Figure 4.27 on the left where we run Parareal on the corresponding problem. For Burgers' equation, we do not have an as precise theoretical analysis as for the advection-diffusion equation in Figure 4.26, but the results in Figure 4.27 on the right show that Parareal also converges more and more slowly for small ν. If we continue to reduce the parameter ν, meaning the advection term becomes increasingly dominant, Parareal eventually diverges, approximately when $\nu \leq 10^{-3}$, except that the finite step convergence still holds if one iterates long enough.

For hyperbolic problems, such as the second-order wave equation (2.7), Parareal is also not convergent, as was already shown in (Gander and Vandewalle 2007); see also (Gander and Lunet 2020*b*, Gander and Lunet 2020*a*, Gander, Lunet and Pogoželskytė 2023*a*, Gander, Lunet, Ruprecht and Speck 2023*b*) for more recent and detailed analyses. This degeneration can be attributed to the fact that for hyperbolic problems, as illustrated in Figure 2.5, arbitrarily small high frequency components, i.e. small oscillations, propagate

arbitrarily far in both space and time. Consequently, it becomes very challenging to achieve high-accuracy solutions in the coarse solver $\mathcal{G}$ that are comparable to the fine solver $\mathcal{F}$, both in space and time. If we strive for high accuracy in $\mathcal{G}$, the coarse grid correction becomes rather time-consuming, and we fail to achieve any speedup.

In the MGRiT community (MGRiT is a multilevel generalization of Parareal, see Section 4.4), considerable research effort has been directed toward making MGRiT work for advection equations; see (Howse, Sterck, Falgout, MacLachlan and Schroder 2019, De Sterck, Falgout, Friedhoff, Krzysik and MacLachlan 2021, De Sterck, Falgout, Krzysik and Schroder 2023*b*, De Sterck, Falgout and Krzysik 2023*a*) and references therein. The idea is to design an optimized coarse solver through the so-called semi-Lagrangian discretization. This technique performs well for linear advection equations, while research on the nonlinear case is still ongoing, as the semi-Lagrangian discretization is a characteristics-based method that is not easily realized for nonlinear problems. Another idea, proposed in (Gander and Wu 2020), also aims to make Parareal (and MGRiT) work for hyperbolic problems. In this approach, it is relatively easy to handle nonlinear problems, as we will see in Section 4.5.

4.3. PFASST

In this and the next two subsections, we present three variants of the Parareal algorithm. We begin by introducing the Parallel Full Approximation Scheme in Space-Time (PFASST), which was proposed in (Emmett and Minion 2012). The concept of this method emerged two years earlier (Minion 2010), where the author replaced the fine solver with one iteration of SDC (Dutt et al. 2000), in order to reduce the computational cost of one Parareal iteration. PFASST has been successfully applied to several problems (Emmett and Minion 2012, Speck, Ruprecht, Krause, Emmett, Minion, Winkel and Gibbon 2012, Speck, Ruprecht, Emmett, Bolten and Krause 2014), but a clear description and theoretical analysis of this method are rather challenging. Recently, (Bolten, Moser and Speck 2017) described PFASST as a time multigrid method based on an algebraic representation of SDC introduced in (Minion, Speck, Bolten, Emmett and Ruprecht 2015*b*), and provided a convergence analysis in (Bolten, Moser and Speck 2018). In the formalism of Block Iterations, PFASST was precisely described and studied for a model problem in (Gander et al. 2023*b*).

4.4. MGRiT

Multi-grid reduction in time (MGRiT) is another variant of Parareal, introduced by (Falgout et al. 2014). MGRiT can be interpreted in different ways, such as an algebraic multigrid method with the so-called FCF-relaxation, a

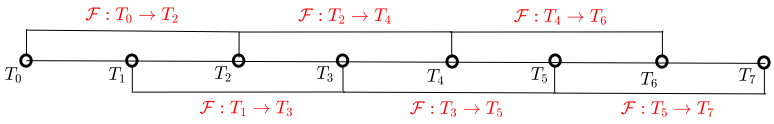

Figure 4.28. Geometric representation of MGRiT with FCF-relaxation as an overlapping variant of Parareal. The dark circles represent the coarse time points where the coarse solver $\mathcal{G}$ runs.

block iteration (Gander et al. 2023*b*) and (Gander and Lunet 2024, Chapter 4.6), or as an overlapping Parareal variant (Gander, Kwok and Zhang 2018*d*, Theorem 4 and Corollary 1). Here, we present MGRiT as an overlapping variant of Parareal applied to non-linear systems of ODEs (2.2), i.e., $\boldsymbol{u}' = f(\boldsymbol{u}, t)$ with initial value $\boldsymbol{u}(0) = \boldsymbol{u}_0$. MGRiT with two levels and FCF-relaxation then corresponds to the iteration

$$\begin{aligned} \boldsymbol{u}_0^{k+1} &= \boldsymbol{u}_0,\ \boldsymbol{u}_1^{k+1} = \mathcal{F}(T_0, T_1, \boldsymbol{u}_0), \\ \boldsymbol{u}_{n+1}^{k+1} &= \mathcal{F}(T_n, T_{n+1}, \mathcal{F}(T_{n-1}, T_n, \boldsymbol{u}_{n-1}^k)) + \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^{k+1}) \\ &\quad -\mathcal{G}(T_n, T_{n+1}, \mathcal{F}(T_{n-1}, T_n, \boldsymbol{u}_{n-1}^k)), \end{aligned} \tag{4.10}$$

where $n = 1, 2, \ldots, N_t - 1$, and $\mathcal{G}$ and $\mathcal{F}$ are the coarse and fine time propagators used in Parareal (cf. (4.1)). We see from (4.10) that MGRiT with FCF-relaxation costs two fine solves in each iteration, compared to only one fine solve in Parareal. A geometric representation of MGRiT with FCF-relaxation is shown in Figure 4.28, illustrating that two level MGRiT with FCF relaxation is a Parareal algorithm with overlap size ΔT, and it thus converges also in a finite number of iterations, i.e., the global error decays to zero after at most $k = \lceil \frac{N_t}{2} \rceil$ iterations (Gander et al. 2018*d*, Theorem 5). A superlinear convergence result for two-level MGRiT applied to non-linear problems with more general F(CF)$^\nu$-relaxation, $\nu = 1, 2, \ldots$ can be found in (Gander et al. 2018*d*, Theorem 6), and it is shown that this corresponds to Parareal with ν coarse time interval ΔT overlap; see (Gander et al. 2018*d*, Corollary 1).

In the linear case, a linear convergence estimate can be found in (Dobrev, Kolev, Petersson and Schroder 2017*b*), which we show now. We consider the linear system of ODEs (2.1), i.e., $\boldsymbol{u}' = A\boldsymbol{u} + \boldsymbol{g}$ with initial value $\boldsymbol{u}(0) = \boldsymbol{u}_0$, where we assume that A is diagonalizable with spectrum $\sigma(A) \subset \mathbb{C}^-$.

Theorem 4.5. (Dobrev et al. 2017*b*) *With the same notation and assumptions as in Theorem 4.2, MGRiT with FCF-relaxation satisfies the conver-*

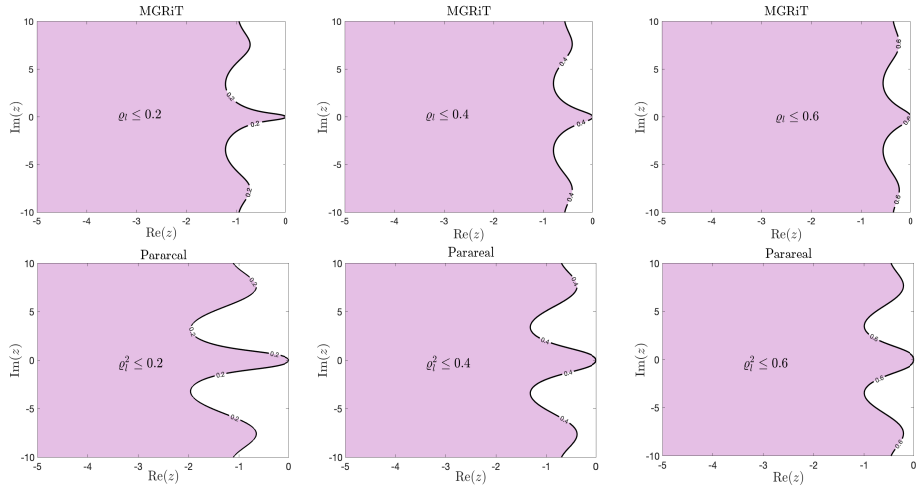

Figure 4.29. The regions where $\varrho_l \le \hat{\varrho}$ in the left half of the complex plane for MGRiT with FCF-relaxation (top row) and Parareal (bottom row).

gence estimate

$$\max_{1\le n\le N_t} \|\boldsymbol{e}_n^k\|_\infty \le \max_{z\in\sigma(\Delta T A)} \varrho_l^k(J,z) \max_{1\le n\le N_t} \|\boldsymbol{e}_n^0\|_\infty,$$
$$\varrho_l(J,z) := \frac{|\mathrm{R}_f^J(z/J)||\mathrm{R}_g(z) - \mathrm{R}_f^J(z/J)|}{1-|\mathrm{R}_g(z)|}, \tag{4.11}$$

where R_g *and* R_f *are the stability functions of the coarse solver* $\mathcal{G}$ *and the fine solver* $\mathcal{F}$, $\boldsymbol{e}_n^k := V_A(\boldsymbol{u}_n^k - \boldsymbol{u}_n)$, *and the coarse solver* $\mathcal{G}$ *is stable in the sense that* $|\mathrm{R}_g(z)| < 1$ *for* $z \in \sigma(\Delta T A)$.

The quantity ϱ_l in (4.11) is the *linear* convergence factor of MGRiT for long-time computations when applied to linear problems. Denoting by $\varrho_{l,\text{parareal}}$ the convergence factor of Parareal (cf. (4.5b)) and $\varrho_{l,\text{mgrit}}$ the convergence factor of MGRiT, it holds that

$$\varrho_{l,\text{mgrit}} = |\mathrm{R}_f^J(z/J)| \times \varrho_{l,\text{parareal}}.$$

In practice, the fine solver $\mathcal{F}$ is stable, i.e., $|\mathrm{R}_f(z)| \le 1\ \forall z \in \sigma(\Delta T A)$, and thus each additional CF relaxation adds a further contraction to MGRiT compared to Parareal, but each such relaxation also costs again an expensive fine parallel solve. To illustrate this, we show in Figure 4.29 the regions where $\varrho_{l,\text{mgrit}} \le \hat{\varrho}$ and $\varrho_{l,\text{parareal}}^2 \le \hat{\varrho}$ with $\hat{\varrho} = 0.2, 0.4, 0.6$ in the left half of the complex plane (i.e., $z \in \mathbb{C}^-$) for Parareal and MGRiT[3]. Here, we chose Backward Euler for $\mathcal{G}$ and the exact time integrator $\mathcal{F} = \exp(\Delta T A)$ for $\mathcal{F}$.

[3] For a fair comparison, we compare two Parareal iterations with one MGRiT iteration with FCF relaxation, since both then use two fine solves.

The stability functions of these two solvers are $\mathrm{R}_g(z) = \frac{1}{1-z}$ and $\mathrm{R}_f(z) = e^z$. We see that for a given upper bound $\hat{\varrho}$, the regions where $\varrho_{l,\mathrm{mgrit}} \leq \hat{\varrho}$ are comparable to those where $\varrho^s_{l,\mathrm{parareal}} \leq \hat{\varrho}$. For other fine time solvers, such as the two SDIRK methods in (4.9), the results look similar, and the above conclusion holds as well.

A more quantitative comparison for the case $z \in \mathbb{R}^-$ is

Theorem 4.6. (Wu and Zhou 2019) *Suppose we use an L-stable time integrator for $\mathcal{F}$ and the ratio between ΔT and Δt, i.e., $J = \frac{\Delta T}{\Delta t}$, satisfies $J = \mathcal{O}(1)$. Then, if we use the backward Euler method for $\mathcal{G}$, it holds that*

$$\max_{z\geq 0}\varrho_l \approx \begin{cases} 0.2984, & \textit{Parareal,} \\ 0.1115, & \textit{MGRiT with FCF-relaxation.} \end{cases}$$

If we use the LIIIC-2 method (i.e., the 2nd-order Lobatto IIIC Runge-Kutta method) for $\mathcal{G}$,

$$\max_{z\geq 0}\varrho_l \approx \begin{cases} 0.0817, & \textit{Parareal,} \\ 0.0197, & \textit{MGRiT with FCF-relaxation.} \end{cases}$$

Therefore, when using Backward Euler for $\mathcal{G}$, the convergence factor of one MGRiT iteration with FCF-relaxation is a bit worse than the convergence factor of 2 Parareal iterations ($0.2984^2 = 0.0890 < 0.1115$ and $0.0817^2 = 0.0067 < 0.0197$), and the cost in fine solves of one MGRiT iteration with FCF-relaxation is the same as the cost of 2 Parareal iterations.

We now compare the convergence rates of Parareal and MGRiT by applying them to the heat equation (2.3) and the advection-diffusion equation (2.5). We impose homogeneous Dirichlet boundary conditions for the heat equation and periodic boundary conditions for ADE. For both PDEs the initial condition is $u(x,0) = \sin^2(8\pi(1-x)^2)$ for $x \in (0,1)$. The problem and discretization parameters are $T = 5$, $J = 20$, $\Delta T = \frac{1}{8}$, and $\Delta x = \frac{1}{160}$. For $\mathcal{G}$, we use Backward Euler, and for $\mathcal{F}$ SDIRK22 from (4.9). We show in Figure 4.30 for both the heat equation and the advection-diffusion equation with two different values of the diffusion parameter ν the quantity $\varrho_l(J,z)$ for $z \in \sigma(\Delta T A)$. The maximum, denoted by $\varrho_{l,\max} := \max_{z\in\sigma(\Delta T A)} \varrho_l(J,z)$, represents the convergence factor for the two methods. It is evident that for the heat equation and the ADE with $\nu = 0.1$ the convergence factor of MGRiT with FCF-relaxation approximately equals to the square of that of Parareal, indicating that MGRiT with FCF-relaxation converges twice as fast as Parareal, but also at twice the cost, since it uses two $\mathcal{F}$ solves, and Parareal only one, which is consistent with Theorem 4.6. For ADE with $\nu = 0.01$, the convergence factor of both MGRiT and Parareal are close to 1 indicating very slow converge for both. This is due to the fact that the coarse propagator is not good enough any more for small ν when advection

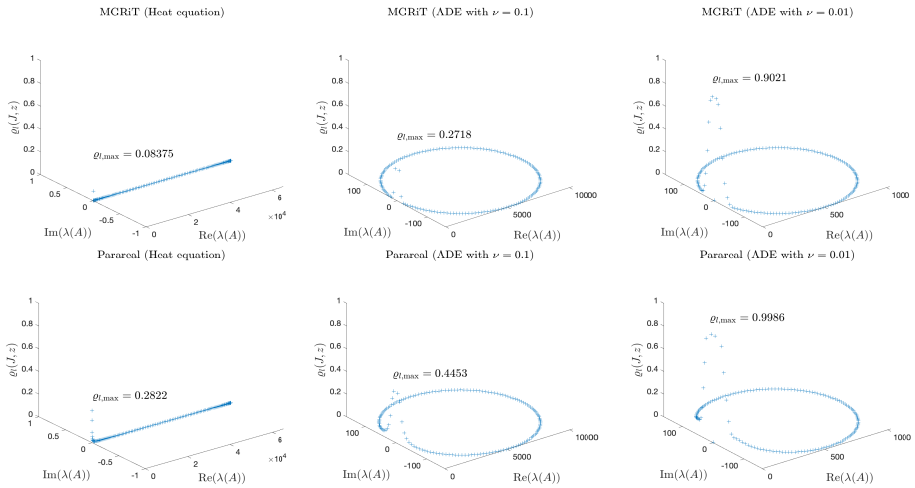

Figure 4.30. The distribution of $\varrho_l(J, z)$ for $z \in \sigma(\Delta T A)$ for the heat equation and the advection-diffusion equation (ADE) with two values of the diffusion parameter ν. Top row: MGRiT; Bottom row: Parareal. In each panel, $\varrho_{l,\max} = \max_{z\in\sigma(\Delta T A)} \varrho_l(z)$.

dominates. Note also that this has a greater impact for Parareal, which uses the coarse propagator after each fine solve, see (4.1), whereas MGRiT with FCF-relaxation performs two consecutive fine solves without coarse solve in between, see (4.10). In Figure 4.31, we present the measured errors for the two methods, where we plot for Parareal each double iteration as one, in order to to use two fine solves, like in MGRiT with FCF-relaxation. We see that for the heat equation and ADE with $\nu = 0.1$ convergence of Parareal and MGRiT is very similar, as expected, but with advection convergence is worse. This is confirmed for ADE with $\nu = 0.01$, where both MGRiT with FCF-relaxation and Parareal now converge very slowly, and we observe that Parareal deterioration is more pronounced due to the use of the ineffective coarse solve after each fine solve, as indicated by the analysis. Finally for $\nu = 0.002$ both Parareal and MGRiT diverge, the corresponding values are $\varrho_{l,\max} = 1.4211$ for Parareal and $\varrho_{l,\max} = 1.2812$ for MGRiT. These results clearly show convergence problems of both methods when approaching the hyperbolic regime.

Like Parareal, MGRiT can also be applied to nonlinear problems, where $\mathcal{G}$ and $\mathcal{F}$ require the use of some nonlinear solver. A convergence analysis of MGRiT for nonlinear cases can be found in (Gander et al. 2018*d*), under the assumption of certain Lipschitz conditions for $\mathcal{G}$, $\mathcal{F}$, and their difference. The main conclusion is as follows: one MGRiT iteration with FCF-relaxation (thus using two fine solves) contracts similarly to two Parareal iterations (also using two fine solves) as long as the coarse solver $\mathcal{G}$ is reasonably

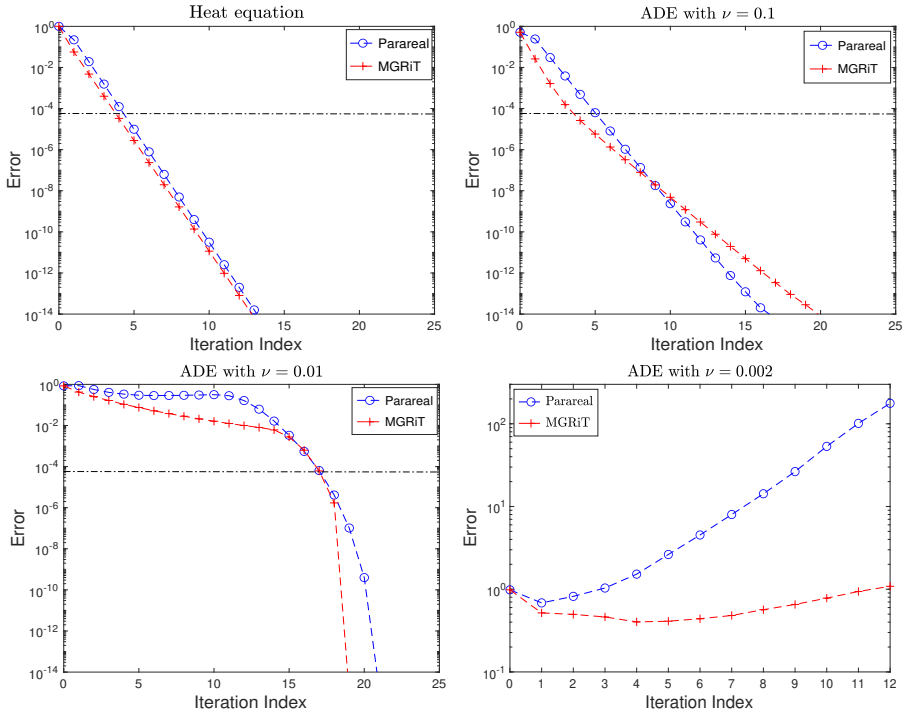

Figure 4.31. Measured errors for Parareal and MGRiT for the heat equation and the advection-diffusion equation (ADE) with three values of the diffusion parameter ν. The dash-dotted line in each panel indicates the order of the truncation error $\max\{\Delta t^2, \Delta x^2\}$ of the discretization, beyond which one would not iterate in practice.

accurate. To illustrate this, we apply MGRiT with FCF-relaxation and Parareal to Burgers' equation (2.6) with homogeneous Dirichlet boundary conditions and initial condition $u(x,0) = \sin^2(8\pi(1-x)^2)$ for $x \in (0,1)$, $T = 5$ and discretization parameters $\Delta T = \frac{1}{16}$, $\Delta x = \frac{1}{160}$, and $J = 10$. We use centered finite differences for the space discretization, and for the time discretization we use Backward Euler for $\mathcal{G}$ and SDIRK22 (4.9) for $\mathcal{F}$. In Figure 4.32, we show the errors for three values of the diffusion parameter ν, plotting again a double iteration of Parareal for one iteration of MGRiT with FCF-relaxation to measure the same number of fine solves. We see that for each ν, MGRiT converges like two steps of Parareal again, like for the linear problems in Figure 4.31.

4.5. Diagonalization-based Parareal

The third variant of Parareal we want to explain uses ParaDiag in the coarse grid correction (CGC). There are two approaches: the first one uses a head-

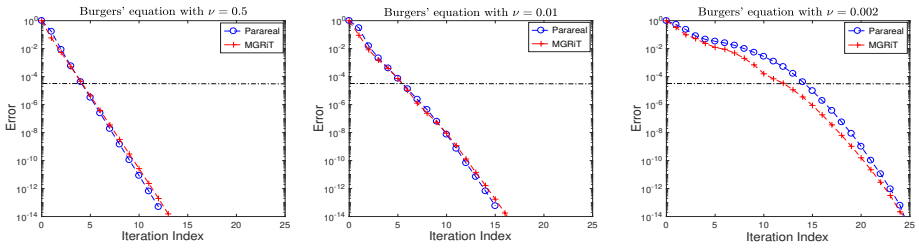

Figure 4.32. Error of Parareal and MGRiT for Burgers' equation (2.6) with three values of the diffusion parameter ν.

tail coupled condition to permit the CGC to be solved in one shot with ParaDiag; see (Wu 2018) and (Wu and Zhou 2019). The second one involves designing a special coarse solver that closely approximates the fine solver, but can be applied at low cost for each large time interval $[T_n, T_{n+1}]$ using ParaDiag (see (Gander and Wu 2020)). The two Parareal variants have distinct mechanisms, convergence properties, and scopes of application.

4.5.1. Diagonalization-based CGC

For the initial-value problem $\boldsymbol{u}' = f(\boldsymbol{u})$ with $\boldsymbol{u}(0) = \boldsymbol{u}_0$, the standard CGC for Parareal follows a sequential procedure. Specifically, we solve for $\{\boldsymbol{u}_n^{k+1}\}$ step-by-step as

$$\boldsymbol{u}_{n+1}^{k+1} = \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^{k+1}) + \boldsymbol{b}_{n+1}^k, \quad n = 0, 1, \ldots, N_t - 1, \tag{4.12}$$

starting from the initial condition $\boldsymbol{u}_0^{k+1} = \boldsymbol{u}_0$, where $\boldsymbol{b}_{n+1}^k = \mathcal{F}(T_n, T_{n+1}, \boldsymbol{u}_n^k) - \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^k)$ is determined from the previous iteration. The idea in (Wu 2018) was to impose a head-tail type coupling condition $\boldsymbol{u}_0^{k+1} = \alpha \boldsymbol{u}_{N_t}^{k+1} + \boldsymbol{u}_0$ for the CGC, which is different from the more natural head-tail condition we have seen in ParaDiag II in (3.55) that appeared a year later in (Gander and Wu 2019), but turned out to work equally well in the present context. To use this heat-tail type coupling condition, we must redefine $\boldsymbol{b}_{n+1}^k$ as

$$\boldsymbol{b}_{n+1}^k = \mathcal{F}(T_n, T_{n+1}, \tilde{\boldsymbol{u}}_n^k) - \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^k),$$

where

$$\tilde{\boldsymbol{u}}_n^k = \begin{cases} \boldsymbol{u}_n^k, & \text{if } n \geq 1, \\ \boldsymbol{u}_0, & \text{if } n = 0. \end{cases}$$

This redefinition is necessary to ensure that the iterates converge to the solution of the underlying ODEs. In summary, the Parareal variant proposed in (Wu 2018) with this head-tail type coupling condition is

$$\begin{aligned} \boldsymbol{u}_{n+1}^{k+1} &= \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^{k+1}) + \mathcal{F}(T_n, T_{n+1}, \tilde{\boldsymbol{u}}_n^k) - \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^k), \\ \boldsymbol{u}_0^{k+1} &= \alpha \boldsymbol{u}_{N_t}^{k+1} + \boldsymbol{u}_0, \end{aligned} \tag{4.13}$$

where $n = 0, \ldots, N_t - 1$.

We first explain how to implement this variant of Parareal for the system of linear ODEs $\boldsymbol{u}' = A\boldsymbol{u}$ with initial condition $\boldsymbol{u}(0) = \boldsymbol{u}_0$ and $t \in (0, T)$, the nonlinear case will be addressed at the end of this section. We use Backward Euler for the coarse solver $\mathcal{G}$ and an arbitrary one-step time integrator for the fine solver $\mathcal{F}$. By noting that $\mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^{k+1}) = (I_x - \Delta T A)^{-1}\boldsymbol{u}_n^{k+1}$, the new CGC (4.13) involves solving the N_t linear equations

$$\begin{cases} (I_x - \Delta T A)\boldsymbol{u}_1^{k+1} = \boldsymbol{u}_0^{k+1} + (I_x - \Delta T A)\boldsymbol{b}_1^k, \\ (I_x - \Delta T A)\boldsymbol{u}_2^{k+1} = \boldsymbol{u}_1^{k+1} + (I_x - \Delta T A)\boldsymbol{b}_2^k, \\ \vdots \\ (I_x - \Delta T A)\boldsymbol{u}_{N_t}^{k+1} = \boldsymbol{u}_{N_t-1}^{k+1} + (I_x - \Delta T A)\boldsymbol{b}_{N_t}^k, \\ \boldsymbol{u}_0^{k+1} = \alpha\boldsymbol{u}_{N_t}^{k+1} + \boldsymbol{u}_0, \end{cases}$$

where $\boldsymbol{b}_{n+1}^k = \mathcal{F}(T_n, T_{n+1}, \tilde{\boldsymbol{u}}_n^k) - \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^k)$ is known from the previous iteration. Note that these linear systems cannot be solved one by one due to the head-tail coupling condition $\boldsymbol{u}_0^{k+1} = \alpha\boldsymbol{u}_{N_t}^{k+1} + \boldsymbol{u}_0$. Substituting this condition into the first equation leads to the all-at-once system

$$(C_\alpha \otimes I_x - I_t \otimes \Delta T A)\boldsymbol{U}^{k+1} = \boldsymbol{g}^k, \tag{4.14}$$

where $\boldsymbol{U}^{k+1} = ((\boldsymbol{u}_1^{k+1})^\top, (\boldsymbol{u}_2^{k+1})^\top, \ldots, (\boldsymbol{u}_{N_t}^{k+1})^\top)^\top$ and

$$C_\alpha = \begin{bmatrix} 1 & & & -\alpha \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{N_t \times N_t}, \quad \boldsymbol{g}^k = \begin{bmatrix} \boldsymbol{u}_0 + (I_x - \Delta T A)\boldsymbol{b}_1^k \\ (I_x - \Delta T A)\boldsymbol{b}_2^k \\ \vdots \\ (I_x - \Delta T A)\boldsymbol{b}_{N_t}^k \end{bmatrix}.$$

Similar to the ParaDiag II methods introduced in Section 3.5.2, we can solve for $\boldsymbol{U}^{k+1}$ using three steps,

$$\begin{cases} \boldsymbol{U}^{a,k+1} = (\mathrm{F} \otimes I_x)\boldsymbol{g}^k, & \text{(step-a)} \\ (\lambda_n I_x - \Delta T A)\boldsymbol{u}_n^{b,k+1} = \boldsymbol{u}_n^{a,k+1}, \ n = 1, 2, \ldots, N_t, & \text{(step-b)} \\ \boldsymbol{U}^{k+1} = (\mathrm{F}^* \otimes I_x)\boldsymbol{U}^{b,k+1}, & \text{(step-c)} \end{cases} \tag{4.15}$$

where $\{\lambda_n\}$ are the eigenvalues of C_α (cf. (3.51)), F is the discrete Fourier matrix (cf. (3.50)), and $\boldsymbol{U}^{a,k+1} := ((\boldsymbol{u}_1^{a,k+1})^\top, \ldots, (\boldsymbol{u}_{N_t}^{a,k+1})^\top)^\top$ along with $\boldsymbol{U}^{b,k+1} := ((\boldsymbol{u}_1^{b,k+1})^\top, \ldots, (\boldsymbol{u}_{N_t}^{b,k+1})^\top)^\top$. Through the diagonalization procedure (4.15), the new CGC (4.13) can be solved in parallel across the coarse time grid.

From (4.13) we see that the head-tail coupled CGC simplifies to the standard CGC (4.12) when $\alpha \to 0$, and hence, one can expect that for sufficiently small α, this Parareal variant converges as fast as the original Parareal algorithm. However, due to the roundoff error stemming from the diago-

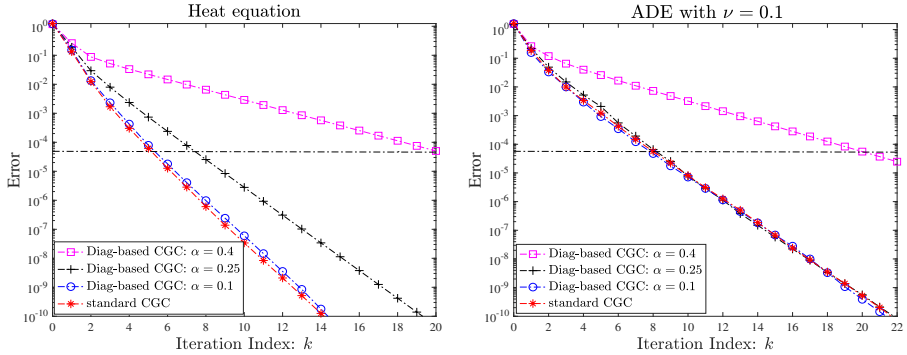

Figure 4.33. Measured error of Parareal using the diagonalization-based CGC (cf. (4.13)) and the standard CGC (i.e., $\alpha = 0$ in (4.12)).

nalization of C_α (as discussed in Section 3.5.2), an arbitrarily small α is impractical, particularly when the working precision is low, e.g., single or half precision. Fortunately, it is not necessary to use an extremely small α for the diagonalization-based CGC to achieve the same convergence rate as the standard CGC.

Theorem 4.7. (Wu 2018) *Let ρ denote the convergence factor of standard Parareal* (4.12)*, and ρ_{new} the convergence factor of the new Parareal variant* (4.13)*, where the coarse solver $\mathcal{G}$ is a stable time integrator. Then it holds that*

$$\rho_{\text{new}} = \rho, \ \text{if } \alpha \le \frac{\rho}{1+\rho}.$$

This result was proved for linear problems $\boldsymbol{u}' = A\boldsymbol{u} + g$ where A has negative real eigenvalues. For other scenarios, such as when A has complex eigenvalues, numerical results suggest its validity as well. Since the roundoff error resulting from the diagonalization procedure grows as α decreases, it is clear that the optimal choice for α is $\alpha = \frac{\rho}{1+\rho}$. In practice, $\rho = \mathcal{O}(10^{-1})$ and hence $\alpha = \frac{\rho}{1+\rho} = \mathcal{O}(10^{-1})$ as well. The roundoff error incurred with a parameter $\alpha = \mathcal{O}(10^{-1})$ is negligible.

To illustrate the convergence of the new Parareal variant (4.13), we consider the heat equation (2.3) and the advection-diffusion equation (ADE) (2.5) with periodic boundary conditions and an initial condition $u(x,0) = \sin(2\pi x)$ for $x \in (0,1)$. We use Backward Euler as the coarse solver $\mathcal{G}$ and SDIRK22 from (4.9) for the fine solver $\mathcal{F}$. The data used here is $T = 4$, $J = 10$, $\Delta T = 0.1$, and $\Delta x = \frac{1}{128}$. In Figure 4.33, we present the measured error of Parareal using both the diagonalization-based and standard CGC. For the diagonalization-based CGC, we use three values of the parameter α to illustrate how the convergence rate depends on this parameter. For the heat equation, the convergence factor of Parareal with standard CGC

is $\rho \approx 0.22$, and thus, according to Theorem 4.7, the threshold for α such that the diagonalization-based CGC achieves the same convergence rate is $\frac{\rho}{1+\rho} \approx 0.18$. If α exceeds this threshold, the diagonalization-based CGC results in a slower convergence rate. Thus the theoretical analysis accurately predicts the numerical results shown in the left panel. For ADE with $\nu = 0.1$, $\rho \approx 0.39$ and the threshold for α is $\frac{\rho}{1+\rho} \approx 0.28$. Hence, as seen in Figure 4.33 on the right, $\alpha = 0.25$ suffices to let the diagonalization-based Parareal converge as fast as the standard Parareal.

We next address how to adapt the diagonalization-based CGC to nonlinear problems of the form $\boldsymbol{u}' = f(\boldsymbol{u})$ with an initial value $\boldsymbol{u}(0) = \boldsymbol{u}_0$. We continue to use backward Euler for the coarse solver $\mathcal{G}$. Then, with $\boldsymbol{b}_{n+1}^k := \mathcal{F}(T_n, T_{n+1}, \tilde{\boldsymbol{u}}_n^k) - \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^k)$ the initial part of the CGC algorithm (4.13), i.e., $\boldsymbol{u}_{n+1}^{k+1} = \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^{k+1}) + \boldsymbol{b}_{n+1}^k$, can be reformulated as

$$\frac{\boldsymbol{u}_{n+1}^{k+1} - \boldsymbol{b}_{n+1}^k - \boldsymbol{u}_n^{k+1}}{\Delta T} = f(\boldsymbol{u}_{n+1}^{k+1} - \boldsymbol{b}_{n+1}^k),\ n = 0, 1, \ldots, N_t - 1.$$

This, together with the head-tail coupling condition $\boldsymbol{u}_0^{k+1} = \alpha \boldsymbol{u}_{N_t}^{k+1} + \boldsymbol{u}_0$, leads to the nonlinear all-at-once system

$$(C_\alpha \otimes I_x)\boldsymbol{U}^{k+1} - \Delta T F(\boldsymbol{U}^{k+1}) = \boldsymbol{g}^k, \tag{4.16}$$

where the definitions for $\boldsymbol{U}^{k+1}$ and C_α remain the same as in (4.14), and

$$F(\boldsymbol{U}^{k+1}) := \begin{bmatrix} f(\boldsymbol{u}_1^{k+1} - \boldsymbol{b}_1^k) \\ f(\boldsymbol{u}_2^{k+1} - \boldsymbol{b}_2^k) \\ \vdots \\ f(\boldsymbol{u}_{N_t}^{k+1} - \boldsymbol{b}_{N_t}^k) \end{bmatrix}, \quad \boldsymbol{g}^k := \begin{bmatrix} \boldsymbol{b}_1^k + \boldsymbol{u}_0 \\ \boldsymbol{b}_2^k \\ \vdots \\ \boldsymbol{b}_{N_t}^k \end{bmatrix}.$$

We solve the nonlinear system (4.16) by a quasi Newton method as previously for the nonlinear ParaDiag method (cf. Section 3.5.1),

$$\begin{aligned} &\mathcal{P}_\alpha^{k+1,l} \Delta \boldsymbol{U}^{k+1,l} = \boldsymbol{g}^k - (C_\alpha \otimes I_x)\boldsymbol{U}^{k+1,l} + \Delta T F(\boldsymbol{U}^{k+1,l}), \\ &\boldsymbol{U}^{k+1,l+1} = \boldsymbol{U}^{k+1,l} + \Delta \boldsymbol{U}^{k+1,l}, \end{aligned} \tag{4.17a}$$

where $l = 0, 1, \ldots, l_{\max}$ denotes the Newton iteration index. The matrix $\mathcal{P}_\alpha^{k+1,l}$ is a block α-circulant matrix given by

$$\mathcal{P}_\alpha^{k+1,l} := C_\alpha \otimes I_x - I_t \otimes \Delta T A^{k+1,l}, \tag{4.17b}$$

where $A^{k+1,l}$ is the average of the Jacobi matrices,

$$A^{k+1,l} := \frac{1}{J} \sum\nolimits_{j=1}^{J} \nabla f(\boldsymbol{u}_n^{k+1,l} - \boldsymbol{b}_n^k).$$

The Kronecker tensor product $I_t \otimes A^{k+1,l}$ serves as an approximation of the Jacobi matrix $\nabla F(\boldsymbol{U}^{k+1,l})$. We note that the nearest Kronecker product approximation (NKA), introduced in Section 3.5.1, could also be used to

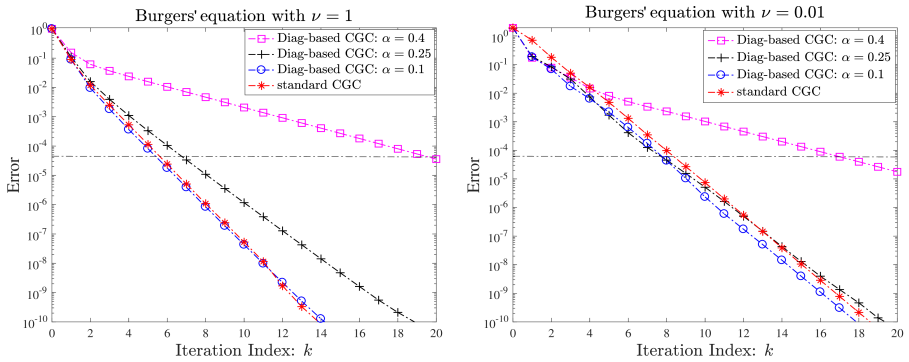

Figure 4.34. Measured error of Parareal using the two CGCs for Burgers' equation with two values of the diffusion parameter ν.

obtain a better approximation of $\nabla F(\boldsymbol{U}^{k+1,l})$, but for simplicity, we do not explore this option here further.

The matrix $\mathcal{P}_\alpha^{k+1,l}$ has the same structure as the coefficient matrix in (4.14), which allows us to solve for the increment $\Delta\boldsymbol{U}^{k+1,l}$ using the diagonalization procedure outlined in (4.15). The convergence analysis of the new Parareal variant (4.13) in the nonlinear context is detailed in (Wu 2018, Section 4), where it is shown that the convergence rate mirrors that of Parareal with standard CGC when α is chosen appropriately small. We illustrate this by applying Parareal with both CGCs to Burgers' equation (2.6), using the same problem setup and discretization parameters as in the previously discussed heat and advection-diffusion equation case. In Figure 4.34, we present the measured errors for two values of the diffusion parameter ν. We see that also for this nonlinear problem, the influence of α on the convergence rate remains as in the linear case.

Remark 4.2 (Extension to MGRiT). *The fundamental mechanism of MGRiT* (4.10) *aligns with that of Parareal, and specifically, the CGC can be similarly represented as in* (4.12)*. However, a direct extension of the diagonalization-based CGC devised for Parareal in* (4.13) *to MGRiT leads to divergence regardless of the choice of* α*. This is because the head-tail type coupling condition* $\boldsymbol{u}_1^{k+1} = \alpha\boldsymbol{u}_{N_t}^{k+1} + \boldsymbol{u}_1$ *used is less natural than the one from (Gander and Wu 2019) that appeared a year later, namely* $\boldsymbol{u}_1^{k+1} = \alpha(\boldsymbol{u}_{N_t}^{k+1} - \boldsymbol{u}_{N_t}^{k}) + \boldsymbol{u}_1$*. Using this more natural head-tail condition which is consistent at convergence for MGRiT was proposed in (Wu and Zhou 2019), leading to the convergent MGRiT variant*

$$\begin{aligned} &\boldsymbol{u}_0^{k+1} = \boldsymbol{u}_0,\ \ \boldsymbol{u}_1^{k+1} = \alpha(\boldsymbol{u}_{N_t}^{k+1} - \boldsymbol{u}_{N_t}^{k}) + \boldsymbol{u}_1, \\ &\boldsymbol{u}_{n+1}^{k+1} = \mathcal{G}(T_n, T_{n+1}, \boldsymbol{u}_n^{k+1}) + \tilde{\boldsymbol{b}}_{n+1}^{k},\ \ n = 1, 2, \ldots, N_t - 1, \end{aligned} \tag{4.18}$$

where $\tilde{\boldsymbol{b}}_{n+1}^k = \mathcal{F}(T_n, T_{n+1}, \tilde{\boldsymbol{s}}_n^k) - \mathcal{G}(T_n, T_{n+1}, \tilde{\boldsymbol{s}}_n^k)$, $\tilde{s}_n^k := \mathcal{F}(T_{n-1}, T_n, \tilde{\boldsymbol{u}}_{n-1}^k)$ *and*

$$\tilde{\boldsymbol{u}}_n^k = \begin{cases} \boldsymbol{u}_n, & n = 0, 1, \\ \boldsymbol{u}_n^k, & n \geq 2. \end{cases}$$

This variant converges at the same rate as the original MGRiT method (4.10)*, provided* α *is suitably small, Theorem 4.7 applies to MGRiT in an analogous manner. For Parareal, one can also use the more natural head-tail condition* $\boldsymbol{u}_0^{k+1} = \alpha(\boldsymbol{u}_{N_t}^{k+1} - \boldsymbol{u}_{N_t}^k) + \boldsymbol{u}_0$ *which is consistent at convergence, instead of* (4.13)*, and one obtains the same convergence rate as the one in Theorem 4.7 shown for the less natural one.*

4.5.2. Diagonalization-based Coarse Solver

A fundamentally distinct idea from the ParaDiag CGC method in Section 4.5.1 that combines ParaDiag with Parareal was introduced in (Gander and Wu 2020). The key innovation lies in using the same time integrator and time step size for both the coarse and fine solvers, but implementing the coarse solver through a diagonalization procedure. This approach can work also for hyperbolic problems, since it transports all frequency components also in the coarse propagator over very long time.

We illustrate this concept for the nonlinear system of ODEs $\boldsymbol{u}' = f(\boldsymbol{u})$ with initial value $\boldsymbol{u}(0) = \boldsymbol{u}_0$. For the time discretization, we use the linear-θ method (a generalization to the s-stage Runge-Kutta method can be found in the Appendix of (Gander and Wu 2020)). On each large time interval $[T_n, T_{n+1}]$, the fine solver $\mathcal{F}(T_n, T_{n+1}, \boldsymbol{u}_n)$ computes the solution at $t = T_{n+1}$ by performing J steps of the linear-θ method sequentially, i.e., $\mathcal{F}(T_n, T_{n+1}, \boldsymbol{u}_n) = \boldsymbol{v}_J$ with $\boldsymbol{v}_J$ being the final solution of

$$\boldsymbol{v}_{j+1} - \boldsymbol{v}_j = \Delta t[\theta f(\boldsymbol{v}_{j+1}) + (1-\theta) f(\boldsymbol{v}_j)],\ j = 0, 1, \ldots, J-1, \tag{4.19}$$

with initial condition $\boldsymbol{v}_0 = \boldsymbol{u}_n$, where $\Delta t = \frac{\Delta T}{J} = \frac{T_{n+1} - T_n}{J}$ and $\theta = 1$ or $\theta = \frac{1}{2}$. For the coarse solver, denoted by $\mathcal{F}_\alpha^*(T_n, T_{n+1}, \boldsymbol{u}_n)$, we solve the nonlinear system with a head-tail coupled condition,

$$\begin{aligned} &\boldsymbol{v}_{j+1} - \boldsymbol{v}_j = \Delta t[\theta f(\boldsymbol{v}_{j+1}) + (1-\theta) f(\boldsymbol{v}_j)],\ j = 0, 1, \ldots, J-1, \\ &\boldsymbol{v}_0 = \alpha \boldsymbol{v}_J + (1-\alpha)\boldsymbol{u}_n. \end{aligned} \tag{4.20}$$

This system can be recast as the nonlinear all-at-once system

$$\underbrace{(C_\alpha \otimes I_x)\boldsymbol{V} - \Delta t F(\boldsymbol{V})}_{:=\mathcal{K}(\boldsymbol{V})} = \boldsymbol{b}(\boldsymbol{u}_n), \tag{4.21}$$

where $\boldsymbol{V} := (\boldsymbol{v}_1^\top, \boldsymbol{v}_2^\top, \ldots, \boldsymbol{v}_J^\top)^\top$, $\boldsymbol{b}(\boldsymbol{u}_n) := ((1-\alpha)\boldsymbol{u}_n^\top, 0, \ldots, 0)^\top$, and

$$\begin{aligned} C_\alpha &:= \begin{bmatrix} 1 & & & -\alpha \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}, \\ F(\boldsymbol{V}) &:= \begin{bmatrix} \theta f(\boldsymbol{v}_1) + (1-\theta) f(\alpha \boldsymbol{v}_J + (1-\alpha)\boldsymbol{u}_n) \\ \theta f(\boldsymbol{v}_2) + (1-\theta) f(\boldsymbol{v}_1) \\ \vdots \\ \theta f(\boldsymbol{v}_J) + (1-\theta) f(\boldsymbol{v}_{J-1}) \end{bmatrix}. \end{aligned} \tag{4.22}$$

We solve (4.21) with the quasi Newton method

$$\mathcal{P}_\alpha(\boldsymbol{V}^l)\Delta \boldsymbol{V}^l = \boldsymbol{b}(\boldsymbol{u}_n) - \mathcal{K}(\boldsymbol{V}^l), \quad \boldsymbol{V}^{l+1} = \boldsymbol{V}^l + \Delta \boldsymbol{V}^l, \tag{4.23a}$$

where $l = 0, 1, \ldots, l_{\max}$, and the matrix $\mathcal{P}_\alpha(\boldsymbol{V}^l)$ is a block α-circulant matrix serving as an approximation to the Jacobi matrix $\nabla \mathcal{K}(\boldsymbol{V}^l) = C_\alpha \otimes I_x - \Delta t(\tilde{C}_{\theta,\alpha} \otimes I_x)\nabla F(\boldsymbol{V}^l)$. It is defined by

$$\mathcal{P}_\alpha(\boldsymbol{V}^l) = C_\alpha \otimes I_x - \Delta t \tilde{C}_{\alpha,\theta} \otimes \overline{\nabla \mathbf{f}}(\boldsymbol{V}^l), \tag{4.23b}$$

where $\tilde{C}_{\alpha,\theta}$ is an α-circulant matrix given by

$$\tilde{C}_{\theta,\alpha} := \begin{bmatrix} \theta & & & (1-\theta)\alpha \\ 1-\theta & \theta & & \\ & \ddots & \ddots & \\ & & 1-\theta & \theta \end{bmatrix},$$

and $\overline{\nabla f}(\boldsymbol{V}^l)$ represents the average of the J Jacobi blocks of $\nabla F(\boldsymbol{V}^l)$,

$$\overline{\nabla f}(\boldsymbol{V}^l) := \frac{1}{J}\left[\sum\nolimits_{j=1}^{J-1} \nabla f(\boldsymbol{v}_j^l) + \nabla f(\alpha \boldsymbol{v}_J^l + (1-\alpha)\boldsymbol{u}_n)\right].$$

Using this notation, we can write the parareal algorithm from (Gander and Wu 2020) as

$$\boldsymbol{u}_{n+1}^{k+1} = \mathcal{F}_\alpha^*(T_n, T_{n+1}, \boldsymbol{u}_n^{k+1}) + \mathcal{F}(T_n, T_{n+1}, \boldsymbol{u}_n^k) - \mathcal{F}_\alpha^*(T_n, T_{n+1}, \boldsymbol{u}_n^k), \tag{4.24}$$

where $n = 0, 1, \ldots, N_t - 1$ denotes the time step index.

For linear problems, i.e., $f(\boldsymbol{u}) = A\boldsymbol{u}$, the all-at-once system (4.21) becomes

$$(C_\alpha \otimes I_x - \tilde{C}_{\theta,\alpha} \otimes \Delta t A)\boldsymbol{V} = \boldsymbol{b}(\boldsymbol{u}_n), \tag{4.25}$$

with

$$\boldsymbol{b}(\boldsymbol{u}_n) = ([(I_x + \Delta t(1-\theta)A)(1-\alpha)\boldsymbol{u}_n]^\top, 0, \ldots, 0)^\top.$$

The coarse solver $\mathcal{F}_\alpha^*(T_n, T_{n+1}, \boldsymbol{u}_n^k)$ is defined by

$$\mathcal{F}_\alpha^*(T_n, T_{n+1}, \boldsymbol{u}_n^k) := (H_J \otimes I_x)\boldsymbol{V},$$

where $H_J := (0, \ldots, 0, 1) \in \mathbb{R}^{1\times J}$. With $\boldsymbol{V} = (\boldsymbol{v}_1^\top, \boldsymbol{v}_2^\top, \ldots, \boldsymbol{v}_J^\top)^\top$, it holds that $\mathcal{F}_\alpha^*(T_n, T_{n+1}, \boldsymbol{u}_n^k) = \boldsymbol{v}_J$ and the computation of $\boldsymbol{V}$ in (4.25) is equivalent to solving the all-at-once system

$$\begin{aligned} &\boldsymbol{v}_{j+1} - \boldsymbol{v}_j = \Delta t A[\theta \boldsymbol{v}_{j+1} + (1-\theta)\boldsymbol{v}_j],\ j = 0, 1, \ldots, J-1, \\ &\boldsymbol{v}_0 = \alpha \boldsymbol{v}_J + (1-\alpha)\boldsymbol{u}_n^k. \end{aligned} \tag{4.26}$$

It is clear that the coarse solver reduces to the fine solver if $\alpha = 0$, and in this limit, Parareal (4.24) converges in only one iteration, but without any speedup because we have to solve (4.26) for $\mathcal{F}_\alpha^*(T_n, T_{n+1}, \boldsymbol{u}_n^k) = \boldsymbol{v}_J$ sequentially. For $\alpha \in (0, 1)$, we solve (4.26) in one shot by diagonalization, which is parallel for the J fine time points and thus the computation time is approximately J times less than the fine solver $\mathcal{F}(T_n, T_{n+1}, \boldsymbol{u}_n^k)$. This Parareal variant has different convergence rates for parabolic and hyperbolic problems:

Theorem 4.8. (Gander and Wu 2020) *For linear initial value problems $\boldsymbol{u}' = A\boldsymbol{u} + g$ with $\boldsymbol{u}(0) = \boldsymbol{u}_0$ and $A \in \mathbb{C}^{N_x\times N_x}$, let $\{\boldsymbol{u}_n^k\}$ be the k-th iterate of the parareal variant* (4.24) *and $\{\boldsymbol{u}_n\}$ be the converged solution. Then, for any stable one-step Runge-Kutta method used for $\mathcal{F}$ and $\mathcal{F}_\alpha^*$, the global error $\boldsymbol{e}^k = \max_{n=1,2,\ldots,N_t} \|\boldsymbol{u}_n - \boldsymbol{u}_n^k\|_\infty$ satisfies the estimate*

$$\boldsymbol{e}^k \le \rho^k \boldsymbol{e}^0,$$

where the convergence factor ρ is given by

$$\rho = \begin{cases} \alpha, & \text{if } \sigma(A) \subset \mathbb{R}^-, \\ \frac{2\alpha N_t}{1+\alpha}, & \text{if } \sigma(A) \subset \mathrm{i}\mathbb{R}. \end{cases} \tag{4.27}$$

Here, $\sigma(A)$ denotes the spectrum of A. When the matrix A arises from semi-discretizing the heat equation, i.e., $A \approx \Delta$, it holds that $\sigma(A) \subset \mathbb{R}^-$. In this case, the convergence factor $\rho = \alpha$ implies that the parareal variant (4.24) converges with a rate independent of N_t. For wave propagation problems, e.g., the second-order wave equation (2.7) and the Schrödinger equation, all the eigenvalues of the discrete matrix A are imaginary, i.e., $\sigma(A) \subset \mathrm{i}\mathbb{R}$. For this kind of problems, the convergence factor increases linearly in N_t. However, this does not necessarily imply that the convergence rate deteriorates, especially when α is relatively small and N_t is not too large.

We now illustrate the convergence of the Parareal variant (4.24) for the heat equation (2.3) with homogeneous Dirichlet boundary conditions and initial condition $u(x, 0) = \sin^2(2\pi x)$ with $x \in (0, 1)$. For both $\mathcal{F}$ and $\mathcal{F}_\alpha^*$, we use the Trapezoidal Rule and the discretization parameters $\Delta T = \frac{1}{12}$, $J = 10$, $\Delta x = \frac{1}{100}$. For two values of N_t and three parameters α, we show

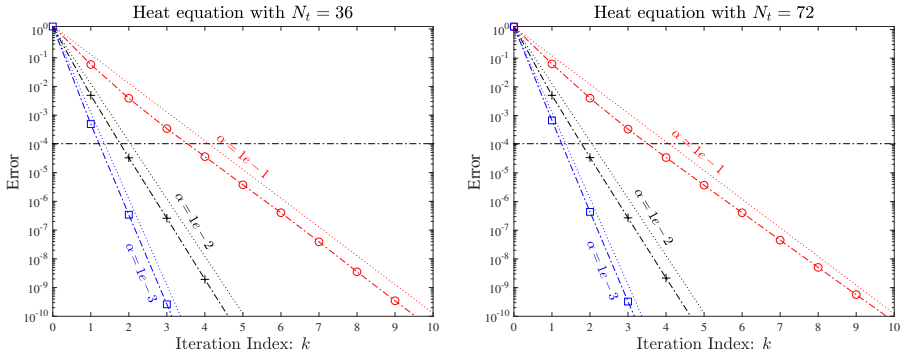

Figure 4.35. Error of the Parareal variant (4.24) for the heat equation. The dotted lines denote the error predicted by the theoretical convergence factor $\rho = \alpha$.

in Figure 4.35 the measured error, where the error predicted by the convergence factor $\rho = \alpha$ is plotted as dotted lines. We see that the theoretical convergence factor is sharp, and the convergence rate is indeed independent of N_t.

We next consider the wave equation (2.7) with periodic boundary conditions and initial conditions $u(x,0) = \sin^2(2\pi x)$ and $\partial_t u(x,0) = 0$. After space discretization, the system of ODEs is given by

$$\boldsymbol{w}' = \boldsymbol{A}\boldsymbol{w},\ \boldsymbol{w}(0) = \begin{bmatrix} \sin^2(2\pi \boldsymbol{x}_h) \\ 0 \end{bmatrix},\ \boldsymbol{A} := \begin{bmatrix} & I_x \\ A & \end{bmatrix},$$

where $\boldsymbol{w} = (\boldsymbol{u}^\top, (\boldsymbol{u}')^\top)^\top$ and $A \approx \Delta$. All the eigenvalues of $\boldsymbol{A}$ are purely imaginary, and thus according to (4.27) the convergence rate of the Parareal variant (4.24) deteriorates as N_t grows. For relatively large α, e.g., $\alpha = 0.01$, this is indeed the case, as shown in Figure 4.36 (left). However, for small α, the influence of N_t on the convergence rate becomes insignificant, as shown in Figure 4.36 (right). For example, as N_t increases from 24 to 960, we only require an additional 2 iterations to reach the stopping tolerance, denoted by the horizontal line, i.e., the order of the discretization error $\max\{\Delta t^2, \Delta x^2\}$.

In contrast to the heat equation, for wave equations the convergence factor given in (4.27) is not always sharp, depending on the product αN_t. This is illustrated in Figure 4.37, where we consider three groups of (α, N_t). For a small product, i.e., $\alpha = 1e-4$ and $N_t = 24$, the convergence factor quite accurately predicts the measured error. For the other two values of (α, N_t), the linear bound is not sharp, and we observe superlinear convergence of the method.

For nonlinear problems, $\boldsymbol{u}' = f(\boldsymbol{u})$, the convergence analysis of the Parareal variant (4.24) can be found in (Gander and Wu 2020, Section 4), under the assumption that the solution of the nonlinear all-at-once system (4.21) is

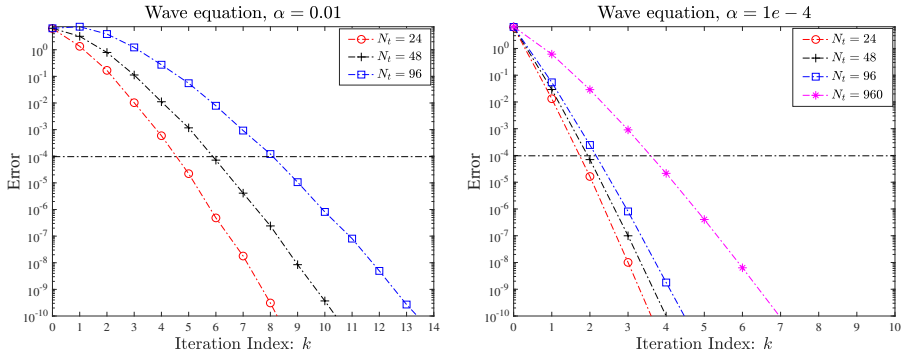

Figure 4.36. Measured error of the Parareal variant (4.24) with two values of the parameter α for the wave equation.

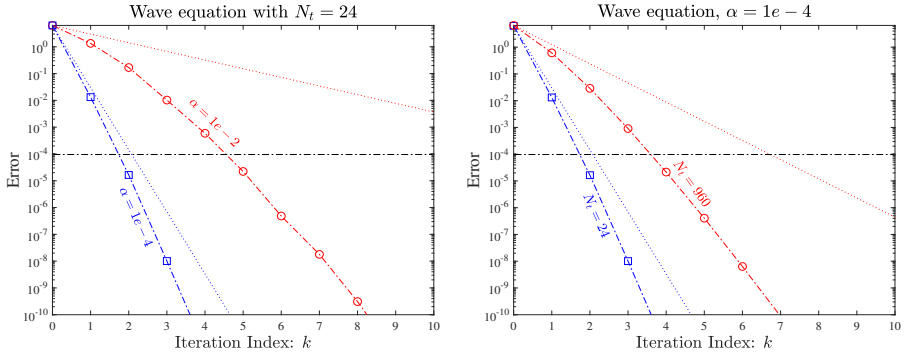

Figure 4.37. The convergence factor ρ (dotted line) for the wave equation gives quite a sharp bound on the measured error for small αN_t, while for large αN_t the bound is not sharp.

solved exactly and that the nonlinear function f satisfies some Lipschitz condition. The main conclusion is that the method converges with a rate $\rho = \mathcal{O}(\alpha)$ when α is small, which is similar to the result for the linear case. We illustrate this for Burgers' equation (2.6) with periodic boundary conditions and initial condition $u(x,0) = \sin^2(2\pi x)$ for $x \in (0,1)$. Let $\Delta T = 0.1$, $J = 10$, and $\Delta x = \frac{1}{100}$. Then, by fixing $N_t = 40$, we show in Figure 4.38 (left) the iteration number for several values of α when the global error reaches 10^{-8}. Clearly, a smaller α accelerates convergence. Concerning the influence of ν, it seems that for small α it only has a minor influence on the convergence rate, but for large α, the convergence rate deteriorates when ν decreases. In the right panel of Figure 4.38, we show the iteration numbers when $\alpha = 10^{-3}$ and N_t varies from 10 to 160, which indicates that the convergence rate is robust in terms of N_t.

The two Parareal variants (4.13) and (4.24) introduced in this section

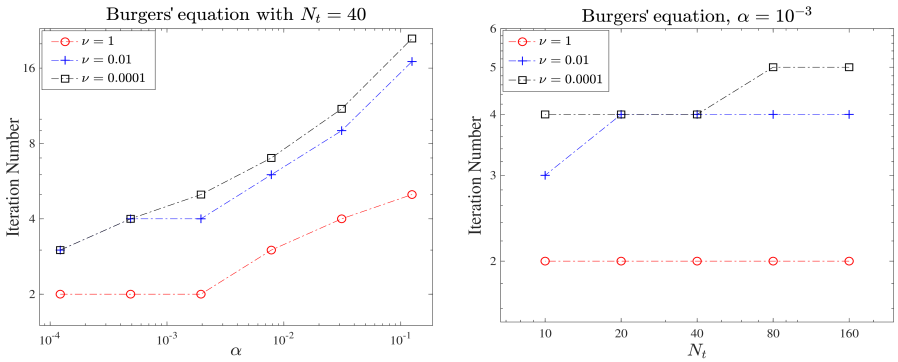

Figure 4.38. Iteration numbers of the parareal variant (4.24) for Burgers' equation when the global error reaches $1e-8$, with three values of the diffusion parameter ν.

apply ParaDiag to standard Parareal in different ways. For the former, diagonalization is used for the N_t coarse time points, changing the CGC. For the second variant, diagonalization is used for each large time interval $[T_n, T_{n+1}]$ across the J fine time points, defining a special coarse solver while keeping the CGC as in the standard Parareal. These two variants have distinct scopes of application: the first one, like the standard Parareal, works primarily for parabolic problems, while the second one is effective for both parabolic and hyperbolic problems.

4.6. Space-time multigrid (STMG)

The final parallel method we wish to introduce is the space-time multigrid (STMG) method, which is based on using the multigrid (MG) method both in space and time. After early seminal contributions (Hackbusch 1984, Horton and Vandewalle 1995), it was recognized that block Jacobi smoothers in time are a crucial component (Gander and Neumüller 2016), leading to a method as effective as when MG is applied to Poisson problems, using only standard multigrid components in STMG. For the heat equation (2.3) or the advection-diffusion equation (2.5), the STMG method can be described as follows: using a spatial discretization with mesh size Δx results in a system of ODEs $\boldsymbol{u}' = A\boldsymbol{u} + \boldsymbol{f}$, to which we apply a one-step time-integrator,

$$r_1 \boldsymbol{u}_{n+1} = r_2 \boldsymbol{u}_n + \tilde{\boldsymbol{f}}_n, \quad n = 0, 1, \ldots, N_t - 1, \tag{4.28}$$

with $\boldsymbol{u}_0$ the initial value, and r_1 and r_2 are matrix polynomials of $\Delta t A$ (cf. (3.57) for Backward Euler and the Trapezoidal Rule). The matrix $A \in \mathbb{R}^{N_x \times N_x}$ is the discrete matrix of the Laplacian ∂_{xx} or the advection-diffusion operator $-\partial_x + \nu \partial_{xx}$ with mesh size Δx. As in ParaDiag described

in Section 3.5, we collect the N_t difference equations in the all-at-once system

$$\underbrace{\begin{bmatrix} r_1 & & & \\ -r_2 & r_1 & & \\ & \ddots & \ddots & \\ & & -r_2 & r_1 \end{bmatrix}}_{:=\mathcal{K}} \underbrace{\begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{N_t} \end{bmatrix}}_{:=\boldsymbol{U}} = \boldsymbol{b}, \tag{4.29}$$

where $\boldsymbol{b}$ is a suitable right hand side vector.

STMG solves for $\boldsymbol{U}$ within a multigrid framework, using a damped block Jacobi iteration as smoother. Starting from an initial approximation $\boldsymbol{U}^{\text{ini}}$ of $\boldsymbol{U}$, the smoother $\mathcal{S}$ produces a new approximation $\boldsymbol{U}^{\text{new}}$ by computing

$$\boldsymbol{U}^{\text{new}} = \mathcal{S}_\eta(\boldsymbol{b}, \boldsymbol{U}^{\text{ini}}, s) : \begin{cases} \boldsymbol{U}^0 = \boldsymbol{U}^{\text{ini}}, \\ \text{for } j = 0, 1, \ldots, s-1 : \\ \quad (I_t \otimes r_1)\Delta \boldsymbol{U}^j = \eta(\boldsymbol{b} - \mathcal{K}\boldsymbol{U}^j), \\ \quad \boldsymbol{U}^{j+1} = \boldsymbol{U}^j + \Delta \boldsymbol{U}^j, \\ \boldsymbol{U}^{\text{new}} = \boldsymbol{U}^s, \end{cases} \tag{4.30}$$

where s is the number of smoothing iterations and η is the damping parameter. Since $I_t \otimes r_1$ is a block diagonal matrix, for each smoothing step the N_t subvectors of $\Delta \boldsymbol{U}^j$ can be solved in parallel, making this a parallel-in-time smoother. We also need restriction and prolongation operators in space and time. For illustration, we show these two operators in space with $N_x = 7$,

$$P_x := \begin{bmatrix} \frac{1}{2} & & \\ 1 & & \\ \frac{1}{2} & \frac{1}{2} & \\ & 1 & \\ & \frac{1}{2} & \frac{1}{2} \\ & & 1 \\ & & \frac{1}{2} \end{bmatrix} \in \mathbb{R}^{7\times 3}, \quad R_x = \frac{1}{2} P_x^\top \in \mathbb{R}^{3\times 7}. \tag{4.31}$$

Similar notations apply to the other two operators P_t and R_t for the time variable. We can now define the 2-level variant of STMG from iteration k to $k+1$ as

$$\begin{cases} \boldsymbol{U}^{k+\frac{1}{3}} = \mathcal{S}_\eta(\boldsymbol{b}, \boldsymbol{U}^k, s_1), \\ \boldsymbol{r} = \boldsymbol{b} - \mathcal{K}\boldsymbol{U}^{k+\frac{1}{3}}, \quad \boldsymbol{r}_{\text{c}} = [R_x \text{Mat}(\boldsymbol{r})] R_t^\top, \\ \boldsymbol{e}_{\text{c}} = \mathcal{K}_{\text{c}}^{-1} \text{Vec}(\boldsymbol{r}_{\text{c}}), \quad \boldsymbol{e} = [P_x \text{Mat}(\boldsymbol{e}_{\text{c}})] P_t^\top, \\ \boldsymbol{U}^{k+\frac{2}{3}} = \boldsymbol{U}^{k+\frac{1}{3}} + \text{Vec}(\boldsymbol{e}), \\ \boldsymbol{U}^{k+1} = \mathcal{S}_\eta(\boldsymbol{b}, \boldsymbol{U}^{k+\frac{2}{3}}, s_2), \end{cases} \tag{4.32}$$

where 'Vec' denotes the vectorization operation from a matrix, and 'Mat' the reverse operation, converting a vector to the corresponding matrix. In

practice, we use the `reshape` command in Matlab for this. The matrix $\mathcal{K}_{\rm c}$ is the all-at-once matrix obtained with larger space and time discretization parameters $\Delta T = 2\Delta t$ and $\Delta X = 2\Delta x$, i.e.,

$$\mathcal{K}_{\rm c} = \underbrace{\begin{bmatrix} r_1^c & & & \\ -r_2^c & r_1^c & & \\ & \ddots & \ddots & \\ & & -r_2^c & r_1^c \end{bmatrix}}_{N_t^c \text{ blocks}},$$

where r_1^c and r_2^c are matrix polynomials of $\Delta T A_c$ with $A_c \in \mathbb{R}^{N_x^c \times N_x^c}$ being the *coarse* discrete matrix of the space derivative(s) with ΔX, e.g.,

$$\begin{cases} r_1^c = I_x^c - \Delta T A_c,\ r_2^c = I_x^c, & \text{Backward Euler,} \\ r_1^c = I_x^c - \frac{1}{2}\Delta T A_c,\ r_2^c = I_x^c + \frac{1}{2}\Delta T A_c, & \text{Trapezoidal Rule.} \end{cases}$$

In practice, we let $N_x = 2^{l_x} - 1$ and $N_t = 2^{l_t} - 1$ with $l_x, l_t \geq 2$ being integers, and thus $N_x^c = 2^{l_x - 1} - 1$ and $N_t^c = 2^{l_t - 1} - 1$. STMG is obtained naturally by applying the 2-level variant recursively.

There is an important difference between STMG and the parabolic MG method proposed forty years ago (Hackbusch 1984): for parabolic MG, one uses a *pointwise* Gauss-Seidel iteration as smoother, defined by

$$\boldsymbol{U}^{\rm new} = \mathcal{S}_{\rm GS}(\boldsymbol{b}, \boldsymbol{U}^{\rm ini}, s): \quad \begin{cases} \text{for } n = 0, 1, \ldots, N_t - 1 \\ \quad \boldsymbol{u}_{n+1}^0 = \boldsymbol{u}_{n+1}^{\rm ini}, \\ \quad \text{for } j = 0, 1, \ldots, s - 1 \\ \quad\quad (D + L)\Delta \boldsymbol{u}_{n+1}^j = \tilde{\boldsymbol{f}}_n + r_2 \boldsymbol{u}_n^s - r_1 \boldsymbol{u}_{n+1}^j, \\ \quad\quad \boldsymbol{u}_{n+1}^{j+1} = \boldsymbol{u}_{n+1}^j + \Delta \boldsymbol{u}_{n+1}^j, \\ \quad \boldsymbol{u}_{n+1}^{\rm new} = \boldsymbol{u}_{n+1}^s, \end{cases} \tag{4.33}$$

where $\boldsymbol{u}_0^s = \boldsymbol{u}_0$ and D and L represent the diagonal and upper triangular parts of r_1. Here, $\boldsymbol{U}^{\rm ini} := (\boldsymbol{u}_0^\top, (\boldsymbol{u}_1^{\rm ini})^\top, \ldots, (\boldsymbol{u}_{N_t}^{\rm ini})^\top)^\top$ and similarly $\boldsymbol{U}^{\rm new}$ consists of the vectors $\boldsymbol{u}_0$ and $\boldsymbol{u}_1^{\rm new}, \ldots, \boldsymbol{u}_{N_t}^{\rm new}$. This smoother operates sequentially in time: one must complete the smoothing iteration at time step n to obtain $\boldsymbol{u}_n^s$, which is necessary for performing the smoothing iteration at time step $n+1$. After smoothing, one restricts the residual $\boldsymbol{b} - \mathcal{K}\boldsymbol{U}^{\rm new}$ in space-time, like in standard multigrid, to a coarser grid. There, one solves a coarse problem (and repeats this procedure recursively in practice). Hackbusch in 1984 focused on coarsening in space for this method and found that for the heat equation, parabolic MG converges very rapidly. Gander and Lunet (Gander and Lunet 2024) examined the performance of a 2-level version of the parabolic MG method that coarsens both in space and time and found that it converges only slowly then. This slow convergence was

already improved upon in (Horton and Vandewalle 1995) by using special multigrid components adapted to the interpretation of the time direction as a strongly advective term, see also (Janssen and Vandewalle 1996, Van Lent and Vandewalle 2002) for multigrid waveform relaxation variants.

Returning to STMG (4.32), the smoother plays a crucial role in achieving good performance. The fundamental concept in designing an effective smoother is to eliminate as much of the high-frequency error components as possible within a minimal number of smoothing iterations. This allows the remaining low-frequency errors to be well-represented on the coarse grids and then being removed there through coarse grid correction. A valuable tool for accomplishing this objective is *Local Fourier Analysis* (LFA), which involves neglecting the initial and boundary conditions of the problem and just focusing on how the finite difference stencil affects a given Fourier mode in the error,

$$u_{n,m}^{j} = C_{\omega,\xi}^{j} e^{\mathrm{i}\omega n\Delta t} e^{\mathrm{i}\xi m\Delta x}, \tag{4.34}$$

where $\boldsymbol{u}_n^j := (u_{n,1}^j, \ldots, u_{n,N_x}^j)^\top$ and $\mathrm{i} = \sqrt{-1}$. To apply LFA for the damped Jacobi iteration (4.30), we consider the 1D heat equation (2.3) discretized using centered finite differences in space and backward Euler in time. In this case,

$$A = \frac{1}{\Delta x^2}\mathrm{tri}(1,-2,1), \quad r_1 = I_x - \Delta t A, \quad r_2 = I_x.$$

For each iteration j, the block Jacobi iteration (4.30) consists of the N_t difference equations

$$r_1(\boldsymbol{u}_{n+1}^{j+1} - \boldsymbol{u}_{n+1}^{j}) = -\eta(r_1\boldsymbol{u}_{n+1}^{j} - r_2\boldsymbol{u}_n^{j}), \tag{4.35}$$

where we set the right-hand side $\boldsymbol{b}$ in (4.30) to zero and consider $\boldsymbol{U}^j$ as the error at the j-th iteration.

To apply LFA to (4.35), we first consider the result when the space discrete operator A acts on the Fourier mode $u_{n+1,m}^l$ (with $l = j, j+1$),

$$\begin{aligned} Au_{n+1,m}^{l} &= C_{\omega,\xi}^{l} e^{\mathrm{i}\omega(n+1)\Delta t} \frac{e^{\mathrm{i}\xi(m-1)\Delta x} - 2e^{\mathrm{i}\xi m\Delta x} + e^{\mathrm{i}\xi(m+1)\Delta x}}{\Delta x^2} \\ &= C_{\omega,\xi}^{l} e^{\mathrm{i}\omega(n+1)\Delta t} e^{\mathrm{i}\xi m\Delta x} \frac{e^{-\mathrm{i}\xi\Delta x} - 2 + e^{\mathrm{i}\xi\Delta x}}{\Delta x^2} \\ &= \frac{2(\cos(\xi\Delta x) - 1)}{\Delta x^2} C_{\omega,\xi}^{l} e^{\mathrm{i}\omega(n+1)\Delta t} e^{\mathrm{i}\xi m\Delta x}. \end{aligned} \tag{4.36}$$

Hence,

$$r_1(\boldsymbol{u}_{n+1}^{j+1} - \boldsymbol{u}_{n+1}^{j}) = \left(1 - \frac{2\Delta t(\cos(\xi\Delta x) - 1)}{\Delta x^2}\right)(C_{\omega,\xi}^{j+1} - C_{\omega,\xi}^{j}) e^{\mathrm{i}\omega(n+1)\Delta t} e^{\mathrm{i}\xi \boldsymbol{x}_h},$$

and

$$r_1\boldsymbol{u}_{n+1}^j - r_2\boldsymbol{u}_n^j = \boldsymbol{u}_{n+1}^j - \boldsymbol{u}_n^j - \Delta t A\boldsymbol{u}_{n+1}^j$$
$$= \left(1 - e^{-\mathrm{i}\omega\Delta t} - \frac{2\Delta t(\cos(\xi\Delta x)-1)}{\Delta x^2}\right) C_{\omega,\xi}^j e^{\mathrm{i}\omega(n+1)\Delta t} e^{\mathrm{i}\xi \boldsymbol{x}_h},$$

where $\boldsymbol{x}_h = \mathrm{Vec}(m\Delta x)$. Now, from (4.35) we have

$$\left(1 - \frac{2\Delta t(\cos(\xi\Delta x)-1)}{\Delta x^2}\right)(C_{\omega,\xi}^{j+1} - C_{\omega,\xi}^j)$$
$$= \eta\left(1 - e^{-\mathrm{i}\omega\Delta t} - \frac{2\Delta t(\cos(\xi\Delta x)-1)}{\Delta x^2}\right) C_{\omega,\xi}^j,$$

i.e., $C_{\omega,\xi}^{j+1} = \rho(\omega,\xi,\eta)C_{\omega,\xi}^j$ with ρ being the convergence factor given by

$$\rho(\omega,\xi,\eta) = 1 - \eta\left(1 - \frac{e^{-\mathrm{i}\omega\Delta t}}{1 + \frac{2\Delta t}{\Delta x^2}(1-\cos(\xi\Delta x))}\right), \tag{4.37}$$

where $\omega\Delta t \in (-\pi,\pi)$ and $\xi\Delta x \in (-\pi,\pi)$. By calculating the maximum of ρ with respect to ξ and ω and then minimizing the maximum, the following result was proved in (Gander and Lunet 2024, Chapter 4), see (Gander and Neumüller 2016) for a comprehensive analysis for more general discretizations:

Theorem 4.9. *For the 1D heat equation discretized with centered finite differences and backward Euler, the optimal choice of η used in the damped Jacobi smoother* (4.30) *always permitting time coarsening is $\eta = \frac{1}{2}$. With this choice, all high frequencies in time, $\omega \in \pm\left(\frac{\pi}{2\Delta t}, \frac{\pi}{\Delta t}\right)$ are damped by at least a factor of $\frac{1}{\sqrt{2}}$. If in addition the mesh parameters satisfy $\frac{\Delta t}{\Delta x^2} \geq \frac{1}{\sqrt{2}}$, then also the high frequencies in space, $\xi \in \pm\left(\frac{\pi}{2\Delta x}, \frac{\pi}{\Delta x}\right)$ are damped by at least a factor of $\frac{1}{\sqrt{2}}$, and one can do space coarsening as well.*

A refined analysis concerning the optimality can be found in (Chaudet-Dumas, Gander and Pogozelskyte 2024).

For the advection-diffusion equation (2.5), we can also apply LFA to 2-level STMG. Here, the discrete matrix obtained from centered finite differences is $A = \frac{\nu}{\Delta x^2}\mathrm{Tri}(1,-2,1) + \frac{1}{2\Delta x}\mathrm{Tri}(-1,0,1)$. Similar to (4.36), the result when the spatial discrete operator A acts on the Fourier mode $u_{n+1,m}^l$ (with $l = j, j+1$) is

$$Au_{n+1,m}^l = \left[\frac{2\nu(\cos(\xi\Delta x)-1)}{\Delta x^2} + \frac{e^{\mathrm{i}\xi\Delta x} - e^{-\mathrm{i}\xi\Delta x}}{2\Delta x}\right] C_{\omega,\xi}^l e^{\mathrm{i}\omega(n+1)\Delta t} e^{\mathrm{i}\xi m\Delta x}$$
$$= \left[\frac{2\nu(\cos(\xi\Delta x)-1)}{\Delta x^2} + \frac{\mathrm{i}\sin(\xi\Delta x)}{\Delta x}\right] C_{\omega,\xi}^l e^{\mathrm{i}\omega(n+1)\Delta t} e^{\mathrm{i}\xi m\Delta x}.$$

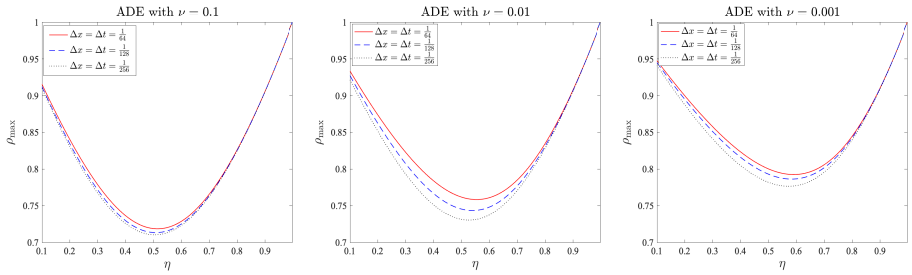

Figure 4.39. Maximum of the convergence factor over the high frequencies for the advection-diffusion equation (ADE) for three values of the diffusion parameter ν, i.e., $\rho_{\max} = \max_{(\Delta x\xi, \Delta t\omega)\in(-\pi,\pi)\times(\frac{\pi}{2},\pi)} \rho(\omega, \xi, \eta)$.

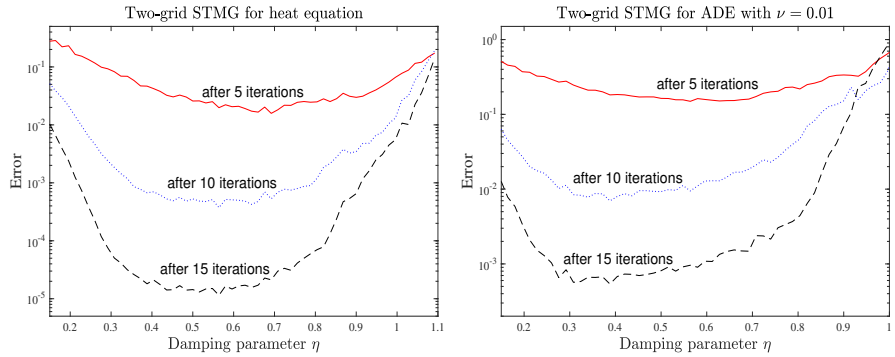

Figure 4.40. Dependence of the 2-level STMG error after $k = 5, 10, 15$ iterations on the choice of the damping parameter η. Left: heat equation; Right: advection-diffusion equation with $\nu = 0.01$. Here, we use 1 block Jacobi smoothing iteration for STMG.

From this, we obtain the convergence factor ρ in Fourier space as

$$\rho(\omega, \xi, \eta) = 1 - \eta \left(1 - \frac{e^{-\mathrm{i}\omega\Delta t}}{1 + \frac{2\nu\Delta t}{\Delta x^2}(1 - \cos(\xi\Delta x)) + \mathrm{i}\frac{\Delta t}{\Delta x}\sin(\xi\Delta x)} \right). \quad (4.38)$$

Heuristically, we can still use $\eta = \frac{1}{2}$ as damping parameter for coarsening in time; see the illustration in Figure 4.39. The validity of the choice $\eta = \frac{1}{2}$ for the damping parameter is further illustrated in Figure 4.40 in a numerical experiment for the 2-level variant of STMG: for both the heat equation and the advection-diffusion equation (ADE), we show the errors after 5, 10, and 15 iterations for several values of η. Clearly, $\eta = \frac{1}{2}$ is a reasonable choice to minimize the error in 2-level STMG for both equations.

We show in Figure 4.41 the convergence behavior of 2-level STMG for both the heat equation and the advection-diffusion equation with $\eta = \frac{1}{2}$ and

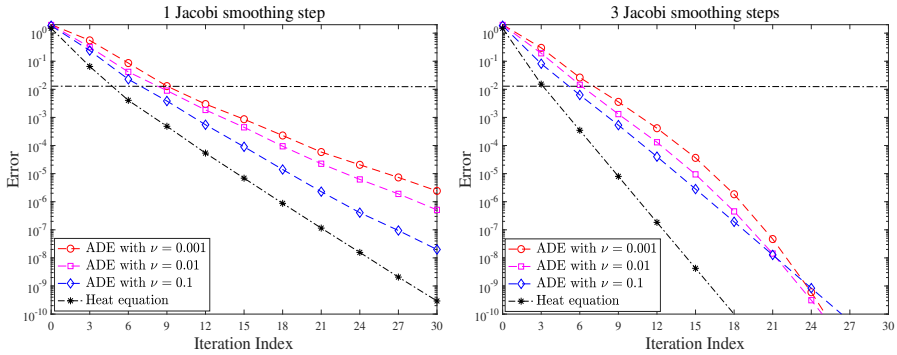

Figure 4.41. Measured error of 2-level STMG with 1 and 3 block Jacobi smoothing steps for the advection-diffusion equation (ADE) and the heat equation with damping parameter $\eta = \frac{1}{2}$.

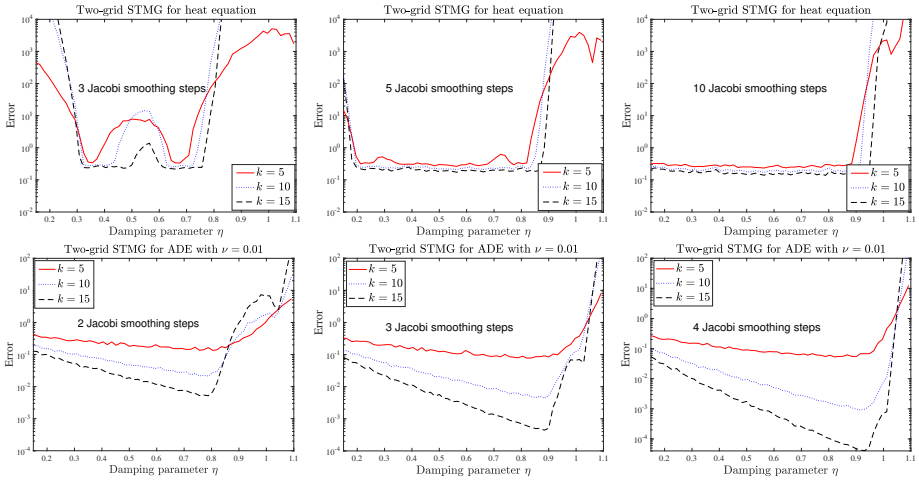

Figure 4.42. Error of STMG using the Trapezoidal Rule and different numbers of smoothing steps for $\eta \in [0.1, 1.1]$.

three values of ν. For both equations, 2-level STMG converges faster when the number of smoothing iterations is increased. Compared to the heat equation, the convergence rate is worse for the advection-diffusion equation, but interestingly it is less sensitive to ν when the number of smoothing iterations is large and a superlinear convergence mechanism sets in, as we see in the right panel.

The convergence rate of STMG, however, depends on the choice of the time integrator. The results in Figure 4.42 for 2-level STMG reveal that using the Trapezoidal Rule as time integrator results in a substantially poorer convergence rate compared to using Backward Euler. In particular, for the

cores	time steps	dof	iter	time	fwd. sub.	time steps	dof	iter	time
1	2	59 768	7	28.8	19.0	512	15 300 608	7	7 635.2
2	4	119 536	7	29.8	37.9	512	15 300 608	7	3 821.7
4	8	239 072	7	29.8	75.9	512	15 300 608	7	1 909.9
8	16	478 144	7	29.9	152.2	512	15 300 608	7	954.2
16	32	956 288	7	29.9	305.4	512	15 300 608	7	477.2
32	64	1 912 576	7	29.9	613.6	512	15 300 608	7	238.9
64	128	3 825 152	7	29.9	1 220.7	512	15 300 608	7	119.5
128	256	7 650 304	7	29.9	2 448.4	512	15 300 608	7	59.7
256	512	15 300 608	7	30.0	4 882.4	512	15 300 608	7	30.0
512	1 024	30 601 216	7	29.9	9 744.2	524 288	15 667 822 592	7	15 205.9
1 024	2 048	61 202 432	7	30.0	19 636.9	524 288	15 667 822 592	7	7 651.5
2 048	4 096	122 404 864	7	29.9	38 993.1	524 288	15 667 822 592	7	3 825.3
4 096	8 192	244 809 728	7	30.0	81 219.6	524 288	15 667 822 592	7	1 913.4
8 192	16 384	489 619 456	7	30.0	162 551.0	524 288	15 667 822 592	7	956.6
16 384	32 768	979 238 912	7	30.0	313 122.0	524 288	15 667 822 592	7	478.1
32 768	65 536	1 958 477 824	7	30.0	625 686.0	524 288	15 667 822 592	7	239.3
65 536	131 072	3 916 955 648	7	30.0	1 250 210.0	524 288	15 667 822 592	7	119.6
131 072	262 144	7 833 911 296	7	30.0	2 500 350.0	524 288	15 667 822 592	7	59.8
262 144	524 288	15 667 822 592	7	30.0	4 988 060.0	524 288	15 667 822 592	7	30.0

Table 4.3. Weak (left) and strong (right) scaling results of a modern Space-Time MultiGrid (STMG) method applied to a 3D heat equation. Solution times of classical time stepping with best possible parallelization in space only are also shown in the column 'fwd. sub.'.

heat equation, 2-level STMG appears to have convergence problems regardless of the damping parameter adjustments, even with up to 10 smoothing steps. Interestingly, 2-level STMG converges for the advection-diffusion equation, and doing more smoothing steps enhances the convergence rate. Nonetheless, the optimal damping parameter is found to be $\eta \approx 0.8$, in contrast to $\eta = \frac{1}{2}$ we obtained for Backward Euler.

In real large scale parallel computations on todays super computers, excellent weak and strong scaling is achieved with the full STMG method, as shown in Table 4.3 for a 3D heat equation model problem, taken from (Gander and Neumüller 2016).

We next extend STMG to nonlinear problems of the form

$$\boldsymbol{u}' = f(\boldsymbol{u}),\ \boldsymbol{u}(0) = \boldsymbol{u}_0,\ t \in (0,T), \tag{4.39}$$

where $\boldsymbol{u} \in \mathbb{R}^{N_x}$ and $f : \mathbb{R}^{N_x} \to \mathbb{R}^{N_x}$ is defined by the discretization of a PDE in space. To describe the idea, we apply the linear-θ method to the nonlinear system of ODEs (4.39), leading to the all-at-once system

$$\underbrace{(B \otimes I_x)\boldsymbol{U} - \Delta t(\tilde{B} \otimes I_x) f(\boldsymbol{U})}_{:=\mathcal{K}(\boldsymbol{U})} = \boldsymbol{b}, \tag{4.40}$$

where $\boldsymbol{b} = (\boldsymbol{u}_0^\top + \Delta t(1-\theta)f^\top(\boldsymbol{u}_0), 0, \ldots, 0)^\top$, $\boldsymbol{U} = (\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_{N_t}^\top)^\top$ and

$$B := \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}, \ \tilde{B} := \begin{bmatrix} \theta & & & \\ 1-\theta & \theta & & \\ & \ddots & \ddots & \\ & & 1-\theta & \theta \end{bmatrix}, \ f(\boldsymbol{U}) := \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{N_t} \end{bmatrix}.$$

To formulate STMG (4.40), similar to (4.30), we first define a nonlinear block Jacobi smoother $\boldsymbol{U}^{\text{new}} = \mathcal{S}_{\text{non},\eta}(\boldsymbol{b}, \boldsymbol{U}^{\text{ini}}, s)$ by

$$\begin{cases} \tilde{\boldsymbol{U}}^0 = \boldsymbol{U}^{\text{ini}}, \\ \text{for } j = 0, 1, \ldots, s-1: \\ \quad \text{solve } \Delta\tilde{\boldsymbol{U}}^j - \Delta t\theta f(\Delta\tilde{\boldsymbol{U}}^j) = \eta(\boldsymbol{b} - \mathcal{K}(\tilde{\boldsymbol{U}}^j)), \\ \quad \tilde{\boldsymbol{U}}^{j+1} = \tilde{\boldsymbol{U}}^j + \Delta\tilde{\boldsymbol{U}}^j, \\ \boldsymbol{U}^{\text{new}} = \tilde{\boldsymbol{U}}^s, \end{cases} \tag{4.41}$$

where the correction term $\Delta\tilde{\boldsymbol{U}}^j$ is solved via an inner solver, e.g., Newton's iteration. We can however not obtain a theoretically optimized estimate for the damping parameter η in (4.41), since LFA can not be used in the nonlinear case.

Following (Brandt 1977), we now define a non-linear STMG method for (4.40) using the full approximation scheme,

$$\begin{cases} \boldsymbol{U}^{k+\frac{1}{3}} = \mathcal{S}_{\text{non},\eta}(\boldsymbol{b}, \boldsymbol{U}^k, s_1), \\ \boldsymbol{r} = \boldsymbol{b} - \mathcal{K}(\boldsymbol{U}^{k+\frac{1}{3}}), \\ \boldsymbol{r}_{\text{c}} = [R_x \text{Mat}(\boldsymbol{r})]R_t^\top, \ \boldsymbol{U}_{\text{c}}^{k+\frac{1}{3}} = [R_x \text{Mat}(\boldsymbol{U}^{k+\frac{1}{3}})]R_t^\top, \\ \text{Solve } \mathcal{K}_{\text{c}}(\boldsymbol{U}_{\text{c}}^{k+\frac{2}{3}}) = \boldsymbol{r}_{\text{c}} + \mathcal{K}_{\text{c}}(\boldsymbol{U}_{\text{c}}^{k+\frac{1}{3}}), \\ \boldsymbol{e}_{\text{c}} = \boldsymbol{U}_{\text{c}}^{k+\frac{2}{3}} - \boldsymbol{U}_{\text{c}}^{k+\frac{1}{3}}, \ \boldsymbol{e} = [P_x \text{Mat}(\boldsymbol{e}_{\text{c}})]P_t^\top, \\ \boldsymbol{U}^{k+\frac{2}{3}} = \boldsymbol{U}^{k+\frac{1}{3}} + \text{Vec}(\boldsymbol{e}), \\ \boldsymbol{U}^{k+1} = \mathcal{S}_{\text{non},\eta}(\boldsymbol{b}, \boldsymbol{U}^{k+\frac{2}{3}}, s_2). \end{cases} \tag{4.42}$$

In Figure 4.43, we show the measured error of this 2-level STMG method for Burgers' equation (2.6) with conditions. We use two values of the diffusion parameter ν, and we see that the convergence is heavily dependent on this parameter: with enough diffusion, STMG works very well also in the nonlinear setting, whereas when the diffusion is getting smaller, the convergence of STMG deteriorates, as in the linear case. Here, we used $\eta = \frac{1}{4}$ for the damping parameter, which was found to be the best choice in our numerical experiments.

For parabolic problems, STMG presented in this section stands out as by far the most effective time-parallel solver currently available, but it is intrusive in nature, unlike the Parareal algorithm. However, when dealing

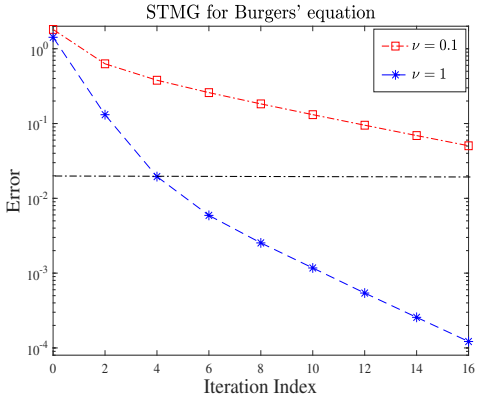

Figure 4.43. Measured error of STMG with two block Jacob smoothing steps for Burgers' equation.

with hyperbolic problems, as shown in Figures 4.41 and 4.43, STMG appears to be less efficient, indicating that additional efforts are required in this domain. Furthermore, as highlighted in the top row of Figure 4.42, even for parabolic problems the convergence rate of STMG depends on the time integrator used, and this dependency deserves further investigation as well.

5. Conclusions

We explained in this paper the important difference for time parallel time integration, or PinT (Parallel in Time) methods, when applied to hyperbolic or parabolic problems. For parabolic problems, which tend to forget a lot of information in time, and thus have solutions that are local in time, there are many very effective PinT techniques, like Parareal, Space-Time MultiGrid (STMG), ParaExp and ParaDiag, and Waveform Relaxation (WR) techniques based on Domain Decomposition (DD). For hyperbolic problems, which retain very fine solution features over very long time, only some of these techniques are effective, like ParaExp, ParaDiag, and Schwarz Waveform Relaxation (SWR), especially in relation to tent pitching. For more information, please take a look at the recent research monograph (Gander and Lunet 2024), which contains an up to date treatment of PinT methods, giving for each the historical content, a simple but complete and self contained convergence analysis, and also small Matlab codes that can be directly executed. Codes used to produce the results in this paper are available from `https://github.com/wushulin/ActaPinT`.

REFERENCES

X. Antoine and E. Lorin (2016), 'Lagrange–Schwarz waveform relaxation domain decomposition methods for linear and nonlinear quantum wave problems',

Applied Mathematics Letters **57**, 38–45.

X. Antoine and E. Lorin (2017), 'An analysis of Schwarz waveform relaxation domain decomposition methods for the imaginary-time linear Schrödinger and Gross-Pitaevskii equations', *Numerische Mathematik* **137**, 923–958.

E. Audusse, P. Dreyfuss and B. Merlet (2010), 'Optimized Schwarz waveform relaxation for the primitive equations of the ocean', *SIAM Journal on Scientific Computing* **32**(5), 2908–2936.

A. O. H. Axelsson and J. G. Verwer (1985), 'Boundary value techniques for initial value problems in ordinary differential equations', *Math. Comput.* **45**(171), 153–171.

A. Bellen and M. Zennaro (1989), 'Parallel algorithms for initial-value problems for difference and differential equations', *J. Comput. Appl. Math.* **25**, 341–350.

D. Bennequin, M. J. Gander and L. Halpern (2009), 'A homographic best approximation problem with application to optimized Schwarz waveform relaxation', *Math. Comput.* **78**(265), 185–223.

D. Bennequin, M. J. Gander, L. Gouarin and L. Halpern (2016*a*), 'Optimized Schwarz waveform relaxation for advection reaction diffusion equations in two dimensions', *Numerische Mathematik* **134**, 513–567.

D. Bennequin, M. J. Gander, L. Gouarin and L. Halpern (2016*b*), 'Optimized Schwarz waveform relaxation for advection reaction diffusion equations in two dimensions', *Numer. Math.* **134**, 513–567.

C. Besse and F. Xing (2017), 'Schwarz waveform relaxation method for one-dimensional Schrödinger equation with general potential', *Numerical Algorithms* **74**, 393–426.

D. A. Bini, G. Latouche and B. Meini (2005), *Numerical Methods for Structured Markov Chains*, Oxford University Press.

M. Bjørhus (1995), On Domain Decomposition, Subdomain Iteration and Waveform Relaxation, PhD thesis, University of Trondheim, Norway.

K. Böhmer and H. J. Stetter (1984), *Defect Correction Methods, Theory and Applications*, Springer–Verlag, New York.

M. Bolten, D. Moser and R. Speck (2017), 'A multigrid perspective on the parallel full approximation scheme in space and time', *Numer. Linear Alg. Appl.* **24**(6), e2110.

M. Bolten, D. Moser and R. Speck (2018), 'Asymptotic convergence of the parallel full approximation scheme in space and time for linear problems', *Numer. Linear Alg. Appl.* **25**(6), e2208.

A. Bouillon, G. Samaey and K. Meerbergen (2023), 'Diagonalization-based preconditioners and generalized convergence bounds for ParaOpt', *arXiv preprint arXiv:2304.09235*.

A. Brandt (1977), 'Multi-level adaptive solutions to boundary-value problems', *Math. Comput.* **31**(138), 333–390.

L. Brugnano and D. Trigiante (2003), *Solving Differential Problems by Multistep Initial and Boundary Value Methods*, Gordon and Breach Science Publ., Amsterdam.

L. Brugnano, F. Mazzia and D. Trigiante (1993), 'Parallel implementation of BVM methods', *Appl. Numer. Math.* **11**(1-3), 115–124.

X.-C. Cai (1991), 'Additive Schwarz algorithms for parabolic convection-diffusion equations', *Numer. Math.* **60**(1), 41–61.

X.-C. Cai (1994), 'Multiplicative Schwarz methods for parabolic problems', *SIAM J. Sci. Comput.* **15**(3), 587–603.

R. H. Chan and M. K. Ng (1996), 'Conjugate gradient methods for Toeplitz systems', *SIAM Rev.* **38**(3), 427–482.

P. Chartier and B. Philippe (1993), 'A parallel shooting technique for solving dissipative ODEs', *Computing* **51**, 209–236.

B. Chaudet-Dumas, M. J. Gander and A. Pogozelskyte (2024), 'An optimized space-time multigrid algorithm for parabolic PDEs', *arXiv preprint arXiv:2302.13881*.

M. M. Chawla (1983), 'Unconditionally stable Noumerov-type methods for second order differential equations', *BIT* **23**, 541–542.

A. J. Christlieb, C. B. Macdonald and B. W. Ong (2010), 'Parallel high-order integrators', *SIAM J. Sci. Comput.* **32**(2), 818–835.

G. Ciaramella and M. J. Gander (2017), 'Analysis of the parallel Schwarz method for growing chains of fixed-sized subdomains: Part I', *SIAM Journal on Numerical Analysis* **55**(3), 1330–1356.

G. Ciaramella and M. J. Gander (2018*a*), 'Analysis of the parallel Schwarz method for growing chains of fixed-sized subdomains: Part II', *SIAM Journal on Numerical Analysis* **56**(3), 1498–1524.

G. Ciaramella and M. J. Gander (2018*b*), 'Analysis of the parallel Schwarz method for growing chains of fixed-sized subdomains: Part III', *Electron. Trans. Numer. Anal* **49**, 201–243.

G. Ciaramella and M. J. Gander (2022), *Iterative methods and preconditioners for systems of linear equations*, SIAM.

G. Ciaramella, M. J. Gander and I. Mazzieri (2023), Unmapped tent pitching schemes by waveform relaxation, in *27th international Conference of Domain Decomposition Methods*, Springer.

J. Cortial and C. Farhat (2009), 'A time-parallel implicit method for accelerating the solution of non-linear structural dynamics problems', *International Journal for Numerical Methods in Engineering* **77**(4), 451–470.

Y. Courvoisier and M. J. Gander (2013), Time domain Maxwell equations solved with Schwarz waveform relaxation methods, in *Domain Decomposition Methods in Science and Engineering XX*, Springer, pp. 263–270.

F. Danieli and A. J. Wathen (2021), 'All-at-once solution of linear wave equations', *Numer. Linear Algebra Appl.* **28**(6), e2386.

F. Danieli, B. S. Southworth and A. J. Wathen (2022), 'Space-time block preconditioning for incompressible flow', *SIAM J. Sci. Comput.* **44**(1), A337–A363.

H. De Sterck, R. D. Falgout and O. A. Krzysik (2023*a*), 'Fast multigrid reduction-in-time for advection via modified semi-lagrangian coarse-grid operators', *SIAM J. Sci. Comput.* **45**(4), A1890–A1916.

H. De Sterck, R. D. Falgout, S. Friedhoff, O. A. Krzysik and S. P. MacLachlan (2021), 'Optimizing multigrid reduction-in-time and parareal coarse-grid operators for linear advection', *Numer. Linear Alg. Appl.* **28**(4), e2367.

H. De Sterck, R. D. Falgout, O. A. Krzysik and J. B. Schroder (2023*b*), 'Efficient multigrid reduction-in-time for method-of-lines discretizations of linear advection', *J. Sci. Comput.* **96**(1), 1.

P. Deuflhard (2004), *Newton Methods for Nonlinear Problems*, Springer, Berlin.

V. A. Dobrev, T. Kolev, N. A. Petersson and J. B. Schroder (2017*a*), 'Two-level convergence theory for multigrid reduction in time (mgrit)', *SIAM Journal on Scientific Computing* **39**(5), S501–S527.

V. A. Dobrev, T. V. Kolev, N. A. Petersson and J. B. Schroder (2017*b*), 'Two-level convergence theory for multigrid reduction in time (mgrit)', *SIAM J. Sci. Comput.* **39**(5), S501–S527.

A. Dutt, L. Greengard and V. Rokhlin (2000), 'Spectral deferred correction methods for ordinary differential equations', *BIT* **40**(2), 241–266.

M. Emmett and M. L. Minion (2012), 'Toward an efficient parallel in time method for partial differential equations', *Communi. Appl. Math. Comput. Sci.* **7**(1), 105–132.

R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan and J. B. Schroder (2014), 'Parallel time integration with multigrid', *SIAM Journal on Scientific Computing* **36**(6), C635–C661.

C. Farhat and M. Chandesris (2003), 'Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid–structure applications', *International Journal for Numerical Methods in Engineering* **58**(9), 1397–1434.

C. Farhat, J. Cortial, C. Dastillung and H. Bavestrello (2006), 'Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses', *International journal for numerical methods in engineering* **67**(5), 697–724.

L. Fox (1954), 'A note on the numerical integration of first-order differential equations', *Quart. J. Mech. Appl. Math.* **7**(3), 367–378.

L. Fox and A. R. Mitchell (1957), 'Boundary-value techniques for the numerical solution of initial-value problems in ordinary differential equations', *Quart. J. Mech. Appl. Math.* **10**(2), 232–243.

M. J. Gander (1997), Analysis of Parallel Algorithms for Time Dependent Partial Differential Equations, PhD thesis, Stanford.

M. J. Gander (1998), 'A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations', *Numerical Linear Algebra with Applications* **6**, 125–145.

M. J. Gander (1999), 'A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations', *Numer. Linear Algebra Appl.* **6**(2), 125–145.

M. J. Gander (2008), 'Schwarz methods over the course of time', *Electron. Trans. Numer. Anal* **31**(5), 228–255.

M. J. Gander (2015), 50 years of time parallel time integration, in *Multiple Shooting and Time Domain Decomposition Methods*, Springer, pp. 69–113.

M. J. Gander (2017), 'Three different multigrid interpretations of the parareal algorithm and an adaptive variant', *Workshop on Space-time Methods for Time-dependent Partial Differential Equations*.

M. J. Gander and S. Güttel (2013), 'ParaExp: A parallel integrator for linear initial-value problems', *SIAM J. Sci. Comput.* **35**(2), C123–C142.

M. J. Gander and E. Hairer (2008), Nonlinear convergence analysis for the parareal algorithm, in *Domain Decomposition Methods in Science and Engineering XVII* (O. B. Widlund and D. E. Keyes, eds), Vol. 60 of *Lecture Notes in Computational Science and Engineering*, Springer, pp. 45–56.

M. J. Gander and E. Hairer (2014), 'Analysis for parareal algorithms applied to Hamiltonian differential equations', *Journal of Computational and Applied Mathematics* **259**, 2–13.

M. J. Gander and L. Halpern (2004), 'Absorbing boundary conditions for the wave equation and parallel computing', *Math. Comput.* **74**(249), 153–176.

M. J. Gander and L. Halpern (2007), 'Optimized Schwarz waveform relaxation methods for advection reaction diffusion problems', *SIAM J. Numer. Anal.* **45**(2), 666–697.

M. J. Gander and L. Halpern (2017), Time parallelization for nonlinear problems based on diagonalization, in *Domain decomposition methods in science and engineering XXIII 23*, Springer, pp. 163–170.

M. J. Gander and T. Lunet (2020*a*), A reynolds number dependent convergence estimate for the parareal algorithm, in *Domain Decomposition Methods in Science and Engineering XXV 25*, Springer, pp. 277–284.

M. J. Gander and T. Lunet (2020*b*), 'Toward error estimates for general space-time discretizations of the advection equation', *Comput. Visual Sci.* **23**, 1–14.

M. J. Gander and T. Lunet (2024), *Time Parallel Time Integration*, SIAM.

M. J. Gander and M. Neumüller (2016), 'Analysis of a new space-time parallel multigrid algorithm for parabolic problems', *SIAM J. Sci. Comput.* **38**(4), A2173–A2208.

M. J. Gander and D. Palitta (2024), 'A new ParaDiag time-parallel time integration method', *SIAM J. Sci. Comput.* **46**(2), A697–A718.

M. J. Gander and C. Rohde (2005*a*), 'Overlapping Schwarz waveform relaxation for convection-dominated nonlinear conservation laws', *SIAM journal on Scientific Computing* **27**(2), 415–439.

M. J. Gander and C. Rohde (2005*b*), 'Overlapping Schwarz waveform relaxation for convection-dominated nonlinear conservation laws', *SIAM J. Sci. Comput.* **27**(2), 415–439.

M. J. Gander and A. M. Stuart (1998), 'Space-time continuous analysis of waveform relaxation for the heat equation', *SIAM J. Sci. Comput.* **19**(6), 2014–2031.

M. J. Gander and S. Vandewalle (2007), 'Analysis of the parareal time-parallel time-integration method', *SIAM J. Sci. Comput.* **29**(2), 556–578.

M. J. Gander and S.-L. Wu (2019), 'Convergence analysis of a periodic-like waveform relaxation method for initial-value problems via the diagonalization technique', *Numer. Math.* **143**, 489–527.

M. J. Gander and S.-L. Wu (2020), 'A diagonalization-based parareal algorithm for dissipative and wave propagation problems', *SIAM J. Numer. Anal.* **58**(5), 2981–3009.

M. J. Gander, S. Güttel and M. Petcu (2018*a*), A nonlinear paraexp algorithm, in *Domain Decomposition Methods in Science and Engineering XXIV 24*, Springer, pp. 261–270.

M. J. Gander, L. Halpern and F. Nataf (1999), Optimal convergence for overlapping and non-overlapping Schwarz waveform relaxation, in *Eleventh international Conference of Domain Decomposition Methods* (C.-H. Lai, P. Bjørstad, M. Cross and O. Widlund, eds), ddm.org.

M. J. Gander, L. Halpern and F. Nataf (2003), 'Optimal Schwarz waveform relaxation for the one dimensional wave equation', *SIAM J. Numer. Anal.* **41**(5), 1643–1681.

M. J. Gander, L. Halpern, F. Hubert and S. Krell (2018*b*), 'Optimized Schwarz methods for anisotropic diffusion with discrete duality finite volume discretizations', *Moroccan Journal of Pure and Applied Analysis*.

M. J. Gander, L. Halpern, F. Hubert and S. Krell (2021*a*), 'Discrete optimization of Robin transmission conditions for anisotropic diffusion with discrete duality finite volume methods', *Vietnam Journal of Mathematics* **49**(4), 1349–1378.

M. J. Gander, L. Halpern, J. Rannou and J. Ryan (2019), 'A direct time parallel solver by diagonalization for the wave equation', *SIAM J. Sci. Comput.* **41**(1), A220–A245.

M. J. Gander, L. Halpern, J. Ryan and T. T. B. Tran (2016*a*), A direct solver for time parallelization, in *Domain decomposition methods in science and engineering XXII 22*, Springer, pp. 491–499.

M. J. Gander, F. Kwok and B. Mandal (2016*b*), 'Dirichlet-Neumann and Neumann-Neumann waveform relaxation algorithms for parabolic problems', *ETNA* **45**, 424–456.

M. J. Gander, F. Kwok and B. C. Mandal (2021*b*), 'Dirichlet–Neumann waveform relaxation methods for parabolic and hyperbolic problems in multiple subdomains', *BIT Numerical Mathematics* **61**, 173–207.

M. J. Gander, F. Kwok and H. Zhang (2018*c*), 'Multigrid interpretations of the parareal algorithm leading to an overlapping variant and mgrit', *Journal of Computing and Visualization in Science* **19**(3), 59–74.

M. J. Gander, F. Kwok and H. Zhang (2018*d*), 'Multigrid interpretations of the parareal algorithm leading to an overlapping variant and MGRIT', *Comput. Visual Sci.* **19**(3), 59–74.

M. J. Gander, J. Liu, S.-L. Wu, X. Yue and T. Zhou (2021*c*), 'ParaDiag: Parallel-in-time algorithms based on the diagonalization technique', *arXiv preprint arXiv:2005.09158*.

M. J. Gander, T. Lunet and A. Pogoželskytė (2023*a*), Convergence of parareal for a vibrating string with viscoelastic damping, in *Domain Decomposition Methods in Science and Engineering XXVI*, Springer, pp. 435–442.

M. J. Gander, T. Lunet, D. Ruprecht and R. Speck (2023*b*), 'A unified analysis framework for iterative parallel-in-time algorithms', *SIAM J. Sci. Comput.* **45**(5), A2275–A2303.

M. J. Gander, S. B. Lunowa and C. Rohde (2023*c*), 'Non-overlapping Schwarz waveform-relaxation for nonlinear advection-diffusion equations', *SIAM Journal on Scientific Computing* **45**(1), A49–A73.

M. J. Gander, M. Ohlberger and S. Rave (2024), A Parareal algorithm without coarse propagator?, in *Domain Decomposition Methods in Science and Engineering XXVIII*, submitted.

E. Giladi and H. B. Keller (2002), 'Space-time domain decomposition for parabolic problems', *Numer. Math.* **93**, 279–313.

J. Gopalakrishnan, M. Hochsteger, J. Schöberl and C. Wintersteiger (2020), An explicit mapped tent pitching scheme for Maxwell equations, in *Spectral and high order methods for partial differential equations—ICOSAHOM 2018*, pp. 359–369.

J. Gopalakrishnan, J. Schöberl and C. Wintersteiger (2017), 'Mapped tent pitching schemes for hyperbolic systems', *SIAM J. Sci. Comput.* **39**(6), B1043–B1063.

X. M. Gu and S.-L. Wu (2020), 'A parallel-in-time iterative algorithm for Volterra partial integro-differential problems with weakly singular kernel', *J. Comput. Phys.* **417**, 109576.

D. Guibert and D. Tromeur-Dervout (2007), Parallel deferred correction method for CFD problems, in *Parallel Computational Fluid Dynamics 2006*, Elsevier, pp. 131–138.

W. Hackbusch (1984), Parabolic multi-grid methods, in *Computing Methods in Applied Sciences and Engineering, VI* (R. Glowinski and J.-L. Lions, eds), North-Holland, pp. 189–197.

L. Halpern and J. Szeftel (2010), 'Optimized and quasi-optimal Schwarz waveform relaxation for the one-dimensional Schrödinger equation', *Mathematical Models and Methods in Applied Sciences* **20**(12), 2167–2199.

B. Heinzelreiter and J. W. Pearson (2024), 'Diagonalization-based parallel-in-time preconditioners for instationary fluid flow control problems', *arXiv preprint arXiv:2405.18964*.

A. Hessenthaler, B. S. Southworth, D. Nordsletten, O. Röhrle, R. D. Falgout and J. B. Schroder (2020), 'Multilevel convergence analysis of multigrid-reduction-in-time', *SIAM Journal on Scientific Computing* **42**(2), A771–A796.

N. J. Higham (2008), *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia.

G. Horton and S. Vandewalle (1995), 'A space-time multigrid method for parabolic partial differential equations', *SIAM J. Sci. Comput.* **16**(4), 848–864.

G. Horton, S. Vandewalle and P. Worley (1995), 'An algorithm with polylog parallel complexity for solving parabolic partial differential equations', *SIAM J. Sci. Comput.* **16**(3), 531–541.

A. J. Howse, H. D. Sterck, R. D. Falgout, S. MacLachlan and J. Schroder (2019), 'Parallel-in-time multigrid with adaptive spatial coarsening for the linear advection and inviscid Burgers equations', *SIAM J. Sci. Comput.* **41**(1), A538–A565.

J. Janssen and S. Vandewalle (1996), 'Multigrid waveform relaxation of spatial finite element meshes: the continuous-time case', *SIAM Journal on Numerical Analysis* **33**(2), 456–474.

G. L. Kooij, M. A. Botchev and B. J. Geurts (2017), 'A block krylov subspace implementation of the time-parallel ParaExp method and its extension for nonlinear partial differential equations', *Journal of computational and applied mathematics* **316**, 229–246.

D. Kressner, S. Massei and J. Zhu (2022), 'Improved parallel-in-time integration via low-rank updates and interpolation', *ArXiv Preprint: 2204.03073*.

E. Lelarasmee, A. E. Ruehli and A. L. Sangiovanni-Vincentelli (1982*a*), 'The waveform relaxation method for time-domain analysis of large scale integrated circuits', *IEEE Trans. on CAD of IC and Syst.* **1**, 131–145.

E. Lelarasmee, A. E. Ruehli and A. L. Sangiovanni-Vincentelli (1982*b*), 'The waveform relaxation method for time-domain analysis of large scale integrated circuits', *IEEE Trans. CAD IC Syst.* **1**(3), 131–145.

X. L. Lin and M. Ng (2021), 'An all-at-once preconditioner for evolutionary partial differential equations', *SIAM J. Sci. Comput.* **43**(4), A2766–A2784.

J.-L. Lions, Y. Maday and G. Turinici (2001), 'A parareal in time discretization of PDEs', *C.R. Acad. Sci. Paris, Serie I* **332**, 661–668.

J. Liu and S.-L. Wu (2020), 'A fast block α-circulant preconditoner for all-at-once systems from wave equations', *SIAM J. Matrix Anal. Appl.* **41**(4), 1912–1943.

J. Liu and S.-L. Wu (2022*a*), 'Parallel-in-time preconditioner for the sinc-nyström systems', *SIAM J. Sci. Comput.* **44**, A2386–A2411.

J. Liu and S.-L. Wu (2022*b*), 'Parallel-in-time preconditioner for the Sinc-Nyström systems', *SIAM J. Sci. Comput.* **44**(4), A2386–A2411.

J. Liu, X.-S. Wang, S.-L. Wu and T. Zhou (2022), 'A well-conditioned direct PinT algorithm for first-and second-order evolutionary equations', *Adv. Comput. Math.* **48**(3), 16.

C. Lubich and A. Ostermann (1987), 'Multi-grid dynamic iteration for parabolic equations', *BIT* **27**(2), 216–234.

Y. Maday and O. Mula (2020), 'An adaptive parareal algorithm', *J. Comput. Appl. Math.* **377**, 112915.

Y. Maday and E. M. Rønquist (2008), 'Parallelization in time through tensor-product space–time solvers', *C. R. Math. Acad. Sci. Paris* **346**(1), 113–118.

V. Martin (2009), 'Schwarz waveform relaxation algorithms for the linear viscous equatorial shallow water equations', *SIAM Journal on Scientific Computing* **31**(5), 3595–3625.

T. P. Mathew, M. Sarkis and C. E. Schaerer (2010), 'Analysis of block parareal preconditioners for parabolic optimal control problems', *SIAM J. Sci. Comput.* **32**(3), 1180–1200.

E. McDonald, J. Pestana and A. Wathen (2018), 'Preconditioning and iterative solution of all-at-once systems for evolutionary partial differential equations', *SIAM J. Sci. Comput.* **40**(2), A1012–A1033.

M. Merkel, I. Niyonzima and S. Schöps (2017), 'ParaExp using leapfrog as integrator for high-frequency electromagnetic simulations', *Radio Science* **52**(12), 1558–1569.

G. A. Meurant (1991), Numerical experiments with a domain decomposition method for parabolic problems on parallel computers, in *Proceedings of the Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SlAM, Philadelphia, PA, pp. 394–408.

M. Minion (2011), 'A hybrid parareal spectral deferred corrections method', *Communications in Applied Mathematics and Computational Science* **5**(2), 265–301.

M. L. Minion (2010), 'A hybrid parareal spectral deferred corrections method', *Commun. Appl. Math. Comput. Sci.* **5**(2), 265–301.

M. L. Minion, R. Speck, M. Bolten, M. Emmett and D. Ruprecht (2015*a*), 'Interweaving PFASST and parallel multigrid', *SIAM journal on scientific computing* **37**(5), S244–S263.

M. L. Minion, R. Speck, M. Bolten, M. Emmett and D. Ruprecht (2015*b*), 'Interweaving PFASST and parallel multigrid', *SIAM J. Sci. Comput.* **37**(5), S244–S263.

W. L. Miranker and W. Liniger (1967), 'Parallel methods for the numerical integration of ordinary differential equations', *Math. Comp.* **91**, 303–320.

C. Moler and C. Van Loan (2003), 'Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later', *SIAM Rev.* **45**(1), 3–49.

M. Neumüller and I. Smears (2019), 'Time-parallel iterative solvers for parabolic evolution equations', *SIAM Journal on Scientific Computing* **41**(1), C28–C51.

O. Nevanlinna (1989), 'Remarks on Picard-Lindelöf iteration: Part I', *BIT* **29**(2), 328–346.

M. K. Ng (2004), *Iterative Methods for Toeplitz Systems*, Oxford Academic.

J. Nievergelt (1964), 'Parallel methods for integrating ordinary differential equations', *Comm. ACM* **7**, 731–733.

B. W. Ong and J. B. Schröder (2020), 'Applications of time parallelization', *Computing and Visualization in Science* **23**, 1–15.

J. M. Ortega and W. C. Rheinboldt (2000), *Iterative Solution of Nonlinear Equations in Several Variables*, SIAM, Philadelphia, PA, USA.

J. W. Pearson, M. Stoll and A. J. Wathen (2012), 'Regularization-robust preconditioners for time-dependent pde-constrained optimization problems', *SIAM J. Matrix Anal. Appl.* **33**(4), 1126–1152.

P. Saha, J. Stadel and S. Tremaine (1997), 'A parallel integration method for solar system dynamics', *The Astronomical Journal* **114**(1), 409–415.

M. Schreiber, P. S. Peixoto, T. Haut and B. Wingate (2018), 'Beyond spatial scalability limitations with a massively parallel method for linear oscillatory problems', *The International Journal of High Performance Computing Applications* **32**(6), 913–933.

H. Schwarz (1870), 'Über einen Grenzübergang durch alternierendes Verfahren', *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich* **15**, 272–286.

J. Simoens and S. Vandewalle (2000), 'Waveform relaxation with fast direct methods as preconditioner', *SIAM J. Sci. Comput.* **21**(5), 1755–1773.

R. Speck, D. Ruprecht, M. Emmett, M. Bolten and R. Krause (2014), A space-time parallel solver for the three-dimensional heat equation, in *Parallel Computing: Accelerating Computational Science and Engineering (CSE)*, Vol. 25, IOS Press, pp. 263–272.

R. Speck, D. Ruprecht, R. Krause, M. Emmett, M. Minion, M. Winkel and P. Gibbon (2012), A massively space-time parallel N-body solver, in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, IEEE Computer Society Press, pp. 1–11.

G. Strang (1986), 'A proposal for Toeplitz matrix calculations', *Stud. Appl. Math.* **74**(2), 171–176.

S. Thery, C. Pelletier, F. Lemarié and E. Blayo (2022), 'Analysis of Schwarz waveform relaxation for the coupled Ekman boundary layer problem with continuously variable coefficients', *Numerical Algorithms* pp. 1–37.

J. Van Lent and S. Vandewalle (2002), 'Multigrid waveform relaxation for anisotropic partial differential equations', *Numerical Algorithms* **31**, 361–380.

F. Van Loan, C and N. Pitsianis (1993), 'Approximations with Kronecker products', *Linear Algebra for Large Scale and Real-Time Apprications* pp. 293–314.

S. Vandewalle and E. Van de Velde (1994), 'Space-time concurrent multigrid waveform relaxation', *Annals of Numer. Math* **1**, 347–363.

D. E. Womble (1990), 'A time-stepping algorithm for parallel computers', *SIAM J. Sci. Stat. Comput.* **11**(5), 824–837.

P. Worley (1991), Parallelizing across time when solving time-dependent partial differential equations, in *Proc. 5th SIAM Conf. on Parallel Processing for Scientific Computing, D. Sorensen, ed., SIAM.*

S.-L. Wu (2015), 'Convergence analysis of some second-order parareal algorithms', *IMA J. Numer. Anal.* **35**(3), 1315–1341.

S.-L. Wu (2017), 'Optimized overlapping Schwarz waveform relaxation for a class of time-fractional diffusion problems', *Journal of Scientific Computing* **72**, 842–862.

S.-L. Wu (2018), 'Toward parallel coarse grid correction for the parareal algorithm', *SIAM J. Sci. Comput.* **40**(3), A1446–A1472.

S.-L. Wu and M. D. Al-Khaleel (2014), 'Semi-discrete Schwarz waveform relaxation algorithms for reaction diffusion equations', *BIT* **54**, 831–866.

S.-L. Wu and J. Liu (2020), 'A parallel-in-time block-circulant preconditioner for optimal control of wave equations', *SIAM J. Sci. Comput.* **42**(3), A1510–A1540.

S.-L. Wu and Y. Xu (2017), 'Convergence analysis of Schwarz waveform relaxation with convolution transmission conditions', *SIAM J. Sci. Comput.* **39**(3), A890–A921.

S.-L. Wu and T. Zhou (2015), 'Convergence analysis for three parareal solvers', *SIAM J. Sci. Comput.* **37**(2), A970–A992.

S.-L. Wu and T. Zhou (2019), 'Acceleration of the two-level MGRIT algorithm via the diagonalization technique', *SIAM J. Sci. Comput.* **41**(5), A3421–A3448.

S.-L. Wu and T. Zhou (2021*a*), 'Parallel implementation for the two-stage SDIRK methods via diagonalization', *J. Comput. Phys.* **428**, 110076.

S.-L. Wu and T. Zhou (2024), 'Convergence analysis of the parareal algorithm with non-uniform fine time grid', *SIAM J. Numer. Anal., to appear in 2024.*

S.-L. Wu, C. M. Huang and T. Z. Huang (2012), 'Convergence analysis of the overlapping Schwarz waveform relaxation algorithm for reaction-diffusion equations with time delay', *IMA J. Numer. Anal.* **32**(2), 632–671.

S.-L. Wu, Z. Wang and T. Zhou (2023), 'PinT preconditioner for forward-backward evolutionary equations', *SIAM J. Matrix Anal. Appl.* **44**(4), 1771–1798.

S.-L. Wu, Z. H. Yang and T. Zhou (2024), 'Mixed precision iterative paradiag algorithm', *submitted.*

S.-L. Wu, T. Zhou and Z. Zhou (2022), 'A uniform spectral analysis for a preconditioned all-at-once system from first-order and second-order evolutionary problems', *SIAM J. Matrix Anal. Appl.* **43**(3), 1331–1353.

S. N. Wu and Z. Zhou (2021*b*), 'A parallel-in-time algorithm for high-order BDF methods for diffusion and subdiffusion equations', *SIAM J. Sci. Comput.* **43**(6), A3627–A3656.

J. Yang, Z. M. Yuan and Z. Zhou (2023), 'Robust convergence of parareal algorithms with arbitrarily high-order fine propagators', *CSIAM Trans. Appl. Math.* **4**(3), 566–591.