Computing and Visualization in Science (2018) 19:45–57
https://doi.org/10.1007/s00791-018-0294-1

SPECIAL ISSUE PARALLEL-IN-TIME METHODS

A numerical study of a semi-Lagrangian Parareal method applied to the viscous Burgers equation

A. Schmitt[1] · M. Schreiber[2] · P. Peixoto[3] · M. Schäfer[1]

Received: 4 May 2017 / Accepted: 8 November 2017 / Published online: 6 June 2018

Abstract
This work focuses on the Parareal parallel-in-time method and its application to the viscous Burgers equation. A crucial component of Parareal is the coarse time stepping scheme, which strongly impacts the convergence of the parallel-in-time method. Three choices of coarse time stepping schemes are investigated in this work: explicit Runge–Kutta, implicit–explicit Runge–Kutta, and implicit Runge–Kutta with semi-Lagrangian advection. Manufactured solutions are used to conduct studies, which provide insight into the viability of each considered time stepping method for the coarse time step of Parareal. One of our main findings is the advantageous convergence behavior of the semi-Lagrangian scheme for advective flows.

Keywords Parareal · Burgers' equation · Semi-Lagrangian · Runge–Kutta · Parallel-in-time

1 Introduction

Keeping the time-to-solution for simulations below a given wall-clock time plays a crucial role for a variety of applications such as wave propagation simulations in the area of medical science, weather predictions to make forecasts as accurate as possible [41], and early-warning systems for Tsunamis to improve evacuation plans [6]. Over recent decades one of the main factors to achieve improved results of such simulations was by an increase in spatial resolution (e.g. [43]). However, for time evolving problems increasing the spatial resolution also usually requires decreasing the time step size, which again leads to an increase in workload and this workload has to be finished within the same time frame. A steady increase in computer clock speeds conveniently compensated for this additional workload. However, this increase has stagnated since about 2004.[1]

Today, performance gains are no longer delivered for free through increasing clock speed [38], but via additional parallelism at the instruction and core levels. However, this yields increased communication and synchronization overheads and makes performance gains for simulations which already stagnated in their scalability very challenging. Performance improvements are in particular required for simulations which have to be finished within a particular time frame, and these simulations are the focus of this work.

Various kinds of approaches are currently under investigation to overcome such limitations. These range from new hardware (networking, new instruction sets, broader vector registers) to the software level (new algorithms for latency hiding, optimized network stacks, parallel-in-time methods). In this work, we concentrate on the software side with parallel-in-time methods and its mathematical realization to exploit resources beyond spatial scalability.

Parallel-in-time methods have gained a growing interest over recent decades with a rich history [17]. A widely used

Schmitt: The work of this author is supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at Technische Universität Darmstadt
Peixoto: Acknowledges the Sao Paulo Research Foundation (FAPESP) under the Grant Number 2016/18445-7 and the National Science and Technology Development Council (CNPq) under Grant Number 441328/2014-8.

✉ A. Schmitt
aschmitt@gsc.tu-darmstadt.de

[1] Institute of Numerical Methods in Mechanical Engineering and Graduate School of Computational Engineering, TU Darmstadt, Dolivostr. 15, 64293 Darmstadt, Germany

[2] University of Exeter, Harrison Building, North Park Road, Exeter EX4 4QF, UK

[3] Instituto de Matemática e Estatística, Universidade de São Paulo, Rua do Matão, 1010, São Paulo, Brazil

[1] See http://www.top500.org statistics.

and studied algorithm of this class is the Parareal algorithm [27], which will be the algorithm of choice in this work.

A closely related approach is the PITA algorithm [14], which adopts a slightly different correction scheme. Expanding spectral deferred corrections methods [12] in a time-parallel fashion leads to the PFASST algorithm [13]. Another strategy is considering a pipeline parallel deferred correction framework, which leads to the RIDC scheme [9]. Also, multigrid-type fashioned solvers have been investigated in the context of parallel-in-time. The space-time multigrid by [22] is a multigrid method which is applied to the whole space-time domain. Applying a multigrid reduction to the time dimension lead to the MGRIT algorithm [15]. Time parallelism was introduced to the multigrid waveform relaxation [28] by using the partition method [40] and later on by replacing the partition method with cyclic reduction [23].

Parallel-in-time algorithms have in common that they are less efficient regarding the improved time-to-solution when applied to realistic and dominantly hyperbolic problems. These problems, mainly related to stability, have already been investigated by multiple authors [14,16,32,35,37].

Various research shows that the convergence of parallel-in-time algorithms is highly dependent on the coarse time stepper used within the algorithms [3,19].

In this work, we study the dependency of different coarse time stepping schemes with respect to the efficiency of a parallel-in-time algorithm applied to the viscous Burgers equation, described in Sect. 2. Burgers' equation is a simplified fluid model frequently used in the development stages of solvers for the Navier–Stokes equations, being particularly relevant for flows with high Reynolds numbers (small viscosities). It is simple enough to allow more detailed theoretical and experimental analyses, but at the same time sufficiently sophisticated in terms of representing one part of the complex non-linear phenomena of fluid dynamics.

For high Reynolds numbers, Burgers' equation is dominated by advection (dominantly hyperbolic), which creates challenges for parallel-in-time schemes. To overcome stability and convergence issues while ensuring sufficient accuracy with large time steps, we investigate the use of semi-Lagrangian schemes for the non-linear advection part of the problem (see Sect. 4.4). Semi-Lagrangian schemes are widely used in the geophysical fluid dynamics community [36], mainly due their property of allowing time step sizes beyond Eulerian CFL limitations. Apart from a couple of suggestions of its benefits to parallel-in-time frameworks [10,31], it appears to be a potentially promising scheme to be investigated.

Our parallel-in-time algorithm of choice is the Parareal algorithm, which is explained in Sect. 3. We combine the semi-Lagrangian scheme for the coarse time-integrator of a Parareal algorithm and compare this approach to standard time integrators (explicit and implicit–explicit Runge–Kutta methods), as described in Sect. 4. Additionally, the application of the semi-Lagrangian method requires modifications of the communication patterns in the Parareal algorithm as depicted in Sect. 5. The results of all numerical experiments are summarized in Sect. 6. Finally, a conclusion is drawn in Sect. 7.

2 Burgers' equation

The Navier–Stokes equations are the fundamental equations for many problems in computational fluid dynamics (CFD) [42]. Here, in particular flow problems with high Reynolds numbers (high ratio between advection and diffusion) lead to a dominantly hyperbolic problem. This poses particular challenges for Parareal and leads to a decrease in the Parareal convergence rate [35]. In this work, we put the focus on the part of the Navier–Stokes equations with high Reynolds numbers which can be expressed by the viscous Burgers equation with small viscosities. This allows us to put the focus on this particular effect for convergence studies with Parareal.

The viscous Burgers equation was introduced by [5] and extensively studied by Burgers (e.g. [8]). The close relation of these equations can be seen easily by starting from the momentum conservation equation of the incompressible Navier–Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{F}, \tag{1}$$

where $\mathbf{u}$ denotes the velocity, ρ the density, p the pressure, ν the kinematic viscosity, and $\mathbf{F}$ the external forces. By dropping the pressure and external forces, we avoid coping with the mass conservation equation. This leads to the viscous Burgers equation

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = \nu \nabla^2 \mathbf{u}. \tag{2}$$

For a better presentation of the results, we use only one dimension in space leading to

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} + Q, \tag{3}$$

where Q is a forcing term. Analytic solutions of Burgers' equation exist for some special cases [42], but, in this work, we apply manufactured solutions using the source term Q.

3 Parareal

The Parareal algorithm was first presented by [27] and has gained steadily increasing interest. This section provides a short introduction to the idea and the algorithm itself.

We are interested in the numerical solution of an autonomous system. For the sake of simplicity, we consider this to be an ordinary differential equations (ODE),

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = f(\mathbf{u}), \qquad \text{with } \mathbf{u}(t_0) = \mathbf{u}_0, \tag{4}$$

where $f(\mathbf{u})$ is a Lipschitz continuous function.

Using the Parareal algorithm for this ODE requires two different time stepping methods with different time step sizes: a fine time step Δt and a coarse time step ΔT. The fine time stepping integrator is denoted by the functional $\mathcal{F}(\mathbf{u}_n, t_n, t_{n+1})$, and it uses many small time steps Δt within a period of a large time step ΔT between t_n and $t_{n+1} = t_n + \Delta T$. Usually, an accurate state-of-the-art time integrator is adopted as the fine time stepping scheme. These are naturally sequential in time resulting in limitations with respect to the time-to-solution.

The idea of Parareal is to split the time domain $[t_0, T]$ over which Eq. (4) is to be solved into multiple time slices, each of size ΔT, and to compute the numerical solution on those slices in parallel. In order to do this, an initial condition is required for each time slice $[t_n, t_{n+1}]$, and the parallel solution will only give adequate results if this initial condition agrees with the final condition obtained in the previous time slice. An estimator is used to predict these initial values for each slice with a coarse propagator $\mathcal{C}(\mathbf{u}_n, t_n, t_{n+1})$. This coarse propagator has to be able to cope with large time steps (of size ΔT) and should take significantly less computation time than the fine time stepping method.

Accurate results are obtained considering an iterative scheme, using the coarse integrator as prediction and the fine integrator as correction, where the fine integrator can run in parallel. Starting with the initial conditions $\mathbf{u}_0^0$ the resulting iteration can be written as

$$\mathbf{u}_{n+1}^{k+1} = \mathcal{C}(\mathbf{u}_n^{k+1}) + \mathcal{F}(\mathbf{u}_n^k) - \mathcal{C}(\mathbf{u}_n^k) \tag{5}$$

for time step t_n to t_{n+1}, where the superscripts k refer to the iterations of the scheme [2]. The pseudocode representation of the scheme is given in Algorithm 1.

Based on the accuracy and smoothness of the fine and coarse integrators, it is possible to estimate analytical upper bounds of the error achieved after a certain number of iterations [3,18].

Clearly, if the number of iterations is equal to or exceeds the number of time slices, then no acceleration is gained. Additional overhead due to the execution of the coarse propagator and communication would even lead to an increase in wall-clock time compared to a non-Parareal execution. Therefore, two requirements have to be met for this method to gain a speedup:

1. The coarse solver $\mathcal{C}$ needs to take substantially less wall-clock time than the fine solver $\mathcal{F}$ per time slice, by adopting either a reduced order method (with reduced cost) and/or very large time step sizes.
2. The Parareal algorithm needs to take far fewer iterations than the number of time slices, which are computed in parallel.

These two requirements are typically in contradiction to each other. This poses the main challenge of finding an adequate coarse propagator. The challenge is highly problem specific, and ODEs of different nature can require different schemes.

$\mathbf{U}_0^0 \leftarrow \tilde{\mathbf{U}}_0^0 \leftarrow \mathbf{u}_0$
for $n = 1$ **to** N **do**
  $\tilde{\mathbf{U}}_n^0 \leftarrow \mathcal{C}\left(\mathbf{U}_{n-1}^0\right)$
  $\mathbf{U}_n^0 \leftarrow \tilde{\mathbf{U}}_n^0$
end
while $|\mathbf{U}_n^k - \mathbf{U}_n^{k-1}| > \epsilon \; \exists n$ **do**
  $\mathbf{U}_0^k \leftarrow \mathbf{u}_0$
  for $n = 1$ **to** N **do**
    $\hat{\mathbf{U}}_n^{k-1} \leftarrow \mathcal{F}\left(\mathbf{U}_{n-1}^{k-1}\right)$ `// Parallel step`
  end
  for $n = 1$ **to** N **do**
    $\tilde{\mathbf{U}}_n^k \leftarrow \mathcal{C}\left(\mathbf{U}_{n-1}^k\right)$ `// Predict`
    $\mathbf{U}_n^k \leftarrow \tilde{\mathbf{U}}_n^k + \hat{\mathbf{U}}_n^{k-1} - \tilde{\mathbf{U}}_n^{k-1}$ `// Correct`
  end
end

Algorithm 1: Pseudo code of the Parareal algorithm. $\mathcal{C}$ and $\mathcal{F}$ denote the coarse and fine solver respectively. The initial condition is $\mathbf{u}_0$, $\mathbf{U}_n^k$ denotes the solution at iteration k and time point t_n. The solutions of $\mathcal{C}$ and $\mathcal{F}$ are $\tilde{\mathbf{U}}_n^k$ and $\hat{\mathbf{U}}_n^k$ respectively.

4 Time stepping methods

In this section, we describe the numerical schemes used in the numerical studies for the Parareal method. All the methods have in common at least a 1st order scheme (in time and space) for the diffusive part and the forcing term, and a 2nd order scheme (in time and space) for the advective term. A special focus lies on the semi-Lagrangian formulation. Table 1 summarizes the schemes employed.

Table 1 Summary with description and accuracy of the schemes used in this work

Scheme	Spatial approximations		Time stepping		Reference equations
	Advection	Diffusion	Advection	Diffusion	
Explicit	Spec (expl)	Spec (expl)	RK2 $O(\Delta t^2)$ (expl)	Euler $O(\Delta t)$ (expl)	(6), (7)
IMEX	Spec (expl)	Spec (Helm)	RK2 $O(\Delta t^2)$ (expl)	Euler $O(\Delta t)$ (impl)	(11), (12), (13)
SL	Interp $O(\Delta x^2)$	Spec (Helm)	SETTLS $O(\Delta t^2)$ (expl)	Euler $O(\Delta t)$ (impl)	(22), (23), (24)

"Spec" refers to spectral differentiation, "expl" and "impl" refer to explicit and implicit schemes, respectively. "Helm" refers to the requirement of solving a definite Helmholtz equation. "Interp" refers to interpolation schemes, which are of order 2 or higher

4.1 Spatial discretizations

The primary aim of this study is to investigate effects of the time-integration scheme, so we will adopt accurate spectral methods for the spatial discretizations when possible. We use the periodic Fourier basis to represent the solution in spectral space (see e.g. [11]).

Therefore, all linear operators are directly applied element-wise in spectral space with spectral accuracy. For functions exactly represented in the spectral space with a given number of modes, the error of the linear operator is of the same order of magnitude as the round-off errors.

The non-linear terms could, in principle, be calculated in spectral space with a convolution of all spectral series, which would be of quadratic complexity. To avoid this complexity, a pseudo-spectral approach is usually adopted [4,20], where the non-linearities are computed node-wise in physical space. This can lead to spurious modes, and a standard anti-aliasing technique is applied to overcome this, in which a higher resolution in physical space followed by a truncation of modes in spectral space is used after each non-linear operation (see e.g. [30]).

Using a Fourier spectral basis also provides advantages for the solution of linear systems, which usually correspond to a definite Helmholtz equation [see Eqs. (13), (24)]. This can be solved accurately and efficiently with an element-wise vector-vector multiplication in spectral space (see e.g.[33, 39]).

4.2 Explicit Runge–Kutta

The simplest method we use for our study is a two stage explicit Runge–Kutta (RK) method [26]. To reach the aforementioned orders, we use the midpoint rule for the advective part of Eq. (3) and a 2-stage explicit Euler method for the diffusive part and the forcing term. This results in

$$u_1 = u^n + \Delta t \left(\nu \frac{\partial^2 u^n}{\partial x^2} + Q^n \right) - \frac{\Delta t}{2} u^n \frac{\partial u^n}{\partial x}, \tag{6}$$

$$u^{n+1} = u^n + \Delta t \left(\nu \frac{\partial^2 u_1}{\partial x^2} + Q_1 - u_1 \frac{\partial u_1}{\partial x} \right), \tag{7}$$

where u_1 denotes the intermediate solution of stage one, and Q_1 is evaluated at $t^{n+\frac{1}{2}}$. This scheme is 2nd order accurate in time for the non-linear advection term and 1st order for the linear terms.

4.3 Implicit–explicit Runge–Kutta

In this section, we recap the idea behind implicit–explicit (IMEX) Runge–Kutta methods. For further information the reader is referred to e.g. [1,24]. The IMEX methods are based on the idea that terms with different stability restrictions are treated accordingly. The stiff (linear) terms are treated implicitly, and the non-stiff (non-linear) terms are treated explicitly. Following the algorithm of [1], we use the explicit midpoint rule for the non-stiff terms and the implicit Euler for the stiff terms. Given the equation

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = f(\mathbf{u}) + g(\mathbf{u}), \quad \text{with } \mathbf{u}(t_0) = \mathbf{u}_0, \tag{8}$$

where f is a linear stiff term, and g is a non-linear term, discretization in time yields

$$\mathbf{u}_1 = \mathbf{u}^n + \Delta t f(\mathbf{u}_1) + \frac{\Delta t}{2} g(\mathbf{u}^n) \tag{9}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t (f(\mathbf{u}_1) + g(\mathbf{u}_1)), \tag{10}$$

where we integrate from t_n to t_{n+1} with a time step of Δt. The implicit treatment of the stiff term in Eq. (9) ensures that it imposes less stability restrictions on the time step size for the stiff term; however, it introduces larger phase and amplitude errors for larger time steps (see [11]).

Applying the scheme to the viscous Burgers equation [see Eq. (3)] and treating the forcing term implicitly yields

$$u_1 = u^n + \Delta t \left(\nu \frac{\partial^2 u_1}{\partial x^2} + Q_1 \right) - \frac{\Delta t}{2} u^n \frac{\partial u^n}{\partial x}, \tag{11}$$

$$u^{n+1} = u^n + \Delta t \left(\nu \frac{\partial^2 u_1}{\partial x^2} + Q_1 - u_1 \frac{\partial u_1}{\partial x} \right), \tag{12}$$

where Q_1 is evaluated at $t^{n+\frac{1}{2}}$.

Using the implicit Euler step in this two stage scheme leads to an explicit handling of the diffusion in the second stage. This yields conditional stability for diffusion dominated problems. A von Neumann analysis of this scheme applied to the linearized equation shows a larger stability region than the fully explicit scheme.

Equation (11) can be written as a definite Helmholtz problem of the following form

$$\left(\mathsf{I} - \Delta t \nu \frac{\partial^2}{\partial x^2}\right) u_1 = u^n - \frac{\Delta t}{2} u^n \frac{\partial u^n}{\partial x} + \Delta t Q_1, \tag{13}$$

where I is the identity operator.

4.4 Semi-Lagrangian formulation

Semi-Lagrangian schemes are frequently and successfully used in geophysical fluid dynamics as a way to obtain an increase in the time step size for advection dominated problems [36], which inspired this investigation with Parareal. In this section, we review the semi-Lagrangian formulation used for this work. For a comprehensive introduction to semi-Lagrangian schemes we refer the reader to [7,11].

The basic idea behind the semi-Lagrangian method is to use a Lagrangian formulation of the equation with respect to a fixed Eulerian grid. In the Eulerian framework, an observer at a fixed position observes an entity moving past the observer. The Lagrangian formulation implies an observer which moves with the observed entity, which means that the computational grid also moves through space over time. With a semi-Lagrangian framework, the Lagrangian framework is used for the particle movement, however, the simulation data is stored on an Eulerian grid. Values for the particle grids are then interpolated on the Eulerian grid.

Since with this scheme we resolve the Lagrangian trajectory, the numerical domain of dependence includes the physical domain of dependence, which ensures the fulfillment of a necessary condition for unconditional stability (independent of the time step size) with respect to the advection term. Therefore, using a stable trajectory calculation scheme, the time step size will not be restricted by the CFL stability condition (which limits both RK and IMEX schemes), but will only be restricted to accuracy conditions.

Burgers' equation can be written within a Lagrangian framework using the concept of total or material derivatives,

$$\frac{\mathrm{d}u(t, x(t))}{\mathrm{d}t} = \frac{\partial u}{\partial t} + \dot{x}(t)\left(\frac{\partial u}{\partial x}\right) = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x}, \tag{14}$$

where $\frac{\mathrm{d}}{\mathrm{d}t}$ denotes the total derivative and the velocity $\dot{x}(t) = u(t, x(t))$. Therefore, the Lagrangian formulation of Burgers' equation reads

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \nu \frac{\partial^2 u}{\partial x^2} + Q. \tag{15}$$

For a backward Euler time integration we get

$$\frac{u(t_{n+1}, x(t_{n+1})) - u(t_n, x(t_n))}{\Delta t} = \nu \frac{\partial^2 u}{\partial x^2}(t_{n+1}, x(t_{n+1})) + Q(t_{n+1}, x(t_{n+1})). \tag{16}$$

The velocity for time step t_{n+1} is stored at grid points defined as $x(t_{n+1})$. The set of grid points defined at time t_{n+1} are commonly denoted as the *arrival* points x_a. The key point now is to estimate $x(t_{n+1/2})$ and $x(t_n)$, which are called *midpoints* x_m and *departure* points x_d, respectively. These points can be obtained by solving the Lagrangian trajectory ODE

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = u(t, x(t)), \tag{17}$$

which when integrated within (t_n, t_{n+1}) results in

$$x_a - x_d = \int_{t_n}^{t_{n+1}} u(t, x(t))\, \mathrm{d}t. \tag{18}$$

Discretizing the integral with the midpoint rule yields

$$x_a - x_d = u(t_{n+1/2}, x_m)\Delta t. \tag{19}$$

Next, the required velocity at a future intermediate step $u(t_{n+1/2}, x_m)$ can be computed by extrapolation. Choosing this extrapolation carefully is important in order to avoid possible instabilities of the scheme (see [11]). We choose the stable extrapolation two-time-level scheme (SETTLS) proposed by [21], which calculates the velocity using information from a previous time step t_{n-1} as

$$u(t_{n+1/2}, x_m) \approx \frac{\Delta t}{2}(2u(t_n, x_d) - u(t_{n-1}, x_d) + u(t_n, x_a)). \tag{20}$$

Joining Eqs. (20) and (19) one obtains a non-linear implicit equation for the unknown x_d, which can be solved with an iterative scheme (index k) as

$$x_d^{k+1} = x_a - \frac{\Delta t}{2}(2u(t_n, x_d^k) - u(t_{n-1}, x_d^k) + u(t_n, x_a)) \tag{21}$$

with initial guess $x_d^0 = x_a$.

We adopt a stopping criterion of maximum absolute distance between departure points obtained from two iterations of $\epsilon = 10^{-8}$ with a maximum of 10 iterations. Generally,

the maximum number of iterations is not reached, since only a few iterations are typically enough to obtain the departure points very accurately. Within the iterative procedure, it is required to calculate the velocity at non-grid points. These values are obtained through 2nd order bilinear interpolation with respect to the nearest grid points [36].

Denoting $(\cdot)_*$ as the value of a field interpolated to its respective departure points we can write the iteration as

$$x_d^{k+1} = x_a - \frac{\Delta t}{2} u(t_n) - \frac{\Delta t}{2} (2u(t_n) - u(t_{n-1}))_*. \tag{22}$$

Using the same notation, but now ignoring the subscript a, we can apply the semi-Lagrangian formulation to the discrete Burgers equation (Eq. 16) resulting in

$$u^{n+1} = u_*^n + \Delta t \nu \nabla^2 u^{n+1} + \Delta t Q^{n+1}. \tag{23}$$

Reformulating Eq. (23), and ignoring the forcing, leaves us with

$$\left(\mathrm{I} - \nu \Delta t \nabla^2\right) u^{n+1} = u_*^n, \tag{24}$$

which is again a definite Helmholtz problem [like Eq. (13)]. This notation also shows that it is possible to solve the semi-Lagrangian formulation in two steps. First, the departure points need to be estimated, and the velocity at the current time step must be interpolated to these points. Next, these interpolated values are used in the right-hand-side of the Helmholtz problem, which is solved with the aid of an accurate and efficient spectral solver.

All interpolations at the departure points are done with 4th order accuracy (bicubic interpolation), which combined with the bilinear interpolation of the velocities ensures an overall 2nd order accurate scheme with respect to advection (see [29]).

5 Communication patterns in Parareal with SL

The 2nd order semi-Lagrangian scheme, see Sect. 4.4, poses either additional requirements on the communication of Parareal with additional interfaces required, or an increase in memory used. Since this plays an important role for future parallel implementations and efficiency, we discuss this in more detail.

Expanding the standard Parareal algorithm (Algorithm 1) with the two steps used for solving the semi-Lagrangian scheme leads to the SL-Parareal Algorithm 2, where all changes are underlined. Here, we see the additional velocity $\mathbf{U}_*^k$ at the departure points necessary for the semi-Lagrangian formulation. Depending on whether Parareal communication

$\mathbf{U}_0^0 \leftarrow \underline{\mathbf{U}_{-1}^0} \leftarrow \tilde{\mathbf{U}}_0^0 \leftarrow \mathbf{u}_0$
for $n = 1$ **to** N **do**
  $\underline{\mathbf{U}_*^0 = \mathcal{SL}\left(\mathbf{U}_{n-1}^0, \mathbf{U}_{n-2}^0\right)}$
  $\tilde{\mathbf{U}}_n^0 \leftarrow \mathcal{C}\left(\underline{\mathbf{U}_*^0}\right)$
  $\mathbf{U}_n^0 \leftarrow \tilde{\mathbf{U}}_n^0$
end
while $|\mathbf{U}_n^k - \mathbf{U}_n^{k-1}| > \epsilon \; \exists n$ **do**
  $\mathbf{U}_0^k \leftarrow \underline{\mathbf{U}_{-1}^k} \leftarrow \mathbf{u}_0$
  for $n = 1$ **to** N **do**
    $\hat{\mathbf{U}}_n^{k-1} \leftarrow \mathcal{F}\left(\mathbf{U}_{n-1}^{k-1}\right)$ `// Parallel step`
  end
  for $n = 1$ **to** N **do**
    $\underline{\mathbf{U}_*^k = \mathcal{SL}\left(\mathbf{U}_{n-1}^k, \mathbf{U}_{n-2}^k\right)}$
    $\tilde{\mathbf{U}}_n^k \leftarrow \mathcal{C}\left(\underline{\mathbf{U}_*^k}\right)$ `// Predict`
    $\mathbf{U}_n^k \leftarrow \tilde{\mathbf{U}}_n^k + \hat{\mathbf{U}}_n^{k-1} - \tilde{\mathbf{U}}_n^{k-1}$ `// Correct`
  end
end

Algorithm 2: Pseudo code of the SL-Parareal algorithm, where the semi-Lagrangian formulation with SETTLS is used as the coarse solver. Parts added to the standard algorithm are underlined. $\mathbf{U}$ denotes the solution of the algorithm, $\tilde{\mathbf{U}}$ denotes the solution of the coarse solver and $\hat{\mathbf{U}}$ denotes the solution of the fine solver. The superscript k stands for the Parareal iteration, the index n for the time step and the index $*$ for the evaluation at the departure point.

is realized on a shared or distributed memory system, different ways exist to handle this additional velocity at the departure points.

1. In a shared memory or partitioned global address space (PGAS) Parareal environment the velocity $\mathbf{U}_*^k$ can be stored as a new variable and can be made directly accessible with a pointer.
2. In a distributed memory Parareal environment additional communication is necessary, as described below.

Even though the code used for this work is only serial, the realization is done in a fashion which resembles the communication on distributed memory systems. Here, we target the investigation of parallel-in-time algorithms for large systems.

In our implementation each time slice has its independent memory areas. For the standard Parareal algorithm, we send the result of time slice $T_{n-1} = [t_{n-2}, t_{n-1}]$ as initial condition to time slice $T_n = [t_{n-1}, t_n]$. The SL-Parareal scheme requires additionally sending the result of time slice $T_{n-2} = [t_{n-3}, t_{n-2}]$. Due to overwriting $\mathbf{U}_{n-1}^k$ with $\mathbf{U}_*^k$ in time slice T_{n-1} to reduce memory consumption the data is received from T_{n-2}.

A sketch for the new data dependencies, which shows the additionally required communication, is shown in Fig. 1. Here, the communication during the serial prediction-correction

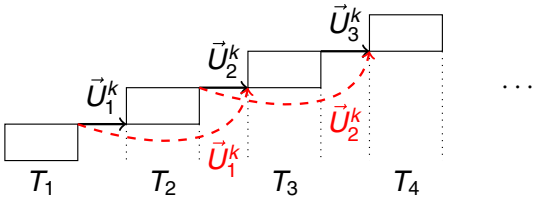

Fig. 1 Sketch of the communication pattern for the prediction and correction step (as boxes) of the distributed memory Parareal algorithm in iterations k with the semi-Lagrangian formulation applied to the coarse solver. The time slices are denoted by $T_n = [t_{n-1}, t_n]$. Arrows show the communication of the velocity $\mathbf{U}_n^k$ (solid: standard Parareal; dashed: additional for SL-Parareal)

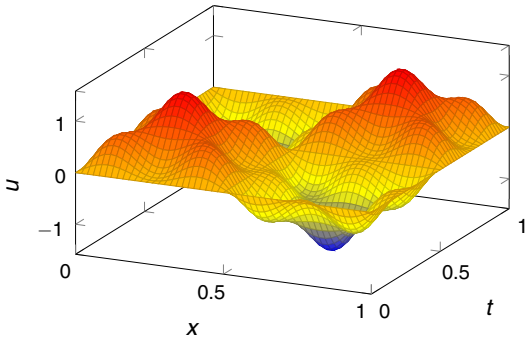

Fig. 2 Analytical solution u of benchmark B1 over the computational domain $(x, t) \in [0, 1]^2$

step is indicated by the arrows. The dashed red arrows visualize the additional communication necessary for the 2nd order SL-Parareal scheme used in this work.

6 Numerical experiments

We conducted various numerical experiments which exploit different challenges for the Burgers equations. First, we describe our benchmarking test cases followed by a stability study of serial time stepping for the numerical methods presented in Sect. 4. This is followed by an examination of these different methods in combination with the Parareal algorithm.

For the sake of reproducibility, we have published the source code for all aforementioned methods in the repository [34] of the SWEET development.

6.1 Benchmarks

The test cases are based on [25] and mimic turbulent fluid flows at high Reynolds numbers modeled through multiple length scales.

Both benchmarks are based on the method of manufactured solutions. This means we define a solution $\overline{u}(x, t)$ to calculate the source term Q of Eq. (3), followed by using the calculated source term and the initial condition $\overline{u}(x, t_0)$ in the solver. The analytical solution is then used for error comparisons. The benchmarks have in common that they are carried out on the space-time domain $(x, t) \in [0, 1]^2$.

We next describe parameters, which are specific to the Parareal studies. The maximum absolute difference (in space, for all times) between two Parareal iterations is used to evaluate the convergence of the Parareal algorithm, and a tolerance of $tol = 10^{-8}$ is used as a stopping criterion. The Parareal studies are executed with a time discretization of $\Delta T = 10^{-2}$ and $\Delta t = 10^{-6}$ for the coarse and fine solver, respectively. Such large ratios $\Delta T / \Delta t$ are typically more challenging. Each coarse time step represents one time slice leading to a total number of $N_T = 100$ time slices in the time interval $[0, 1]$.

For Parareal we use IMEX as the fine solver and one of the three time stepping methods (RK, IMEX, SL) as the coarse solver. We refer to the different combinations by their coarse solver. IMEX was chosen as the fine integrator for all experiments as it provides 2nd order accuracy in time at smaller cost compared to the SL scheme, and the stability constraints on Δt are smaller compared to RK.

6.1.1 B1: Sinusoidal waves

For our first benchmark (B1) the defined solution

$$\overline{u}(t, x) = \sin(2\pi x)\sin(2\pi t) + \frac{1}{k}\sin(2\pi k x)\sin(2\pi k t) \quad (25)$$

consists of a sum of two sinusoidal waves, where k denotes an arbitrary frequency. We fix the frequency to $k = 3$. A visual representation of the solution over the computational domain is given in Fig. 2.

We use this benchmark for both a serial and a Parareal study. Therefore, we split B1 further in two subcases:

(a) For the serial study, used to investigate the stability of the time stepping schemes, we use the discretization of the spatial domain with $N \in \{42, 85, 170\}$ spectral modes, corresponding to a spatial discretization width of $\Delta x \in \{1/64, 1/128, 1/256\}$, and the time domain with step sizes of $\Delta t \in \{10^{-4}, 10^{-3}, 10^{-2}\}$. For the viscosity we study the values $\nu \in \{0, 10^{-4}, 10^{-3}, \ldots, 1\}$.
(b) In the Parareal study, used to analyze the coarse propagators in a parameter region found by (a), we use $N = 170$ spectral modes in space, which corresponds to a spatial discretization width of $\Delta x = 1/256$. The viscosity is chosen from the set $\nu \in \{n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$.

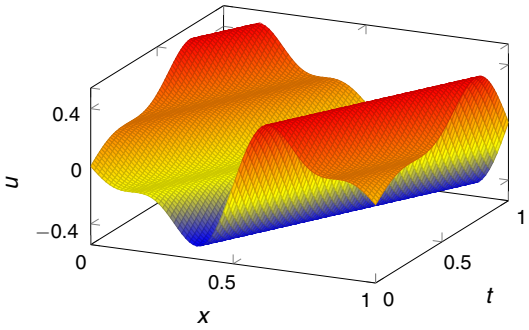

Fig. 3 Analytical solution u of benchmark B2 over the computational domain $(x, t) \in [0, 1]^2$

6.1.2 B2: Transport of a wave over time

The second benchmark (B2) is based on the smoothed sawtooth function used in [25], which is described by the finite series

$$\overline{u}(t, x) = \frac{1}{2} \sum_{k=1}^{k_{max}} \sin(2\pi kx - \pi kt + \pi k)\Phi(k, \epsilon), \tag{26}$$

with

$$\Phi(k, \epsilon) = \frac{\epsilon}{\sinh(\frac{1}{2}\epsilon\pi k)} \tag{27}$$

being a smoothing function used to suppress the amplitudes of high wave numbers. The parameters in this work are set to $k_{max} = 3$ and $\epsilon = 0.1$. This choice results in a transport of a wave over time (see Fig. 3). The spatial domain is discretized with $N = 170$ spectral modes (spatial discretization width $\Delta x = 1/256$). The studies are conducted with viscosities of $\nu \in \{n \times 10^{-4}, n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$.

6.2 Stability study of serial time stepping

We start with a stability study considering the settings of Benchmark B1 in order to get a general idea of time step limits for each scheme described in Sect. 4. These studies also illustrate the time step limitations, which motivate the development of parallel-in-time methods. The results are given in Table 2. Parameter combinations which lead to a stable computation are marked with a check mark. Unstable computations are indicated by 'X'.

The results for the fully explicit RK scheme are shown in the first row of tables. For all of the three time step sizes, we see instabilities for diffusion dominated problems. Only with the coarsest time step size can an unstable behavior be observed in the advection dominated parameter region (small viscosities).

Table 2 Results of the serial time stepping stability study with the RK, IMEX and SL scheme for all parameter combinations

$\Delta t = 10^{-4}$				$\Delta t = 10^{-3}$				$\Delta t = 10^{-2}$			
$\nu \backslash N$	42	85	170	$\nu \backslash N$	42	85	170	$\nu \backslash N$	42	85	170
RK											
0	✓	✓	✓	0	✓	✓	✓	0	✓	X	X
10^{-4}	✓	✓	✓	10^{-4}	✓	✓	✓	10^{-4}	✓	X	X
10^{-3}	✓	✓	✓	10^{-3}	✓	✓	✓	10^{-3}	✓	✓	X
10^{-2}	✓	✓	✓	10^{-2}	✓	✓	X	10^{-2}	X	X	X
10^{-1}	✓	✓	X	10^{-1}	X	X	X	10^{-1}	X	X	X
1	X	X	X	1	X	X	X	1	X	X	X
IMEX											
0	✓	✓	✓	0	✓	✓	✓	0	✓	X	X
10^{-4}	✓	✓	✓	10^{-4}	✓	✓	✓	10^{-4}	✓	X	X
10^{-3}	✓	✓	✓	10^{-3}	✓	✓	✓	10^{-3}	✓	✓	✓
10^{-2}	✓	✓	✓	10^{-2}	✓	✓	✓	10^{-2}	✓	✓	✓
10^{-1}	✓	✓	✓	10^{-1}	✓	✓	✓	10^{-1}	✓	✓	X
1	✓	✓	✓	1	✓	✓	✓	1	✓	✓	X
SL											
All stable				All stable				All stable			

Checkmarks indicate a stable calculation and unstable calculations are indicated with 'X'

Comparing these results with the results given in the second row of tables, corresponding to the IMEX scheme, we see the impact of treating the diffusive stiffness with an implicit time discretization, hence reducing the stability restrictions of this term. Therefore, we see fewer parameter combinations which are unstable in the diffusion dominated region. Since the explicit discretization of the advective part is the same as with the RK scheme, we get the same stability behavior in the advective region.

All parameter combinations with the SL formulation lead to a stable computation, so it is only constrained by its accuracy. This shows the potential of the SL formulation as a coarse time stepper for parallel-in-time schemes, as larger coarse time steps are possible. As SL schemes accurately handle advection, this is also particularly relevant for advection dominated problems.

6.3 Parareal B1: Sinusoidal waves

In this section, we apply the Parareal algorithm to the first benchmark case (B1) described in Sect. 6.1.

First, we compare the three different time stepping scheme combinations within the Parareal algorithm at a moderate viscosity. Second, we investigate IMEX and SL in greater depth to examine a wider range of viscosities and the behavior with advection dominated flows. Finally, we take a look at

the influence of the time step size of the coarse solver on the convergence behavior of the Parareal algorithm.

6.3.1 Comparison of the time stepping schemes

We compare the coarse solvers RK, IMEX, and SL with respect to their stability within the Parareal algorithm. The settings of B1 are applied with the exception that we use only a fixed viscosity of $\nu = 10^{-2}$.

In Fig. 4 the maximal absolute error of the spatial domain between the numerical and analytical solution $\epsilon_{max} = \max(|u - \overline{u}|)$ is plotted over the time domain for the first four Parareal iterations. Additionally, ϵ_{max} calculated with the fine time stepping scheme in serial is plotted with a dashed line.

Figure 4a shows in each iteration a diverging behavior of the coarse solver RK after a few time slices, which propagates into the solution. This is expected because RK is not stable for the parameter set of B1, see Sect. 6.2. For this reason we skip RK in all further calculations. Even though, the coarse solver diverges within each iteration of the Parareal algorithm, the algorithm converges finally after $k = 100$ iterations. This is exactly the number of time slices and, therefore, the maximum expected number of iterations [2], since the fine solver has passed through all slices as a serial algorithm.

In comparison to RK, the coarse solver IMEX shows a good approximation of the solution already in the first Parareal iteration, see Fig. 4b. Further iterations show a fast convergence for the first 23 time slices to the error of the fine solver on the order of $\epsilon_{max} = \mathcal{O}(10^{-7})$. For the other time slices the error increases in each iteration resulting from a divergent behavior of the coarse solver. This shows that stable coarse and fine solvers do not ensure stability over all Parareal iterations. The Parareal algorithm converges to the serial fine solution within the pre-set tolerance after $k = 100$ Parareal iterations, as expected.

Finally, Fig. 4c shows the results for SL. In contrast to both previous coarse solvers SL is stable over the whole time domain for each iteration. Due to this we can see a fast convergence to the serial fine solution uniformly over the whole time domain. After just four iterations the error is visually nearly indistinguishable from the error of the serial fine solution. The preset tolerance of the Parareal algorithm leads to fulfilling the convergence criterion after $k = 9$ iterations. Additionally, we have included the serial fine solution of the SL scheme (dash-dotted) to underline the motivation for the choice of IMEX as the fine time stepping scheme of the Parareal algorithm. The large error of the SL method is caused due to its dependency on its spatial discretization errors, which are not of spectral accuracy.

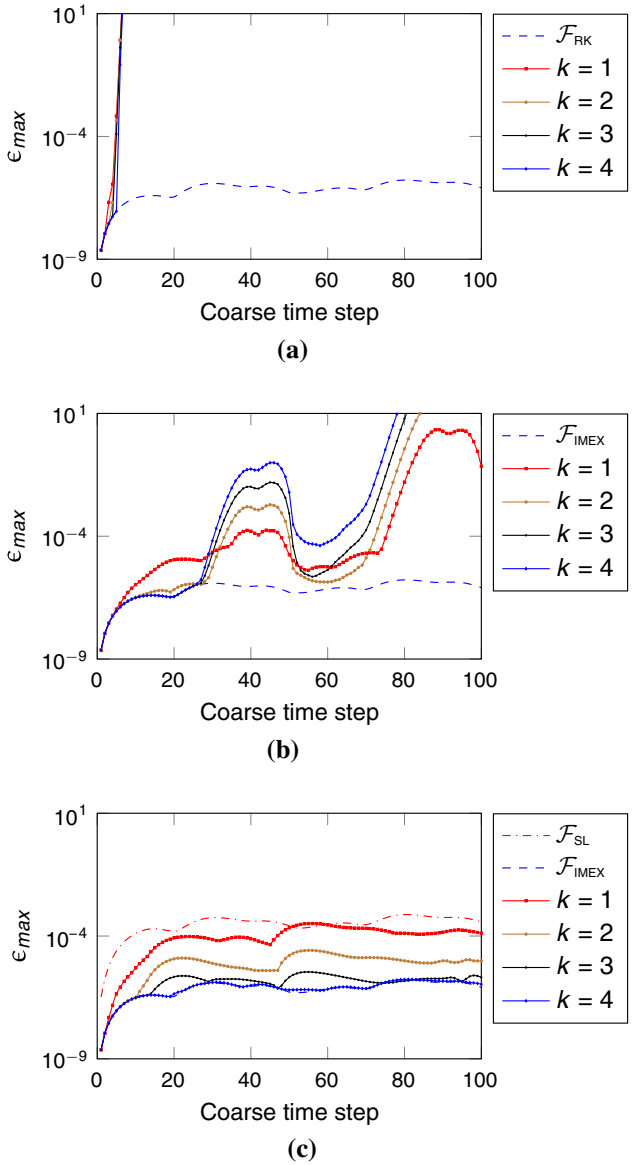

Fig. 4 Maximal absolute error between numerical and analytical solution $\epsilon_{max} = \max(|u - \overline{u}|)$ plotted over the time domain for combinations of the fine $\mathcal{F}$ solver IMEX and three different coarse $\mathcal{C}$ solvers of the Parareal algorithm. Given are the errors for the first four Parareal iterations k and the error of $\mathcal{F}$. **a** $\mathcal{C}$: RK, **b** $\mathcal{C}$:IMEX, **c** $\mathcal{C}$: SL

6.3.2 Influence of the SL on the advective problem

We investigate the influence of the semi-Lagrangian formulation on the convergence behavior of the Parareal algorithm applied to advective problems. To reach this goal we now use the set of viscosities described in benchmark B1.

In Fig. 5 the error ϵ_{max} is shown against the time domain for some of the viscosities ν. In the case of IMEX we show iterations 1, 2, 40 and 80 of the Parareal algorithm. We can observe a stable behavior of the coarse solver for $\nu \geq 0.002$ in the first Parareal iteration. The second iteration shows

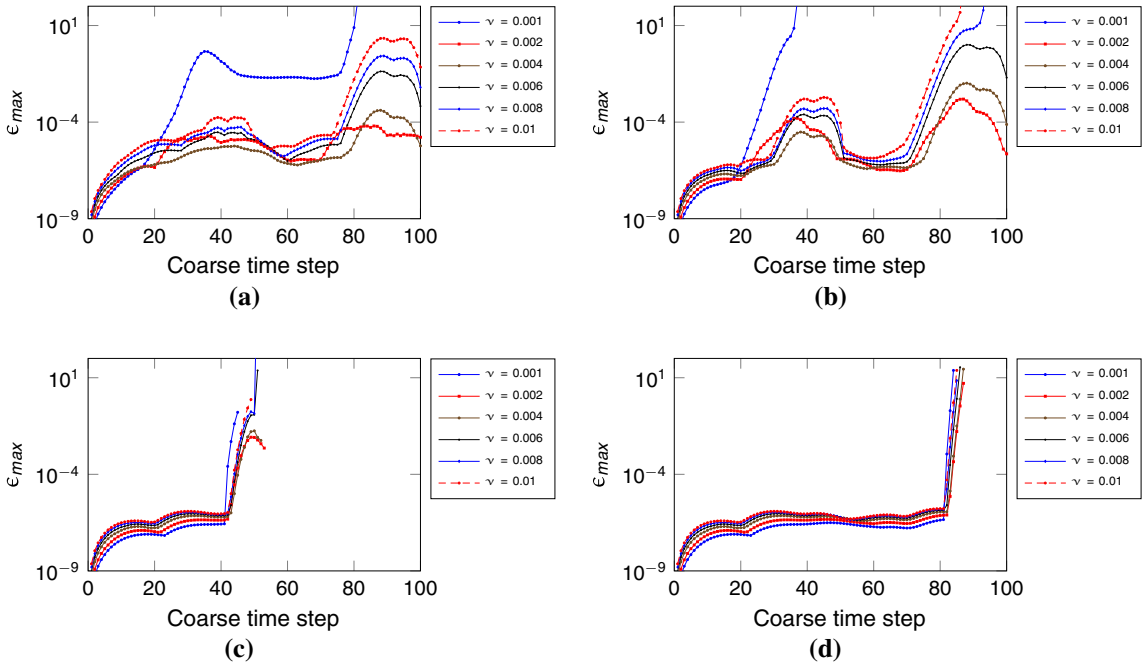

Fig. 5 Maximal absolute error between numerical and analytical solution $\epsilon_{max} = \max(|u - \overline{u}|)$ plotted over the time domain for the Parareal iterations $k \in \{1, 2, 40, 80\}$. Fine $\mathcal{F}$ and coarse $\mathcal{C}$ solver of the used Parareal algorithm is IMEX. Given are the errors for chosen viscosities of the set $\nu \in \{n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. **a** $k = 1$, **b** $k = 2$, **c** $k = 40$, **d** $k = 80$

increasing errors for all ν with already unstable behavior of the coarse solver for $\nu \geq 0.008$. None of these calculations shows a rapid convergence with Parareal, as $k \geq 97$ iterations are necessary to reach the desired tolerance. Iterations 40 and 80 show how the solution of the fine solver is propagated one time slice per iteration.

Studies for SL are provided in Fig. 6. Here, only the first two iterations are visualized since a sufficient approximation to the solution is already obtained. First of all, we observe a stable behavior for the coarse solver with all examined viscosities. The SL scheme stably solves the advection part without bounds on the CFL number independent of possible (large) errors in the velocity caused by the large time step size used in the coarse propagator. Also, the velocities and trajectories are calculated using averaged (interpolated) velocities, which smooths out near-grid scale velocity variations, which are usually responsible for instabilities [11]. On the other hand, the IMEX scheme is very sensitive to velocity variations, especially for very small viscosities, since no smoothing is used. When large time steps are used in Parareal, large errors in velocity are expected, particularly at the end of the time frame. Such errors may cause the CFL number bounds to be violated, and, therefore, the solutions do not contain the information of their domain of dependence. This can naturally drive the numerical solution away from the expected one.

We also notice in Fig. 7, where the total error of the space-time domain is plotted over the first three iterations, an equally fast convergence for all different viscosities from iteration one to two. Between iteration two and three the error for the calculations with the larger viscosities is reduced less than for the smaller viscosities, in the end leading to one additional iteration needed for convergence for $\nu = 0.01$. For all SL experiments performed in the test case the algorithm converged within $k \leq 9$ iterations, which corroborates the statement made in Sect. 6.2 that a SL formulation has potential as an efficient coarse solver (fast convergence and cheap compared to the fine time stepper) for parallel-in-time methods.

Additionally, we want to mention that the Parareal algorithm converges to the fine solution with SL for even smaller viscosities up to $\nu = 0$. The number of iterations until convergence increases gradually with decreasing viscosities. Convergence is reached with $k = 12$ Parareal iterations for $\nu = 0.0001$, $k = 16$ iterations with $\nu = 0.00005$ and $k = 23$ iterations with $\nu = 0$.

6.3.3 Influence of the coarse time step size on the convergence of Parareal

In this section, we study the influence of the coarse time step size ΔT on the convergence of the Parareal algorithm.

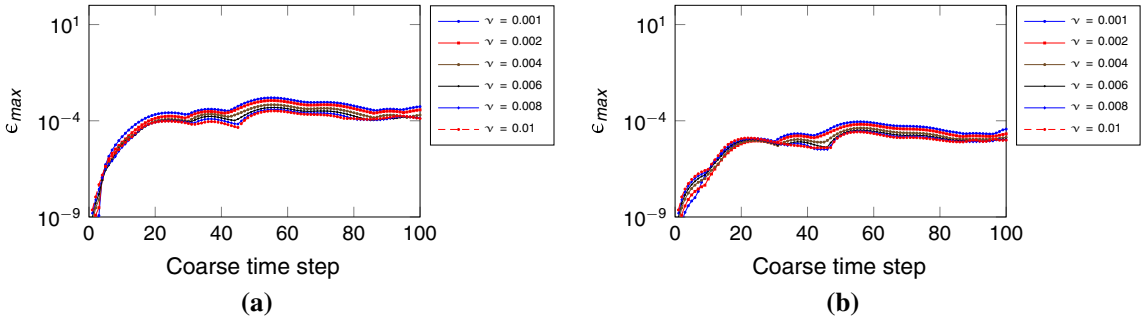

Fig. 6 Maximal absolute error between numerical and analytical solution $\epsilon_{max} = \max(|u - \overline{u}|)$ plotted over the time domain for the first two Parareal iterations k. Fine $\mathcal{F}$ and coarse $\mathcal{C}$ solver of the Parareal algorithm are IMEX and SL, respectively. Given are the errors for chosen viscosities of the set $\nu \in \{n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. **a** $k = 1$, **b** $k = 2$

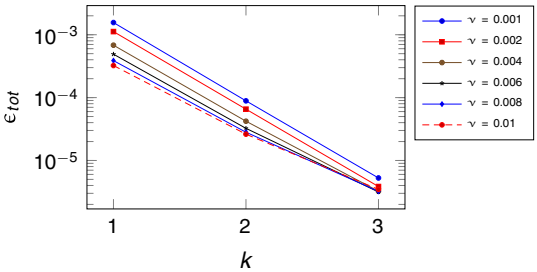

Fig. 7 Total error ϵ_{tot} of the space-time domain of each iteration k of the Parareal algorithm plotted over iterations 1, 2, and 3 for chosen viscosities of the set $\nu \in \{n \times 10^{-3} | n \in \{1, 2, \ldots, 10\}\}$. Fine $\mathcal{F}$ and coarse $\mathcal{C}$ solver are IMEX and SL, respectively

Table 3 Number of iterations to convergence of the Parareal algorithm with $\mathcal{F}$:IMEX and $\mathcal{C}$ as noted in the table for two different viscosities and four different time step sizes of the coarse solver applied to B1

ν	ΔT	$\mathcal{C}$: IMEX	$\mathcal{C}$: SL
0.005	1×10^{-2}	99	8
0.005	5×10^{-3}	5	6
0.005	2.5×10^{-3}	4	4
0.005	1.25×10^{-3}	3	4
0.01	1×10^{-2}	100	9
0.01	5×10^{-3}	32	6
0.01	2.5×10^{-3}	4	5
0.01	1.25×10^{-3}	3	4

The fine time step size is fixed to $\Delta t = 10^{-6}$

For this study, we use benchmark B1 with the following changes: we focus on the two viscosities $\nu \in \{0.005, 0.01\}$ and vary the time step size of the coarse solver $\Delta T \in \{1/2^i \times 10^{-2} | i \in \{0, \ldots, 3\}\}$. For the coarse solvers of the Parareal algorithm we use IMEX and SL.

The number of iterations needed for convergence for the Parareal algorithm is listed in Table 3 for all combinations of viscosities and time step sizes. Based on Sect. 6.3.2, we know that the IMEX coarse solver leads to divergent behavior for $\Delta T = 1 \times 10^{-2}$, which results in the high number of iterations to converge. This is also the explanation for the 32 iterations with $\nu = 0.01$ and $\Delta T = 5 \times 10^{-3}$. The reduced number of iterations for this case is caused by the unstable behavior appearing at a later iteration. All other results show no instabilities of the solvers during all iterations.

A reduction of ΔT leads to a reduction of the number of iterations to converge of the Parareal algorithm. This matches with the theory of [18]. An exception is the reduction from $\Delta T = 2.5 \times 10^{-3}$ to $\Delta T = 1.25 \times 10^{-3}$, which does not lead to a reduction in the number of iterations with the SL coarse solver. A reason for this can be found in the coarse time step size, which does not lead to a reduction in the error of the coarse solver due to SL depending also on the spatial grid size. Reducing the spatial grid size from $\Delta x = 1/256$ to $\Delta x = 1/512$ reduces the error of the coarse solver indicating dominating errors from the 2nd order spatial interpolation of the SL scheme.

The results show the advantage of the SL formulation for the cases where IMEX has no guarantee of stability over the whole Parareal algorithm. In these cases the additional computation and communication of the SL formulation lead to a large decrease in iterations to converge compared to using IMEX as a coarse solver.

6.4 Parareal B2: Transport of a wave over time

We continue with results of benchmark B2 with a focus on the required number of iterations to converge.

From Fig. 8, we can see computations with a viscosity $\nu \geq 0.0003$ converging within $k \leq 5$ iterations for both coarse solvers (IMEX and SL). With IMEX one iteration fewer is needed between $\nu = 0.0003$ and $\nu = 0.001$. This can again be explained by the fact that the SL solver is also influenced by the spatial discretization, and, therefore, the error of the coarse solver has an additional bound preventing it from a better initial guess for the fine solver.

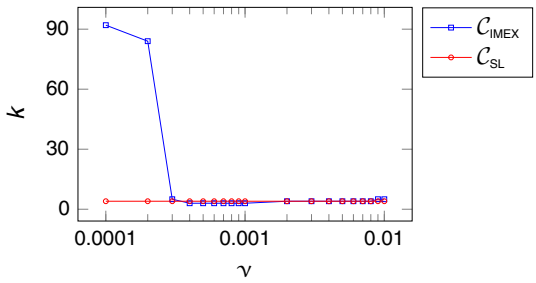

Fig. 8 Parareal iterations k needed for convergence of the algorithm plotted against the viscosities ν for benchmark B2 with IMEX as the fine solver and IMEX and SL as the coarse solver

The fast convergence of these calculations can be attributed to a stable behavior of the solvers in all Parareal iterations. The IMEX coarse solver is more stable with B2 than with B1, because the maximal velocity is smaller for B2, which can be seen by comparing Figs. 2 and 3.

For the two smallest viscosities we observe a drastic increase in iterations to convergence with IMEX whereas the algorithm with SL still converges within $k = 4$ iterations. The increase results again from instabilities of the coarse solver within the Parareal iterations. This shows once more the potential of the SL formulation as a coarse solver for parallel-in-time methods with advection dominated problems.

7 Conclusion

In this work, we have shown the potential of the semi-Lagrangian formulation as a coarse solver for parallel-in-time methods using the Parareal method applied to the viscous Burgers equation.

Since our focus was on the benefits of the semi-Lagrangian formulation as a coarse solver for advection dominated problems, we investigated two benchmarks with different characteristics based on manufactured solutions in the region of small viscosities.

We compared the semi-Lagrangian method in combination with an implicit Euler (SL) to an explicit (RK) and an implicit–explicit (IMEX) Runge–Kutta method for the coarse solver. The fine solver was chosen to be the IMEX method. The RK method was not stable as a coarse solver for the investigated cases and, therefore, did not lead to any speed up regarding the number of iterations to convergence. With the considered benchmarks, we found that for parameter combinations with both IMEX and SL turned out to be stable as the coarse solver all Parareal calculations need a similar number of iterations to convergence. Since SL is computationally more expensive and needs additional communication, the method of choice in these cases is IMEX. In all parameter combinations examined where the IMEX method was unstable as a coarse solver the SL method is the method of choice as it needs far fewer iterations to convergence due to its stable behavior.

The stability of SL makes a larger range of viscosities suitable to the Parareal method compared to IMEX. Continuously decreasing the viscosity also leads to an increasing number of iterations for SL. We were able to show a convergence with potential for speed up even for a viscosity of $\nu = 0$.

Since our implementation of the Parareal algorithm is run serially, we can only show the potential of the SL formulation based on number of iterations needed for convergence. In further work it has to be investigated how large the speed up actually is considering the additional computation and communication necessary with SL.

References

1. Ascher, U.M., Ruuth, S.J., Spiteri, R.J.: Implicit-explicit Runge–Kutta methods for time-dependent partial differential equations. Appl. Numer. Math. **25**(2–3), 151–167 (1997). https://doi.org/10.1016/S0168-9274(97)00056-1
2. Baffico, L., Bernard, S., Maday, Y., Turinici, G., Zrah, G.: Parallel-in-time molecular-dynamics simulations. Phys. Rev. E **66**(057), 701 (2002). https://doi.org/10.1103/PhysRevE.66.057701
3. Bal, G.: On the convergence and the stability of the parareal algorithm to solve partial differential equations. In: Kornhuber, R., et al. (eds.) Domain Decomposition Methods in Science and Engineering. Lecture Notes in Computational Science and Engineering, vol. 40, pp. 426–432. Springer, Berlin (2005). https://doi.org/10.1007/3-540-26825-1_43
4. Barros, S.R., Peixoto, P.S.: Computational aspects of harmonic wavelet Galerkin methods and an application to a precipitation front propagation model. Comput. Math. Appl. **61**(4), 1217–1227 (2011)
5. Bateman, H.: Some recent researches on the motion of fluids. Mon. Weather Rev. **43**(4), 163–170 (1915)
6. Bauer, P., Thorpe, A., Brunet, G.: The quiet revolution of numerical weather prediction. Nature **525**(7567), 47–55 (2015)
7. Bonaventura, L.: An Introduction to Semi-Lagrangian Methods for Geophysical Scale Flows. Lecture Notes ERCOFTAC Leonhard Euler Lectures. SAM-ETH, Zurich (2004)
8. Burgers, J.: A mathematical model illustrating the theory of turbulence. Adv. Appl. Mech. **1**, 171–199 (1948). https://doi.org/10.1016/S0065-2156(08)70100-5
9. Christlieb, A.J., Macdonald, C.B., Ong, B.W.: Parallel high-order integrators. SIAM J. Sci. Comput. **32**(2), 818–835 (2010). https://doi.org/10.1137/09075740X
10. Côté, J.: Time-parallel algorithms for weather prediction and climate simulation. https://www.newton.ac.uk/files/seminar/20121023121012359-153396.pdf. Accessed 03 May 2017 (2012)
11. Durran, D.: Numerical Methods for Fluid Dynamics: With Applications to Geophysics. Texts in Applied Mathematics. Springer, New York (2010). https://doi.org/10.1007/978-1-4419-6412-0
12. Dutt, A., Greengard, L., Rokhlin, V.: Spectral deferred correction methods for ordinary differential equations. BIT Numer. Math. **40**(2), 241–266 (2000). https://doi.org/10.1023/A:1022338906936
13. Emmett, M., Minion, M.L.: Toward an efficient parallel in time method for partial differential equations. Commun. Appl. Math.

Comput. Sci. **7**, 105–132 (2012). https://doi.org/10.2140/camcos.2012.7.105
14. Farhat, C., Chandesris, M.: Time-decomposed parallel time-integrators: Theory and feasibility studies for fluid, structure, and fluid-structure applications. Int. J. Numer. Method Eng. **58**(9), 1397–1434 (2003). https://doi.org/10.1002/nme.860
15. Friedhoff, S., Falgout, R.D., Kolev, T.V., MacLachlan, S.P., Schroder, J.B.: A multigrid-in-time algorithm for solving evolution equations in parallel. In: Presented at Sixteenth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, United States, Mar 17–22, 2013, http://www.osti.gov/scitech/servlets/purl/1073108 (2013)
16. Gander, M.J.: Analysis of the parareal algorithm applied to hyperbolic problems using characteristics. Boletin de la Sociedad Espanola de Matemática Aplicada **42**, 21–35 (2008)
17. Gander, M.J.: 50 years of time parallel time integration. In: Carraro, T., Geiger, M., Körkel, S., Rannacher, R. (eds.) Multiple Shooting and Time Domain Decomposition. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23321-5_3
18. Gander, M.J., Hairer, E.: Analysis for parareal algorithms applied to Hamiltonian differential equations. J. Comput. Appl. Math. 259, Part A(0):2–13. In: Proceedings of the Sixteenth International Congress on Computational and Applied Mathematics (ICCAM-2012), Ghent, Belgium, 9–13 July, 2012 (2014). https://doi.org/10.1016/j.cam.2013.01.011
19. Gander, M.J., Vandewalle, S.: Analysis of the parareal time-parallel time-integration method. SIAM J. Sci. Comput. **29**(2), 556–578 (2007). https://doi.org/10.1137/05064607X
20. Gottlieb, D., Orszag, S.: Numerical Analysis of Spectral Methods: Theory and Applications. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia (1977)
21. Hortal, M.: The development and testing of a new two-time-level semi-lagrangian scheme (SETTLS) in the ECMWF forecast model. Q. J. R. Meteorol. Soc. **128**(583), 1671–1687 (2002). https://doi.org/10.1002/qj.200212858314
22. Horton, G., Vandewalle, S.: A space-time multigrid method for parabolic partial differential equations. SIAM J. Sci. Comput. **16**(4), 848–864 (1995). https://doi.org/10.1137/0916050
23. Horton, G., Vandewalle, S., Worley, P.: An algorithm with polylog parallel complexity for solving parabolic partial differential equations. SIAM J. Sci. Comput. **16**(3), 531–541 (1995). https://doi.org/10.1137/0916034
24. Kennedy, C.A., Carpenter, M.H.: Additive Runge–Kutta schemes for convection-diffusion-reaction equations. Appl. Numer. Math. **44**(1), 139–181 (2003). https://doi.org/10.1016/S0168-9274(02)00138-1
25. Kooij, G.L., Botchev, M.A., Geurts, B.J.: A block Krylov subspace implementation of the time-parallel Paraexp method and its extension for nonlinear partial differential equations. J. Comput. Appl. Math. **316**, 229–246 (2017). https://doi.org/10.1016/j.cam.2016.09.036. (**Selected Papers from NUMDIFF-14**)
26. LeVeque, R.: Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems. SIAM e-books, Society for Industrial and Applied Mathematics, Philadelphia (2007)
27. Lions, J.L., Maday, Y., Turinici, G.: A parareal in time discretization of PDE's. Comptes Rendus de l'Académie des Sci.-Ser. I—Math. **332**, 661–668 (2001). https://doi.org/10.1016/S0764-4442(00)01793-6
28. Lubich, C., Ostermann, A.: Multi-grid dynamic iteration for parabolic equations. BIT Numer. Math. **27**(2), 216–234 (1987). https://doi.org/10.1007/BF01934186
29. Peixoto, P.S., Barros, S.R.: On vector field reconstructions for semi-Lagrangian transport methods on geodesic staggered grids. J. Comput. Phys. **273**, 185–211 (2014)
30. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., et al.: Numerical Recipes, vol. 3. Cambridge University Press, Cambridge (1989)
31. Reynolds-Barredo, J., Newman, D., Sanchez, R., Jenko, F.: A novel, semilagrangian, coarse solver for the Parareal technique and its application to 2D fluid drift-wave (BETA) and 5D gyrokinetic (GENE), turbulence simulations. In: APS Meeting Abstracts, vol 1, p 8128P (2013)
32. Ruprecht, D., Krause, R.: Explicit parallel-in-time integration of a linear acoustic-advection system. Comput. Fluids **59**, 72–83 (2012). https://doi.org/10.1016/j.compfluid.2012.02.015
33. Schreiber, M., Peixoto, P.S., Haut, T., Wingate, B.: Beyond spatial scalability limitations with a massively parallel method for linear oscillatory problems. Int. J. High Perform. Comput. Appl. **29**(3), 261–273 (2016)
34. Schreiber, M., Peixoto, P., Schmitt, A.: (2017) https://github.com/schreiberx/sweet. Accessed 4 May 2017
35. Staff, G.A., Rønquist, E.M.: Stability of the parareal algorithm. In: Kornhuber, R., et al. (eds.) Domain Decomposition Methods in Science and Engineering. Lecture Notes in Computational Science and Engineering, vol. 40, pp. 449–456. Springer, Berlin (2005). https://doi.org/10.1007/3-540-26825-1_46
36. Staniforth, A., Côté, J.: Semi-Lagrangian integration schemes for atmospheric models—a review. Mon. Weather Rev. **119**(9), 2206–2223 (1991)
37. Steiner, J., Ruprecht, D., Speck, R., Krause, R.: Convergence of Parareal for the Navier–Stokes Equations Depending on the Reynolds Number, pp. 195–202. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-10705-9_19
38. Sutter, H.: The free lunch is over: a fundamental turn toward concurrency in software. Dr Dobbs J. **30**(3), 202–210 (2005)
39. Swarztrauber, P.N., Sweet, R.A.: The Fourier and Cyclic Reduction Methods for Solving Poissons Equation. Wiley, New York (1996)
40. Vandewalle, S., Van de Velde, E.: Space-time concurrent multigrid waveform relaxation. Ann. Numer. Math. **1**(1–4), 347–360 (1994). https://doi.org/10.13140/2.1.1146.1761
41. Wedi, N.P., Bauer, P., Deconinck, W., Diamantakis, M., Hamrud, M., Kuehnlein, C., Malardel, S., Mogensen, K., Mozdzynski, G., Smolarkiewicz, P.K.: The modelling infrastructure of the integrated forecasting system: recent advances and future challenges. European Centre for Medium-Range Weather Forecasts (2015)
42. Wesseling, P.: Principles of Computational Fluid Dynamics. Springer Series in Computational Mathematics. Springer, Berlin (2009)
43. Williamson, D.L.: The evolution of dynamical cores for global atmospheric models. J. Meteorol. Soc. Japan Ser. II **85**, 241–269 (2007)