Journal of Computational Physics: X 7 (2020) 100057

Contents lists available at ScienceDirect

Journal of Computational Physics: X

www.elsevier.com/locate/jcpx

Parallel-in-time integration of kinematic dynamos

Andrew T. Clarke [a,*], Christopher J. Davies [b], Daniel Ruprecht [c,e], Steven M. Tobias [d]

[a] Centre for Doctoral Training in Fluid Dynamics, School of Computing, University of Leeds, Leeds, LS2 9JT, UK
[b] School of Earth and Environment, University of Leeds, Leeds, LS2 9JT, UK
[c] Lehrstuhl Computational Mathematics, Institut für Mathematik, Technische Universität Hamburg, 21073 Hamburg, Germany
[d] Department of Applied Mathematics, University of Leeds, Leeds, LS2 9JT, UK
[e] School of Mechanical Engineering, University of Leeds, Leeds, LS2 9JT, UK

ARTICLE INFO

Article history:
Received 1 February 2019
Received in revised form 10 January 2020
Accepted 15 March 2020
Available online 24 March 2020

Keywords:
Parareal
Parallel-in-time
Kinematic dynamo
Induction equation
Spectral methods
IMEX

ABSTRACT

The precise mechanisms responsible for the natural dynamos in the Earth and Sun are still not fully understood. Numerical simulations of natural dynamos are extremely computationally intensive, and are carried out in parameter regimes many orders of magnitude away from real conditions. Parallelization in space is a common strategy to speed up simulations on high performance computers, but eventually hits a scaling limit. Additional directions of parallelization are desirable to utilise the high number of processor cores now available. Parallel-in-time methods can deliver speed up in addition to that offered by spatial partitioning but have not yet been applied to dynamo simulations. This paper investigates the feasibility of using the parallel-in-time algorithm Parareal to speed up initial value problem simulations of the kinematic dynamo, using the open source Dedalus spectral solver. Both the time independent Roberts and time dependent Galloway-Proctor 2.5D dynamos are investigated over a range of magnetic Reynolds numbers. Speedups beyond those possible from spatial parallelisation are found in both cases. Results for the Galloway-Proctor flow are promising, with Parareal efficiency found to be close to 0.3. Roberts flow results are less efficient, but Parareal still shows some speed up over spatial parallelisation alone. Parallel in space and time speed ups of ~ 300 were found for 1600 cores for the Galloway-Proctor flow, with total parallel efficiency of ~ 0.16.

1. Introduction

Dynamo theory seeks to explain the processes that generate magnetic fields in stars, planets, and galaxies. These magnetic fields are sustained by currents generated by the flow of a conducting fluid [1,2], such as molten iron in the Earth's core, or plasma in the Sun and stars. The fluid velocity, $\boldsymbol{u}$, twists, stretches, and shears field lines, counteracting the Ohmic diffusion of the magnetic field, $\boldsymbol{B}$, which would otherwise cause the field to decay [3]. The dynamo problem is typically modelled using the induction equation

$$\frac{\partial \boldsymbol{B}}{\partial t} = \nabla \times (\boldsymbol{u} \times \boldsymbol{B}) + \frac{1}{R_m}\nabla^2 \boldsymbol{B}, \tag{1}$$

* Corresponding author.
E-mail addresses: andyclarke101@gmail.com, scatc@leeds.ac.uk (A.T. Clarke).

https://doi.org/10.1016/j.jcpx.2020.100057

coupled with the momentum equation for the fluid, which determines $\boldsymbol{u}$. The magnetic Reynolds number is defined by

$$R_m = \frac{UL}{\eta}, \tag{2}$$

where U is a characteristic speed, L a characteristic length scale, and η is the magnetic diffusivity [4]. A solution of this coupled system is called a dynamo if the magnetic field continues to be generated as time $t \to \infty$. The smallest scale of magnetic features created by the evolution of the induction equation (1) is the resistive scale which is proportional to ${R_m}^{-1/2}$ [3], so that the spatial resolution required to capture these structures increases as ${R_m}^{1/2}$. This leads to very high computational requirements that limit the parameter regimes that can be studied. Simulations are currently carried out at parameter regimes far removed from those found in natural dynamos of real stars and planets [5].

Owing to the complexity of the full dynamo problem much attention has focused on the simpler problem of kinematic dynamos where $\boldsymbol{u}$ is prescribed. As the induction equation is linear in $\boldsymbol{B}$, solutions to the kinematic dynamo problem are either exponentially growing or decaying with a well defined growth-rate for a statistically stationary $\boldsymbol{u}$. Due to a number of anti-dynamo theorems [6,7], relatively complex space or time dependent flows are required in order to generate a growing magnetic field. Such a problem is difficult to address analytically, so numerical and computational methods are employed.

If $\boldsymbol{u}$ is steady, then the growth-rate can be found by solving an eigenvalue problem, whilst if the flow is periodic in time, the growth-rate can be found by solving a Floquet problem. If in addition the flow is independent of one co-ordinate, say the z-co-ordinate (2.5D flow), then one can seek monochromatic solutions of the form $\boldsymbol{B}(x, y, z, t) = \boldsymbol{b}(x, y, t)\mathrm{e}^{\mathrm{i}k_z z}$, so that k_z becomes a parameter and the problem becomes 2D. The behaviour of dynamos at different R_m has received much attention [8–10]. Low R_m systems are dominated by diffusion, while high R_m systems are dominated by advection. Laboratory and engineering type flows tend to have $R_m < 1$ [11], flows in the Earth's core are characterised by $R_m \sim 10^2 - 10^3$ [12], whilst flows in the Sun have $R_m \sim 10^6 - 10^{10}$ [13].

Behaviour of dynamos at large R_m is of particular interest astrophysically. If the growth-rate stays bounded away from zero as $R_m \to \infty$ then a dynamo is called *fast*, otherwise it is called *slow*. It is a necessary (but not sufficient) condition that a fast dynamo have chaotic Lagrangian particle paths [14]. In 2.5D flows, this can only be the case for unsteady flows. In fully 3D flows, the paths tend to be chaotic even without time dependence. No fast dynamos have yet been proven mathematically, but some have been found to act as fast dynamos in numerical simulations.

A particular choice of 2.5D cellular steady flow was considered by Roberts [8]. The Galloway-Proctor circularly polarised (CP) [9] flow is an extension of this flow to include time dependence. While the Roberts flow must act as a slow dynamo, as it is steady, the Galloway-Proctor dynamo is thought to be fast. The Roberts flow dynamo has been used to investigate the experimental dynamo at Karlsruhe [10], whilst the Galloway-Proctor flow was used to investigate the formation of large scale magnetic fields at high R_m [15].

High accuracy solutions are needed for dynamo simulations and so the majority of studies use spectral methods to discretise in space [16], expanding in Fourier series, spherical harmonics, or Chebyshev polynomials as best fits the domain, to make use of their spectral convergence properties. Because of the high resolution requirements, very long compute times are found for the full dynamo system, even when simulations are run on large numbers of cores in high performance computing (HPC) facilities. Matsui et al. [17], for example, tested scaling on up to 16,384 cores while Schaeffer et al. [18] ran simulations that required over 10 million cpu hours. Transforming between spectral and spatial coordinates, usually via Fast Fourier Transform (FFT), acts as a limiting factor on the parallel scalability of pseudospectral codes [19]. Owing to the global communications requirements of spectral methods, simulations are currently unable to scale to the huge number of processors available in modern HPC facilities. Further ways to parallelize computations are therefore required to investigate more realistic parameter regimes.

Parallel-in-time methods can increase scalability of computer simulations beyond saturation of spatial parallelisation [20] and have been investigated for over 50 years [21]. The most widely studied parallel-in-time algorithm is Parareal [22], which was introduced in 2001 and has spurred a renewed interest in the subject. Further parallel in time methods have been proposed recently, like PFASST [23], PITA [24], and Paraexp [25]. The Parareal algorithm has been utilised in fluid flow problems [26,20], plasma physics [27], financial simulations [28], and in planetary mantle simulations [29]. The method has been extensively analysed mathematically by Gander and Vandewalle [30], and is thought to perform badly for purely hyperbolic and highly advective systems [31], as it primarily corrects amplitude defects, rather than phase or frequency defects [32]. There has been recent work in parallel-in-time for advective problems, where the linear operator was solved using the method of characteristics [33]. In this work, however, we seek to inform future studies of the non-linear dynamo problem, where the advective terms are no longer linear, and this approach would not be suitable. While some recent works demonstrate efficient application of Parareal to hyperbolic PDEs [34], the applicability of the method for dynamo simulations has not yet been studied.

The present paper studies the ability of Parareal to speed up kinematic dynamo simulations. It presents the first demonstration that parallel-in-time methods can deliver speedup for the induction equation beyond the saturation point of spatial parallelization. We introduce an implementation of the Parareal algorithm in the open source spectral solver Dedalus [35] and investigate its performance for the stationary Roberts [8] as well as the time-dependent Galloway-Proctor [9] flow. Although these flows are relatively simple, they generate complex dynamics in the magnetic field and are good initial test problems to demonstrate Parareal's efficiency for dynamo simulations.

2. The kinematic dynamo problem

In this work, the kinematic dynamo is studied for a subset of the ABC (Arnold, Beltrami and Childress) class of flows

$$\boldsymbol{u} = (C\sin(z) + B\cos(y),\, A\sin(x) + C\cos(z),\, B\sin(y) + A\cos(x))\,, \tag{3}$$

[36], with one of A, B, or C equal to 0, making the flow 2.5D.

2.1. Roberts flow

The Roberts flow is found from the ABC flow by setting $A = B = 1$, $C = 0$, giving $\boldsymbol{u} = (\cos(y), \sin(x), \sin(y) + \cos(x))$. Noting that $\nabla \cdot \boldsymbol{B} = \nabla \cdot \boldsymbol{u} = 0$, equation (1) becomes

$$\partial_t b_x(x,y,t) = -b_y \sin(y) - \cos(y)\partial_x b_x - \sin(x)\partial_y b_x - (\sin(y) + \cos(x))\mathrm{i}k_z b_x + \frac{1}{R_m}\left(\nabla^2 - k_z^2\right) b_x, \tag{4a}$$

$$\partial_t b_y(x,y,t) = b_x \cos(x) - \cos(y)\partial_x b_y - \sin(x)\partial_y b_y - (\sin(y) + \cos(x))\mathrm{i}k_z b_y + \frac{1}{R_m}\left(\nabla^2 - k_z^2\right) b_y, \tag{4b}$$

$$\frac{\partial b_x}{\partial x} + \frac{\partial b_y}{\partial y} + \mathrm{i}k_z b_z = 0. \tag{4c}$$

This flow is periodic in x and y, and equations (4) are solved in a 2π square plane with periodic boundary conditions.

2.2. Galloway-Proctor flow

The Galloway-Proctor circularly polarised (CP) flow adds time dependence to a Roberts like flow

$$\boldsymbol{u} = (C\sin(z + \sin\omega t) + B\cos(y + \cos\omega t),\, C\cos(z + \sin\omega t),\, B\sin(y + \cos\omega t))\,, \tag{5}$$

with $\omega = 1$ and $C = B = \sqrt{3/2}$, $A = 0$. We look for solutions of the form $\boldsymbol{B} = \boldsymbol{b}(y,z,t)\mathrm{e}^{\mathrm{i}k_x x}$. When put into (1), we have

$$\partial_t b_y(y,z,t) = b_z\partial_z v - u\mathrm{i}k_x b_y - v\partial_y b_y - w\partial_z b_y + \frac{1}{R_m}\left(\nabla^2 - {k_z}^2\right) b_y, \tag{6a}$$

$$\partial_t b_z(y,z,t) = b_y\partial_y w - u\mathrm{i}k_x b_z - v\partial_y b_z - w\partial_z b_z + \frac{1}{R_m}\left(\nabla^2 - {k_x}^2\right) b_z, \tag{6b}$$

$$\mathrm{i}k_x b_x + \frac{\partial b_y}{\partial y} + \frac{\partial b_z}{\partial z} = 0. \tag{6c}$$

2.3. Morphology of the Roberts and Galloway-Proctor fields

Fig. 1 shows the contours of the y-component of the magnetic field for $R_m = 3$ and 3000 in the Galloway-Proctor flow, and $R_m = 4$ and 4096 in the Roberts flow. Larger scale and more diffuse structures are present in the $R_m = 3$ and $R_m = 4$ simulations. Finer structures emerge in the $R_m = 3000$ and $R_m = 4096$ cases, showing the effect of the ${R_m}^{-1/2}$ scaling on the smallest structures. The Galloway-Proctor magnetic field morphology changes over time, with the periodic change in the velocity field, which leads to the more complicated pattern shown in Fig. 1d. For the Roberts flow, the magnetic field has a constant morphology and simply grows exponentially in magnitude.

3. Parareal algorithm

We give only a brief description of the Parareal algorithm, for a more detailed description see for example the works by Lions et al. or Gander and Vandewalle [22,30]. Parareal is an iterative method for solving initial value problems (IVPs) of the form

$$\frac{\partial U}{\partial t} = f\left(U(t), t\right), \quad U(0) = U_0, \quad 0 \le t \le T \tag{7}$$

where the right hand side f typically comes from spatially discretizing a partial differential equation. Parareal uses two different time stepping methods, the coarse method $\mathcal{G}$ and the fine method $\mathcal{F}$. The time domain is split up into N_p time slices, defined by the time-points $0 = t_0 < t_1 < \ldots < t_{N_p - 1} < t_{N_p} = T$, where N_p is the number of processors available for time parallelisation. We assume that all time slices are of equal length $\Delta T = T/N_p$. The fine time stepping method has time step δt, while the coarse method has time step Δt.

Parareal starts with running the coarse solver from $t = 0$ to $t = T$, giving an initial approximation to the solution U at time t_n, $U_n^{k=0}$, for every $n = 0, 1, \ldots, N_p$, where k denotes the Parareal iteration number, and n denotes the time slice. Then, the approximation is refined using the iteration

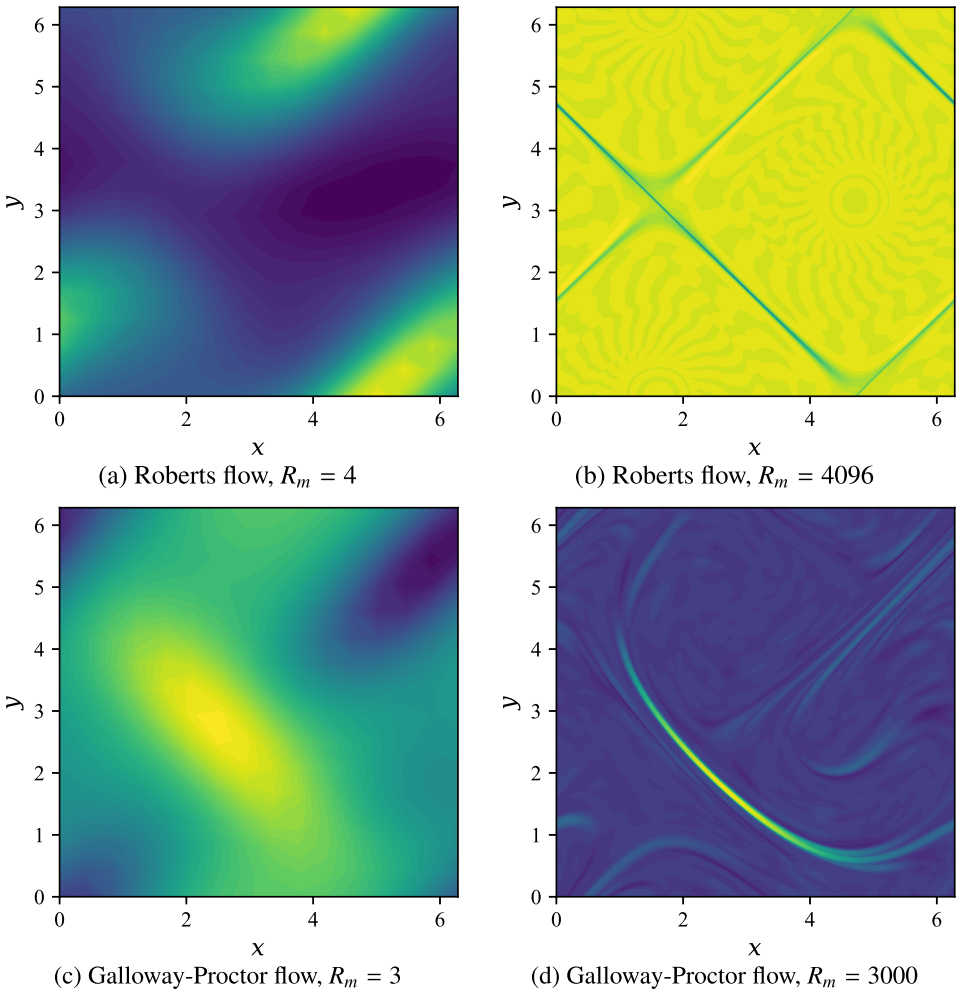


Fig. 1. Contour plots of the y-component of the magnetic fields for the labelled flows at time $T = 50$. The left hand side shows the low R_m field whilst the right hand plots show the high R_m fields. Much finer structures are apparent in the high R_m cases, due to the $R_m{}^{-1/2}$ scaling of the spatial structures. The Galloway Proctor field shows more spatial variability, due to the time dependence of the flow. The Roberts field stays effectively fixed in space, only varying in magnitude over the course of the simulation, whilst the morphology and magnitude of the Galloway Proctor field changes over time.

$$U_{n+1}^{k+1} = \mathcal{G}(t_{n+1}, t_n, U_n^{k+1}) + \mathcal{F}(t_{n+1}, t_n, U_n^k) - \mathcal{G}(t_{n+1}, t_n, U_n^k). \tag{8}$$

Computing the fine method can be parallelized across time slices. However, the correction from the coarse propagator has to be computed in serial, propagating the Parareal correction throughout the time domain.

3.1. Convergence and performance

As $k \to N_p$, Parareal converges to the solution that would have been obtained through the use of the fine method in serial over the time domain $[0, T]$ [30]. Achieving speed up, however, is dependent upon converging to an acceptable tolerance *tol* within a much smaller number of iterations than N_p. We test for convergence by measuring the relative two-norm defect at the last time slice between successive iterations

$$\sigma = \frac{\left\| U_{N_p}^k - U_{N_p}^{k-1} \right\|_2}{\left\| U_{N_p}^k \right\|_2}. \tag{9}$$

We denote as k_{con} the number of iterations required until $\sigma \leq tol$. Speed up s of Parareal can be estimated as

$$s = \left[\left(1 + \frac{k_{con}}{N_p} \right) \frac{R_c}{R_f} + \frac{k_{con}}{N_p} \right]^{-1}, \tag{10}$$

where R_c is the runtime of the coarse method over one time slice and R_f is the runtime of the fine method over one time slice. Speed up is bounded by

$$s \leq \min\left\{\frac{N_p}{k_{con}}, \frac{R_f}{R_c}\right\}, \tag{11}$$

[30]. The bound illustrates the trade off that needs to be optimised to gain optimal performance with Parareal. On the one hand, the ratio between the run times of the fine and coarse solvers should be large, so that the second bound is high. This can be achieved by using a very coarse and cheap $\mathcal{G}$. However, the method also needs to converge in a small number of iterations to ensure that the first bound is high. A less accurate coarse solver will typically require a larger number of iterations to converge. Finding a good compromise between these two competing factors is key to gaining high speed up with Parareal. Parallel efficiency is defined as

$$\epsilon = \frac{s}{N_p} = \frac{\text{parallel speed up}}{\text{number of processors}}, \tag{12}$$

where ideal scaling $s = N_p$ would give an efficiency of one. Parareal has an efficiency bound of $1/k_{con}$, which highlights the need to keep the number of iterations low. The need to test for convergence between two successive iterations of Parareal means there is an effective limit of $1/2$ for parallel efficiency when using Parareal.

3.2. Coarse solvers

A number of strategies have been proposed to design coarse solvers for Parareal. The most simple to implement is an increase in time step size for the coarse solver compared with the fine solver, whilst leaving the timestepping method and spatial discretization unchanged [20,37]. A second strategy is to use an implicit time stepping method with large step size for the coarse solver, with explicit time stepping for the fine solver [38]. Further strategies to find a coarse solver include simplifying the physics so that a less complicated model can be used [39–41]. Another strategy to be considered is to use a reduced spatial resolution, along with a larger time step [42]. Using this strategy, the method of interpolation from coarse to fine grids has been found to be important to Parareal convergence [43]. In section 5.2 we explore different options in the context of the kinematic dynamo problem.

4. Implementation

This work was carried out using the Dedalus [35] open source, spectral solver. We use a collocation-based pseudo-spectral method. Spectral methods benefit from spectral convergence: for a sufficiently smooth solution, the error of the method is $\mathcal{O}[(1/N_s)^{N_s}]$ for a discretisation with N_s collocation points. This compares favourably with a standard finite difference discretisation, which has error $\mathcal{O}[(1/N)^p]$ for a pth order method. Because of the periodic domain, Fourier bases are used. Dedalus uses the FFTW library to perform fast parallel transforms between real and spectral space and parallelizes in space over $n-1$ dimensions of an n dimensional domain using the mpi4py library [44].

Dedalus offers a number of different time stepping methods up to 4th order, including multi-step and Runge Kutta implicit/explicit (IMEX) methods [45–47]. Because multi-step methods in Parareal require restarting in every time slice and every iteration, we focus on Runge-Kutta methods. For optimal serial performance, we rely on IMEX Runge-Kutta methods [45]. Here, we integrate the terms $(\boldsymbol{u}\cdot\nabla)\boldsymbol{B}$ and $(\boldsymbol{B}\cdot\nabla)\boldsymbol{u}$ explicitly, whilst the diffusion term is treated implicitly. Below, we compare the IMEX Runge-Kutta methods RK111 (1 implicit stage, 1 explicit stage, 1st order), RK222 (2 implicit stages, 2 explicit stages, 2nd order), and RK443 (4 implicit stages, 4 explicit stages, 3rd order) to determine the most efficient serial fine solver against which to compare Parareal.

Simulations are carried out in a periodic domain with side length 2π. The spatial resolution is measured in terms of the number of spectral modes in x and y, (N_x, and N_y respectively) for the Roberts flow, or number of modes in y and z, (N_y and N_z respectively) for the Galloway-Proctor flow. Since the domain is a square, we set $N_x = N_y = N$ for the Roberts flow and $N_y = N_z = N$ for the Galloway-Proctor flow in all examples, so that the total spatial problem size is N^2. The initial conditions were set to a random perturbation of magnitude 10^{-5} for B_x and B_y in the Roberts flow, and for B_y and B_z in the Galloway Proctor flow.

In the kinematic approach, the simulation time needs to be long enough that the largest growing mode can be found and a steady growth rate calculated. In this work, this time was found to be around 10-20 turnover time periods. However, in a fully dynamic simulation, much longer simulation times are required to reach a statistically steady state in both the fluid flow and the magnetic field. For example, Smith et al. [48] ran simulations until $T = 450$ to achieve a statistically steady flow. As the aim of the work is to inform future studies of dynamic simulations, we choose a longer time interval of 50 turnover times.

Parareal was implemented by splitting the MPI world communicator into a space communicator and a time communicator. The space communicator was utilised by Dedalus to parallelize in space, whilst the time communicator was used to communicate between time slices. Two instances of Dedalus were created on each time slice, one with the coarse resolution (N_C spectral modes in each direction) and time step Δt, and one with the fine resolution (N_F spectral modes in each direction) and time step δt. Interpolation from coarse to fine grids, and restriction from fine to coarse grids, were carried

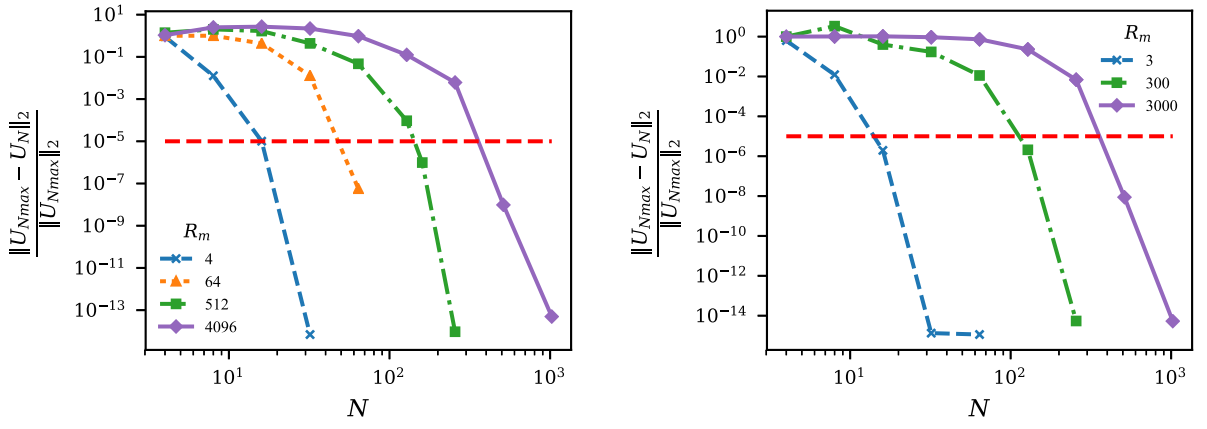
out using the Dedalus `set_scales(ratio)` method on the field objects, which allows efficient interpolation based on Fourier re-sampling.

4.1. Pseudo-code for the Parareal algorithm in Dedalus

```
// Initialization
space_communicator=MPI.COMM_WORLD.Split(time_slice, key);
time_commnicator=MPI.COMM_WORLD.Split(space_slice, key);
calculate time slice size, start and end of time slice for each time slice;
create 2 instances of dedalus solver on each time slice, using the space communicator, with same equations, but
 different time step sizes and resolutions;
set initial conditions of fine solver;
restrict initial conditions down to size of coarse solver, and set coarse solver initial conditions;

// Initial coarse run
for each time slice from 0 to T_end do
    if first time slice then
        get initial conditions
    else
        time_communicator.Recv(coarse_fields,source=time_communicator.rank-1);
        // next step gets fine solver ready for parareal iterations
        coarse_fields.set_scales(N_fine/N_coarse);
        fine_fields=np.copy(coarse_fields)
    end
    for i in range(N_time_steps_per_slice_coarse) do
        coarse_solver.step(coarse_time_step)
    end
    time_communicator.Send(coarse_fields,dest=time_communicator.rank+1);
end

// Beginning of Parareal iterations
while not converged do
    // Parallel step
    for i in range(N_time_steps_per_slice_fine) do
        fine_solver.step(fine_time_step)
    end
    // Serial Step
    for every time slice do
        if first time slice then
            get initial conditions
        else
            time_communicator.Recv(fine_fields,source=time_communicator.rank-1);
        end
        // Parareal correction
        fine_fields.set_scales(N_coarse/N_fine);
        coarse_fields=np.copy(fine_fields);
        for i in range(N_time_steps_per_slice_coarse) do
            coarse_solver.step(coarse_time_step)
        end
        carry out correction (new_coarse - old_coarse + new_fine);
        fine_fields = new_coarse_result - old_coarse_result + fine_fields;
        if not last time slice then
            time_communicator.Send(fine_fields,dest=time_communicator.rank+1);
        end
        save corrected solution;
        set time to t_start for each slice (coarse and fine);
    end
    if final time slice then
        check for convergence with previous solution;
    end
end
```



(a) Roberts Flow (b) Galloway Proctor Flow

Fig. 2. Graphs showing spatial convergence of the solvers for the two flows investigated. Spectral order of convergence is observed, with the decrease in error accelerating as the number of spectral modes, N, is increased. The line at 10^{-5} shows the required level of accuracy in the solution. The number of modes required for each Rm follows the predicted $Rm^{1/2}$ scaling.

```
// Saving state
Function Save_state():
    analysis=fine_solver.evaluator.add_file_handler(file_name,iter=1);
    analysis.set_num=time_communicator.rank analysis.add_system(fine_solver.state,layout='g');
    fine_solver.step();
    analysis.iter=np.inf;
return
```

5. Results

To ensure that reported speedups are a like-to-like comparison and meaningful, we need to make sure that the solution provided by Parareal is of the same accuracy as the fine solver used serially. Given the need for highly accurate results in dynamo simulations, we aim for an accuracy of 10^{-5}. Furthermore, we need to compare Parareal against an efficient fine solver – achieving speedup with Parareal when a much more efficient alternative to the fine solver exists would not provide convincing evidence of Parareal's usefulness. Therefore, we first find the optimal fine solver in Dedalus to deliver the required accuracy and then parallelise it with Parareal.

5.1. Fixing the fine solver

First, we need to fix the required spatial resolution for $\mathcal{F}$. Convergence in space was tested by running simulations at double the previous spatial resolution until the normalised L^2 difference between two solutions was $\sim 10^{-15}$. At this point, the error from the spatial discretisation is of the order of machine precision. We denote U_N as the solution vector containing b_x and b_y in real space with resolution N. The lower resolution solution was interpolated onto the same grid size as the fine solution using spectral interpolation so that the difference could be computed. The most highly resolved solution, U_{Nmax}, was then used as a reference solution to compute relative error of each U_N. The results are shown in Fig. 2, confirming the expected spectral convergence behaviour. For each R_m, we set N_F to the smallest value that gives a solution with error smaller than 10^{-5} (indicated by the dashed red line). Because higher magnetic Reynolds numbers produce smaller scale features, they require better spatial resolution to match our error tolerance.

Next, we fix the time stepping method and time step. Creating a reference solution with a temporal error of the order of machine precision proved to be unfeasible, due to computational constraints, especially in the higher R_m cases. Therefore, a result with error lower than 10^{-7} was used as a reference solution for setting δt. This is two orders of magnitude smaller than the desired result of 10^{-5} and should provide an accurate estimation of the error due to time-stepping. A comparison of the Runge-Kutta time steppers available in Dedalus is shown in Fig. 3. Because RK443 reaches the required tolerance of 10^{-5} with the smallest number of evaluations of the right hand side function, it is the most efficient choice. Similar results were found for other magnetic Reynolds numbers. We therefore use RK443 for the fine method $\mathcal{F}$ throughout this work.

Fig. 4 shows the dependence of the solution error on time step size for a range of R_m for the Roberts and Galloway-Proctor flows. All simulations were carried out with the optimal N_F found from the spatial resolution study (Fig. 2). We can see that smaller step sizes are required to meet a given level of accuracy for the Galloway-Proctor flow than for the Roberts flow. This is likely due to the Galloway-Proctor flow depending explicitly on time. In the case of the Roberts flow, a δt small enough to satisfy the stability requirements for a given N_F is sufficient to also satisfy the accuracy requirement of 10^{-5},

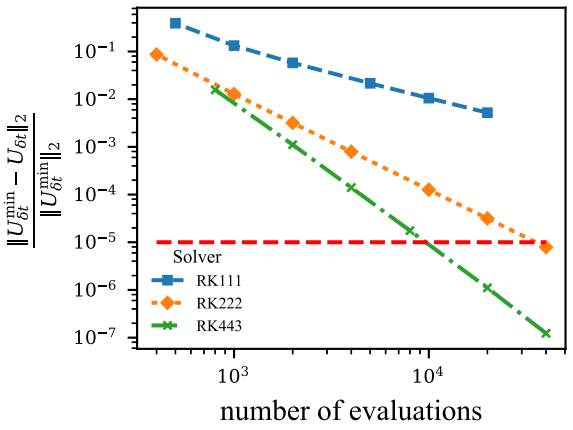


Fig. 3. Work required for different time stepping methods to obtain solutions of a certain accuracy, measured as the relative two-norm of each solution $U_{\delta t}$ and the solution obtained using the smallest timestep with the RK443 stepper ($U_{\delta t}^{\min}$). The number of evaluations required to compute from $T = 0$ to $T = 1$ are shown. This number depends on δt and the number of stages in each time-stepping method. For any degree of accuracy better than 10^{-2}, the RK443 time-stepper requires fewer evaluations than either RK111 or RK222. Results shown are for the Galloway-Proctor flow with $R_m = 300$, $N_F = 256$.

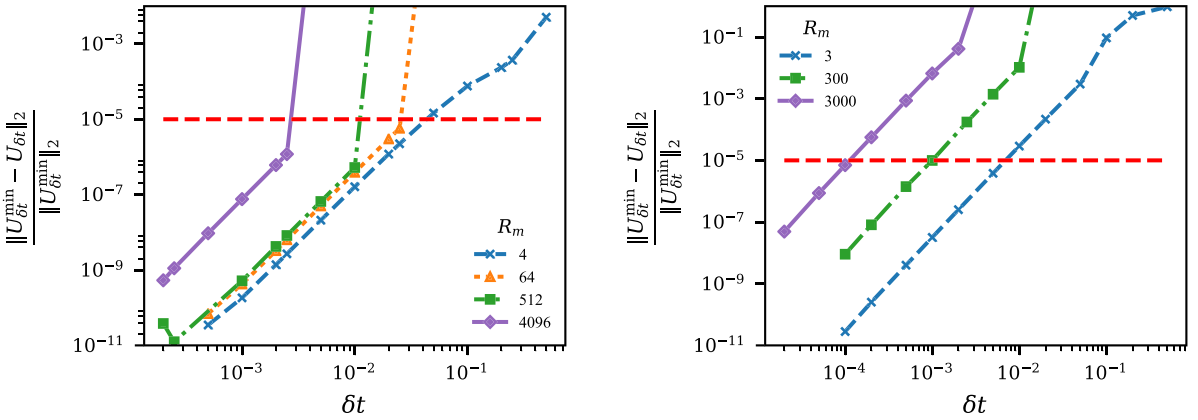
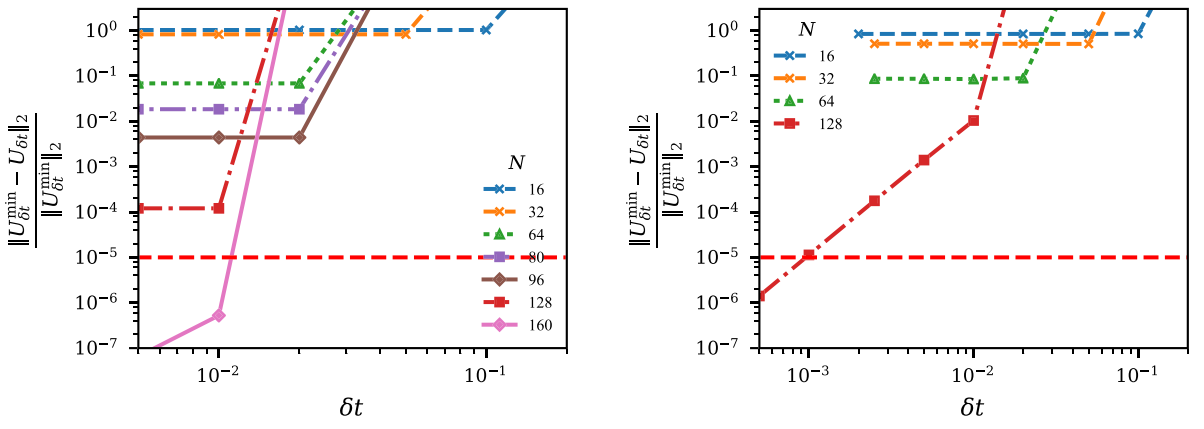


Fig. 4. Convergence with respect to time step size for the different flows and R_m simulated. The Roberts flow, which is independent of time, shows high accuracy for the highest stable time step for all simulations except $R_m = 4$. The Galloway Proctor flow has a larger error for the same time step size. This is believed to be because of the incorporation of time on the right hand side of the equations. Galloway-Proctor flows therefore require smaller time step sizes to reach the desired accuracy. Where the error goes past the top of the figure, the solver has diverged and is unstable for this time step size.

except for the most simple case of $R_m = 4$. This affects the performance of Parareal through the ratio of computational run times R_f/R_c because we have to use essentially the same time step for both the coarse and fine method. In contrast, for the Galloway-Proctor simulations, satisfaction of the stability requirement did not guarantee accuracy within the required tolerance, and a reduced δt must be used for the fine solver, leading to a better coarse-to-fine computation time ratio.

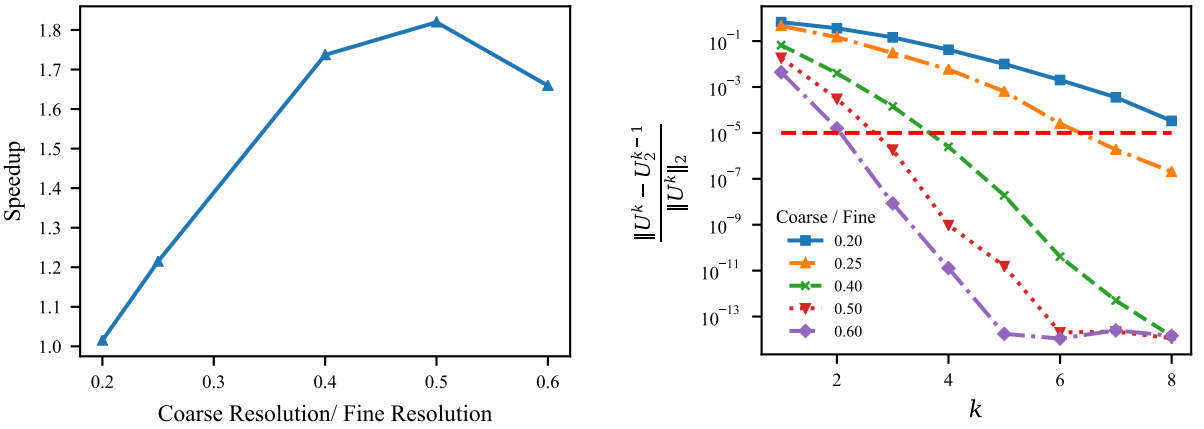
5.2. Fixing the coarse solver

The N_F and δt determined in Section 5.1 are used in the fine solver. We now discuss the different possibilities available for choosing a coarse solver. Using the same spatial resolution with $\Delta t > \delta t$ was not suitable for the Roberts flow, as δt was the largest stable time step for a given resolution. It was also unsuitable for the Galloway-Proctor simulations, as the ratio of $\Delta t/\delta t$ was not large enough to give meaningful speedup. Use of a fully implicit coarse solver was rejected, as the spectral spatial discretisation meant that a dense matrix solve would be required at each time step. This large increase in computational complexity would reduce the difference in computations required between the fine and coarse solvers, leading to smaller speed ups. There was little scope to attempt to use reduced physics in this study, as we are already considering the simplest form of dynamo problem. However, this strategy may be useful in further work on a non-linear dynamo. Coarsening in both space and time was found to be the most promising strategy. As time step stability is linked to spatial resolution, reducing spatial resolution allows a larger time step to be taken, even where the fine solver is at the largest stable time step. This means that $N_C < N_F$ and $\Delta t > \delta t$, opening up the possibility of a large difference in computational complexity between the coarse and fine solvers. However, too aggressive coarsening will lead to a very



(a) Roberts Flow, $R_m = 512$ (b) Galloway-Proctor Flow, $R_m = 300$

Fig. 5. Error vs. time step size for a range of spatial resolutions (N) for the Roberts and Galloway-Proctor flows. Accuracy is constrained by spatial resolution, until the finest resolution is reached in each case. As the resolution reduces, the largest stable time step increases as expected. Error increases above 10^1 (off the top of the graph) indicate that the method has become unstable at that time step. Accuracy for a given resolution/time step size is higher for the Roberts flow than the Galloway-Proctor flow.



(a) Speedup (b) Convergence

Fig. 6. (a) Speedup vs ratio of coarse, fine spatial resolutions for Roberts flow, with $R_m = 512$, $N_F = 160$, and $\delta t = 10^{-2}$. Long run times are found for very low coarse resolutions as the estimated solution is not accurate enough to allow quick convergence. As N_C increases, the run time reduces due to the reduced number of Parareal iterations required to converge. The best performing coarse solver has a resolution of $0.5N_F$. Further increasing the resolution of the coarse solver increases the complexity of the coarse solver to a level close to that of the fine solver, reducing any speed up possible. (b) Graph showing how defect to previous solution changes with number of Parareal iterations for different resolutions of the coarse solver. Very low resolutions results in Parareal taking many iterations to converge, reducing opportunity for speed up. High resolutions show quicker convergence.

inaccurate coarse solver and slow convergence. The most efficient amount of spatial coarsening was studied by carrying out Parareal simulations with a wide range of coarse method spatial resolution.

Simulations were carried out for the Roberts flow with $R_m = 512$. This is moderately high, whilst allowing relatively modest compute resources to be used. The fine solver parameters were fixed, with $N_F = 160$, and $\delta t = 10^{-2}$, while the coarse step Δt was set to the highest stable step for the given N_C. This was found by estimating the error at different time steps for each resolution, as shown in Fig. 5a. A similar study was carried out for the Galloway-Proctor flow (Fig. 5b). The number of Parareal time slices N_P was fixed at 10. Fig. 6 shows that the peak speed up is acquired when $N_C = 0.5N_F$. When $N_C < 0.5N_F$, the speed up is reduced by the extra number of Parareal iterations required to converge, and when $N_C > 0.5N_F$, the difference in computational complexity between the coarse and fine solvers is insufficient.

5.3. Scaling results

Scaling tests were carried out for both the Roberts flow and the Galloway-Proctor flow. Simulations of the Roberts flow were carried out on the ARC 3 HPC facility at the University of Leeds, made up of Intel Xeon E5-2650v4 (Broadwell) CPUs, with a total of 6,048 cores. Simulations of the Galloway-Proctor flow were carried out on the ARCHER HPC facility, made up of Intel Xeon E5-2697v2 (Ivy Bridge) CPUs, with a total of 109,056 cores.

Table 1
Parameters for simulations. R_m: magnetic Reynolds number, k_z: wave number in z co-ordinate, k_x: wave number in x co-ordinate, N_F: number of modes in fine propagator, N_C: number of modes in coarse propagator, δt: time step for fine propagator, Δt: time step for coarse propagator, Growth rate indicates growth rate of the magnetic field.

Flow	R_m	k_z	k_x	N_F	N_C	δt	Δt	Growth rate
Roberts	512	2.87		160	80	10^{-2}	2×10^{-2}	0.11
	4096	7.5		512	256	2.5×10^{-3}	5×10^{-3}	0.097
Galloway-Proctor	3		0.57	16	8	5×10^{-3}	10^{-1}	0.15
	300		0.57	128	64	10^{-3}	2×10^{-2}	0.3
	3000		0.57	512	256	10^{-4}	5×10^{-3}	0.3

A range of R_m were simulated to see the effect on Parareal performance (see Table 1). Scaling performance was compared with pure spatial scaling of the Dedalus solver. Fully parallel in space and in time simulations were also carried out in order to show how Parareal can increase scalability beyond the saturation of spatial scaling. Validation was carried out by comparing computed growth rates with those found in the literature. Growth rates for the Roberts dynamo were found to be consistent with those obtained by Plunian and Radler [10], which were reported to 2 significant digits, with the peak growth rate for each R_m occurring at the same k_z wave number reported. Growth rates for the Galloway-Proctor simulations were consistent with those found by Charbonneau et al. [49], with the peak growth rate occurring at a k_x of 0.57. The Galloway-Proctor flow had the correct behaviour in terms of growth rate for large R_m, with the growth rate staying positive, showing the expected fast-dynamo behaviour.

5.4. Roberts flow

Scaling results for the Roberts flow are shown in Fig. 7 for $R_m = 512$ (upper figures) and $R_m = 4096$ (lower figures). For both values of R_m, both parallel scaling and efficiency in space are superior to Parareal at low processor counts. As expected, spatial scaling is better for the $R_m = 4096$ case with higher spatial resolution, due to higher workload per processor. While Parareal alone is not competitive, in both cases a combined space-time parallelization generates slightly more speedup than a pure spatial parallelization. The theoretical maximum efficiency for Parareal is 1/3, indicated by a horizontal dashed line, due to the simulation requiring three iterations to converge. However, because of the relatively expensive coarse solver, Parareal's observed efficiency is mostly substantially lower. As the efficiency of the combined space-time parallelisation is the product of the parallel in space efficiency and the parallel in time efficiency, it is low for high numbers of processors because of the low efficiency of Parareal for the Roberts flow. Despite the larger overall speedup, with efficiencies below 0.1, space-time parallelization using Parareal may not be particularly attractive.

5.5. Galloway Proctor flow

Results for Rm= 3, 300 and 3000 for the Galloway Proctor flow are shown in Fig. 8. Performance of Parareal is much better than for the Roberts flow. Parareal speed up is competitive with spatial parallelisation at a relatively low number of processors. In all of these cases, the number of iterations required to converge was three, so that the efficiency is bounded by 1/3, and the efficiency of Parareal stays close to this bound over a range of N_p, and does not fall much as the number of processors increases. The poor performance of Dedalus in parallelising the Rm=3 case is attributable to the fact that there are only 16^2 spectral modes. In the $R_m = 3000$ case using 32 processors in space, the results show that speed up above that of spatial parallelisation alone is possible, with an efficiency around 0.16. In all cases, Parareal has not yet reached saturation in its scaling performance, and has almost ideal scaling behaviour, except for a constant offset due to the bounds on Parareal scaling. This is shown in Fig. 9, where the efficiency of the method is tracked over the different R_m. Efficiency is close to the bound of 1/3, and Parareal efficiency does not fall with increasing R_m. Pure Parareal efficiency was estimated in the $R_m = 3000$ case by dividing by the efficiency of the spatial parallelisation found for that particular N_S (32). In that case, total parallel efficiency is lower than for the other Galloway-Proctor cases due to the combination of the spatial and temporal parallelisation, and is approximately the product of the two, as expected. This reduction in overall efficiency is unavoidable, as parallel in space is more efficient than Parareal for lower numbers of processors, and Parareal only becomes competitive after spatial efficiency falls away. Also shown on this Figure are the efficiencies obtained at different R_m for the Roberts flow, highlighting the difference in performance of the method for the two cases.

6. Conclusions

The Parareal algorithm has been found to offer parallel speed up for kinematic dynamo simulations beyond what can be achieved through spatial parallelisation alone. In the case of the simpler steady Roberts dynamo, the speed up is modest and parallel efficiencies are low. Here, owing to the steady nature of the imposed velocity, the difference in computational complexity of the coarse and fine methods is found to be too small for good performance of Parareal. The issue was that the

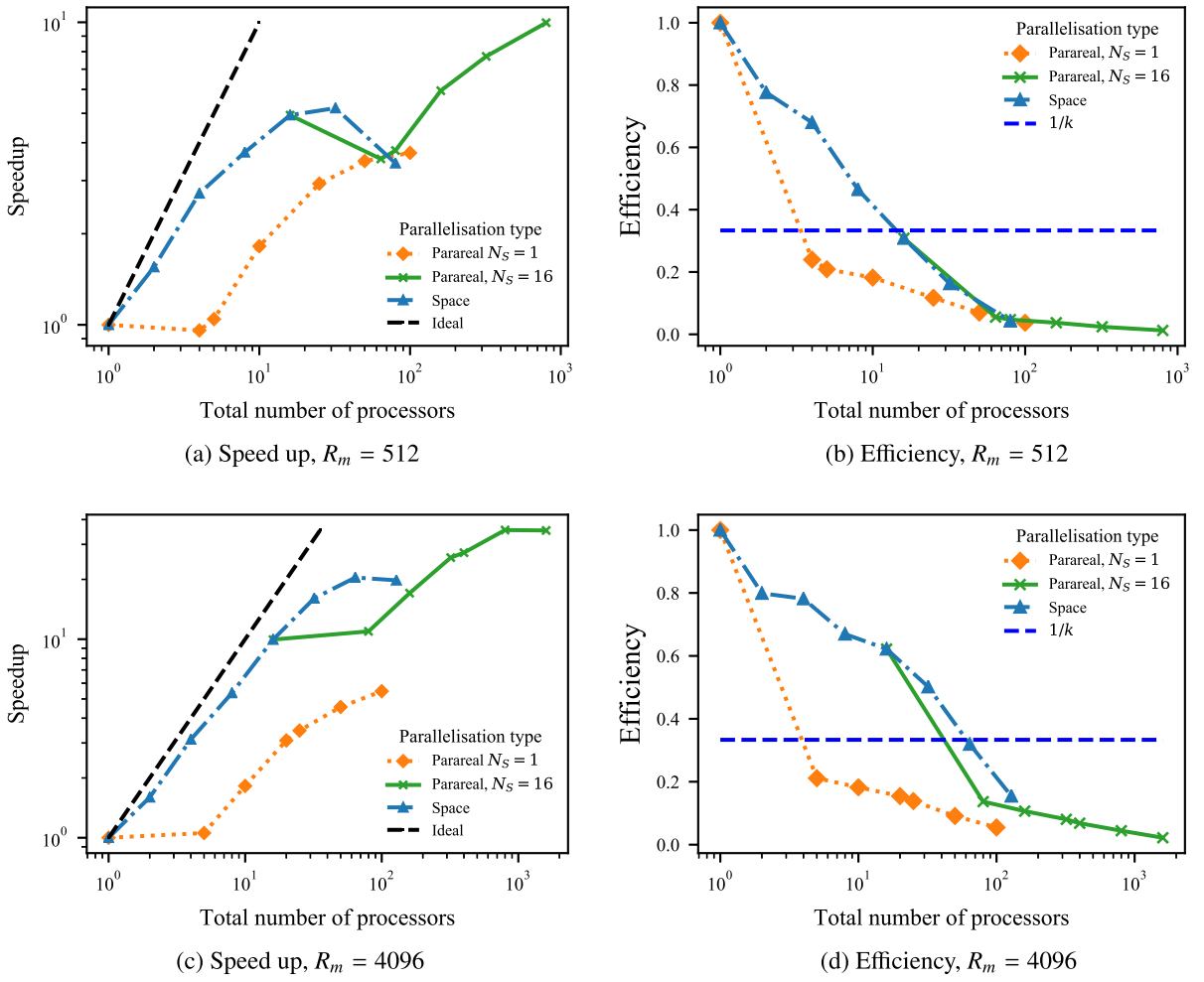


Fig. 7. Speed up (a), (c) and parallel efficiency (b), (d) of the Parareal method compared with spatial parallelisation for simulations of Roberts flow with Rm=512 (a), (b) and R_m=4096 (c), (d). Total number of processors is calculated as number of processors in space (N_S) multiplied by number of processors for Parareal (N_P). Speed up and efficiency are both poor for low numbers of processors for Parareal, but as the parallelisation in space saturates, further gains can be made from Parareal, although they are small. Parareal does not offer any gain over parallelisation in space until parallel efficiency is less than 0.1, and does not come close to the theoretical maximum of $1/k$, where k is number of Parareal iterations.

time step size was not a limiting factor on the accuracy of the fine solver; as long as the time step was stable, it was within the accuracy required. Therefore, there was little room to use a coarser resolution for the coarse propagator in Parareal.

Performance for the time-dependent Galloway-Proctor flow was better and the efficiency stayed close to the theoretical limit over a wide range of magnetic Reynolds numbers, while scaling well to large numbers of processors. In this problem, since evolution of the magnetic field depends explicitly on the current time, the accuracy of the solution depends more on the size of the time step. This means that a time step in the coarse solver much larger than that of the fine step is possible, allowing for better Parareal performance.

To summarise the difference in performance of the two problems, the Roberts flow fine solver is accurate with relatively high timestep size (close to the stability limit). This means that the coarse solver timestep is only a little larger ($\sim$ 2 times) than the fine timestep. In the Galloway-Proctor flow, the accuracy of the fine solver requires a timestep lower than the stability limit —— due to the time dependence of the driving flow. Therefore, the coarse solver, which only has to be stable, can have a timestep $\sim$20 times larger than the fine solver in this case. This leaves more potential for speedup.

Fully coupled dynamic dynamo simulation is complicated, and has non-linear dependencies, and so the accuracy of the fine solver is expected to behave more like the Galloway Proctor flow. Therefore, the good performance of Parareal for the Galloway-Proctor flow suggests that good performance can be expected also for more complex dynamos. This paper therefore gives an interesting example of how studying too simple a problem can lead to an overly negative assessment of the performance of Parareal.

The parallel efficiency of the Parareal algorithm applied to the Galloway Proctor dynamo is close to the theoretical maximum of $1/k_{con}$. This means that the overheads due to communication are small, in comparison to the serial cost of the coarse method, pointing to an efficient implementation of the algorithm. Performance of the algorithm when applied to

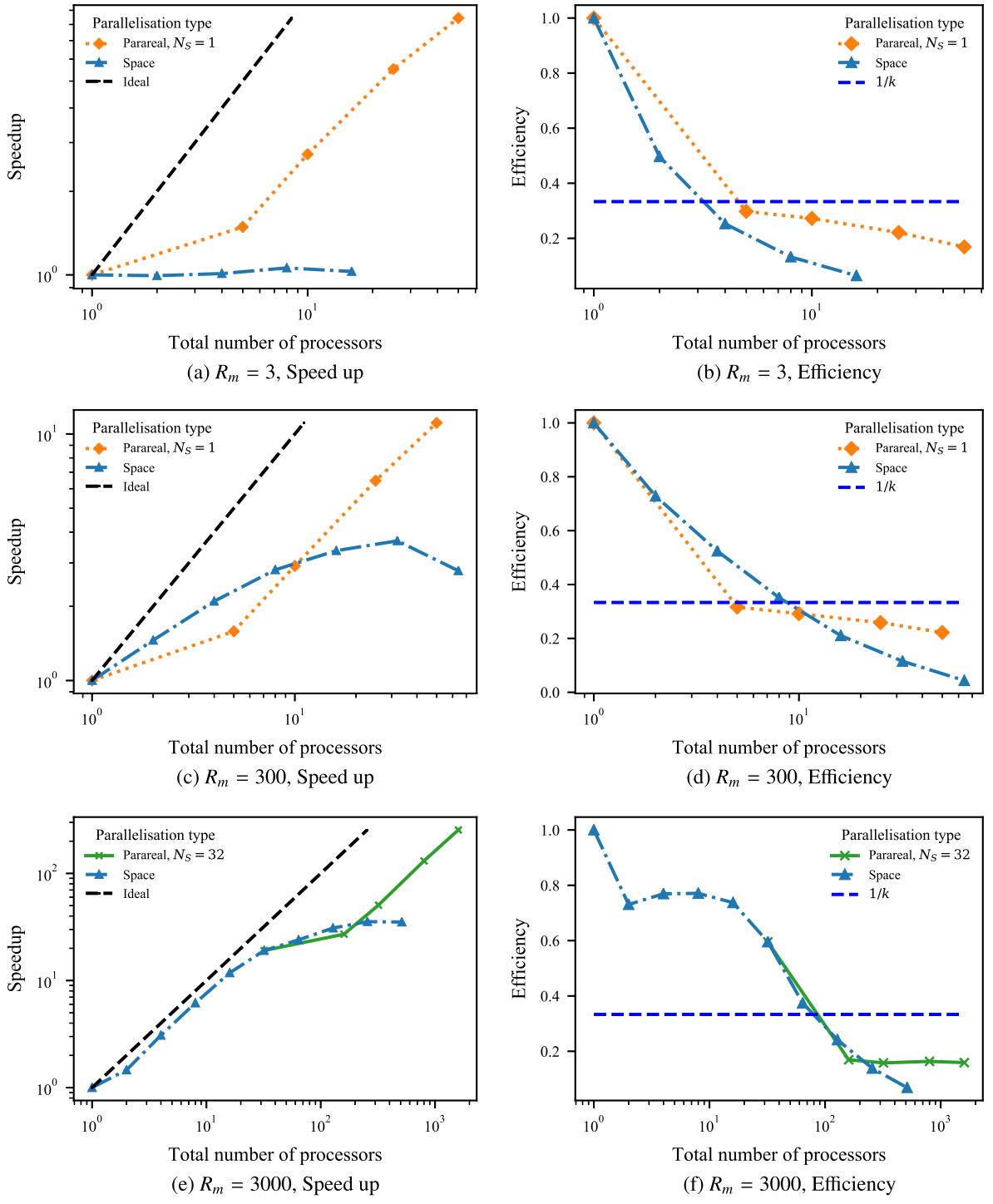


Fig. 8. Speed up and parallel efficiency of the parareal method compared to parallelisation in space for R_m of 3, 300, and 3000, Galloway Proctor flow. Total number of processors is calculated as number of processors in space (N_S) multiplied by number of processors for Parareal (N_P). In the case of R_m=3000, parareal simulations were carried out with 32 processors in space, as serial runs with one processor were time intensive. Spatial resolutions required were 16^2, 128^2, and 512^2 respectively. Results here are more promising than in the Roberts flow. Parareal becomes more efficient than spatial parallelisation for smaller processor numbers, and keeps higher efficiency for longer, closer to the theoretical maximum of $1/k$ (k: number of Parareal iterations). Scaling saturation for parareal has not been reached even at 1600 processors in the R_m=3000 case.

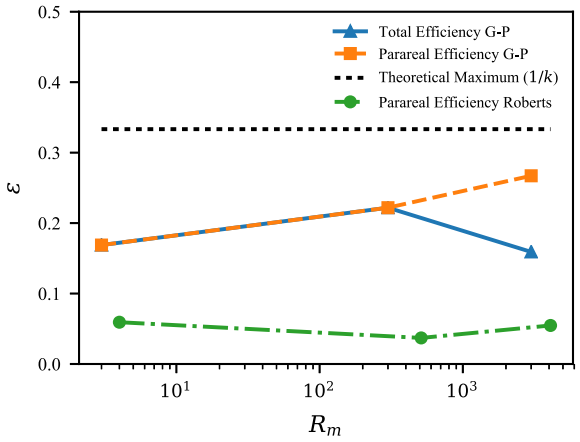


Fig. 9. Parallel efficiency vs R_m for Galloway-Proctor and Roberts dynamos. Galloway-Proctor results show higher efficiency than the Roberts flow. Parallel efficiency of the method does not appear to degrade with increasing R_m. There is a reduction for total efficiency for R_m=3000, however, this is due to a combination of the efficiency of the spatial parallelisation with the parareal efficiency. Efficiency of parareal alone is comparable to the efficiency of the lower R_m simulations in the Galloway-Proctor case.

this problem does not appear to degrade with increased R_m, as can be seen in Fig. 9. Performance has remained constant, with Parareal efficiency not much lower than $1/3$ for $R_m = 3$, $R_m = 300$ and $R_m = 3000$. This is noteworthy since highly advective problems are thought to cause problems with Parareal convergence, but this has not yet been found in the highly advective case with R_m up to $\sim 10^3$.

The results in this paper show that parallel in time methods can speed up kinematic dynamo simulations and are therefore worthy of further study. Better parallelization would enable the study of dynamos at larger magnetic Reynolds numbers by reducing simulation times by harnessing the ever growing number of available cores in HPC facilities. Current and future work involves extending our analysis to non-linear dynamic dynamo systems and the interaction of magnetic fields with convection. The development of successful parallel-in-time methods for these problems could allow the integration of geo- and astro-dynamo simulations in parameter regimes closer to reality than have been hitherto possible.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

A.T.C is supported by the Engineering and Physical Sciences Research Council (EPSRC) Centre for Doctoral Training in Fluid Dynamics (EP/L01615X/1). C.J.D is supported by a Natural Environment Research Council (NERC) Independent Research Fellowship (NE/L011328/1). D. R. thankfully acknowledges support from grants EPSRC EP/P02372X/1 “A new algorithm to track fast ions in fusion reactors” and NERC NE/R008795/1 “Parallel Paradigms for Numerical Weather Prediction”. S.M.T is supported by a Leverhulme Fellowship and by funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement no. D5S-DLV-786780). This work used the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk) as well as ARC2 and ARC3, part of the High Performance Computing facilities at the University of Leeds. Figures were produced using Matplotlib [50]. All codes used to create the results in this manuscript are publicly available on GitHub at [51].

References

[1] P.H. Roberts, A.M. Soward, Dynamo theory, Annu. Rev. Fluid Mech. 24 (1992) 459–512.
[2] N. Weiss, Dynamos in planets, stars and galaxies, Astron. Geophys. 43 (2002) 3–9.
[3] H.K. Moffatt, Field Generation in Electrically Conducting Fluids, Cambridge University Press, Cambridge, London, New York, Melbourne, 1978.
[4] P.H. Roberts, An Introduction to Magnetohydrodynamics, vol. 6, Longmans London, 1967.
[5] C.J. Davies, D. Gubbins, P.K. Jimack, Scalability of pseudospectral methods for geodynamo simulations, Concurr. Comput. 23 (2011) 38–56.
[6] T.G. Cowling, The magnetic field of sunspots, Mon. Not. R. Astron. Soc. 94 (1933) 39–48.
[7] E.C. Bullard, H. Gellman, Homogeneous dynamos and terrestrial magnetism, Philos. Trans. R. Soc. Lond. A 247 (1954) 213–278.
[8] G.O. Roberts, Dynamo action of fluid motions with two-dimensional periodicity, Philos. Trans. R. Soc. Lond. A 271 (1972) 411–454.
[9] D.J. Galloway, M.R. Proctor, Numerical calculations of fast dynamos in smooth velocity fields with realistic diffusion, Nature 356 (1992) 691.
[10] F. Plunian, K.-H. Radler, Harmonic and subharmonic solutions of the Roberts dynamo problem. Application to the Karlsruhe experiment, arXiv preprint arXiv:0905.0847, 2009.
[11] B. Knaepen, R. Moreau, Magnetohydrodynamic turbulence at low magnetic Reynolds number, Annu. Rev. Fluid Mech. 40 (2008) 25–45.

[12] M. Kono, P.H. Roberts, Recent geodynamo simulations and observations of the geomagnetic field, Rev. Geophys. 40 (2002) 4-1.
[13] M. Ossendrijver, The solar dynamo, Astron. Astrophys. Rev. 11 (2003) 287–367.
[14] I. Klapper, L.-S. Young, Rigorous bounds on the fast dynamo growth rate involving topological entropy, Commun. Math. Phys. 173 (1995) 623–646.
[15] S.M. Tobias, F. Cattaneo, Shear-driven dynamo waves at high magnetic Reynolds number, Nature 497 (2013) 463–465.
[16] C.A. Jones, Course 2 dynamo theory, Les Houches 88 (2008) 45–135.
[17] H. Matsui, E. Heien, J. Aubert, J.M. Aurnou, M. Avery, B. Brown, B.A. Buffett, F. Busse, U.R. Christensen, C.J. Davies, et al., Performance benchmarks for a next generation numerical dynamo model, Geochem. Geophys. Geosyst. 17 (2016) 1586–1607.
[18] N. Schaeffer, D. Jault, H.-C. Nataf, A. Fournier, Turbulent geodynamo simulations: a leap towards earth's core, Geophys. J. Int. 211 (2017) 1–29.
[19] P.D. Mininni, D. Rosenberg, R. Reddy, A. Pouquet, A hybrid mpi–openmp scheme for scalable parallel pseudospectral computations for fluid turbulence, Parallel Comput. 37 (2011) 316–326.
[20] R. Croce, D. Ruprecht, R. Krause, Parallel-in-space-and-time simulation of the three-dimensional, unsteady Navier-Stokes equations for incompressible flow, in: Modeling, Simulation and Optimization of Complex Processes-HPSC 2012, Springer, 2014, pp. 13–23.
[21] M.J. Gander, 50 years of time parallel time integration, in: Multiple Shooting and Time Domain Decomposition Methods, MuS-TDD, Heidelberg, May 6-8, 2013, vol. 9, 2015, p. 69.
[22] J.-L. Lions, Y. Maday, G. Turinici, Résolution d'edp par un schéma en temps pararéel, C. R. Acad. Sci., Ser. 1 Math. 332 (2001) 661–668.
[23] M.L. Minion, A hybrid parareal spectral deferred corrections method, Commun. Appl. Math. Comput. Sci. 5 (2010) 265–301.
[24] J. Cortial, C. Farhat, A time-parallel implicit method for accelerating the solution of non-linear structural dynamics problems, Int. J. Numer. Methods Eng. 77 (2009) 451–470.
[25] M.J. Gander, S. Guttel, Paraexp: a parallel integrator for linear initial-value problems, SIAM J. Sci. Comput. 35 (2013) C123–C142.
[26] P.F. Fischer, F. Hecht, Y. Maday, A parareal in time semi-implicit approximation of the Navier-Stokes equations, in: Domain Decomposition Methods in Science and Engineering, Springer, 2005, pp. 433–440.
[27] D. Samaddar, D. Coster, X. Bonnin, C. Bergmeister, E. Havlícková, L.A. Berry, W.R. Elwasif, D.B. Batchelor, Temporal parallelization of edge plasma simulations using the parareal algorithm and the solps code, Comput. Phys. Commun. 221 (2017) 19–27.
[28] G. Bal, Y. Maday, A "parareal" time discretization for non-linear pde's with application to the pricing of an American put, in: Recent Developments in Domain Decomposition Methods, Springer, 2002, pp. 189–202.
[29] H. Samuel, Time domain parallelization for computational geodynamics, Geochem. Geophys. Geosyst. 13 (2012).
[30] M.J. Gander, S. Vandewalle, Analysis of the parareal time-parallel time-integration method, SIAM J. Sci. Comput. 29 (2007) 556–578.
[31] J. Steiner, D. Ruprecht, R. Speck, R. Krause, Convergence of parareal for the Navier-Stokes equations depending on the Reynolds number, in: Numerical Mathematics and Advanced Applications-ENUMATH 2013, Springer, 2015, pp. 195–202.
[32] D. Ruprecht, Wave propagation characteristics of parareal, Comput. Vis. Sci. 19 (2018) 1–17.
[33] H. De Sterck, R.D. Falgout, S. Friedhoff, O.A. Krzysik, S.P. MacLachlan, Optimizing MGRIT and parareal coarse-grid operators for linear advection, arXiv preprint, arXiv:1910.03726, 2019.
[34] X. Dai, Y. Maday, Stable parareal in time method for first- and second-order hyperbolic systems, SIAM J. Sci. Comput. 35 (2013) A52–A78.
[35] K.J. Burns, G.M. Vasil, J.S. Oishi, D. Lecoanet, B. Brown, Dedalus: flexible framework for spectrally solving differential equations, http://adsabs.harvard.edu/abs/2016ascl.soft03015B, 2016, http://dedalus-project.org.
[36] D. Galloway, U. Frisch, A numerical investigation of magnetic field generation in a flow with chaotic streamlines, Geophys. Astrophys. Fluid Dyn. 29 (1984) 13–18.
[37] E. Aubanel, Scheduling of tasks in the parareal algorithm, Parallel Comput. 37 (2011) 172–182.
[38] A. Blouza, B. Laurent, S.M. Kaber, Parallel in time algorithms with reduction methods for solving chemical kinetics, Commun. Appl. Math. Comput. Sci. 5 (2009) 241–263.
[39] L. Baffico, S. Bernard, Y. Maday, G. Turinici, G. Zérah, Parallel-in-time molecular-dynamics simulations, Phys. Rev. E 66 (2002) 057701.
[40] Y. Maday, G. Turinici, Parallel in time algorithms for quantum control: parareal time discretization scheme, Int. J. Quant. Chem. 93 (2003) 223–228.
[41] Y. Maday, Parareal in time algorithm for kinetic systems based on model reduction, in: High-Dimensional Partial Differential Equations in Science and Engineering, vol. 41, 2007, pp. 183–194.
[42] D. Ruprecht, Convergence of parareal with spatial coarsening, PAMM 14 (2014) 1031–1034.
[43] T. Lunet, J. Bodart, S. Gratton, X. Vasseur, Time-parallel simulation of the decay of homogeneous turbulence using parareal with spatial coarsening, Comput. Vis. Sci. 19 (2018) 31–44.
[44] L. Dalcín, R. Paz, M. Storti, Mpi for python, J. Parallel Distrib. Comput. 65 (2005) 1108–1115.
[45] U.M. Ascher, S.J. Ruuth, R.J. Spiteri, Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations, Appl. Numer. Math. 25 (1997) 151–167.
[46] P.R. Spalart, R.D. Moser, M.M. Rogers, Spectral methods for the Navier-Stokes equations with one infinite and two periodic directions, J. Comput. Phys. 96 (1991) 297–324.
[47] D. Wang, S.J. Ruuth, Variable step-size implicit-explicit linear multistep methods for time-dependent partial differential equations, J. Comput. Math. (2008) 838–855.
[48] S.G.L. Smith, S. Tobias, Vortex dynamos, J. Fluid Mech. 498 (2004) 1–21.
[49] P. Charbonneau, Solar and Stellar Dynamos, Saas-Fee Advanced Course 39 Swiss Society for Astrophysics and Astronomy, vol. 39, Springer Science & Business Media, 2012.
[50] J.D. Hunter, Matplotlib: a 2d graphics environment, Comput. Sci. Eng. 9 (2007) 90–95.
[51] A. Clarke, Parareal-dynamo, https://doi.org/10.5281/zenodo.2554026, 2019.