Solving chaotic ODEs with time-parallel algorithms

Giancarlo Antonino Antonucci

St Anne's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2025

Acknowledgements

First, I must thank my supervisor, Raphael Hauser, for his guidance and sharp insights, and for pointing me towards the most interesting, and at times thorny, research directions. His encouragement kept this work going.

I am also deeply grateful to my industrial supervisors: Debasmita Samaddar, for taking on this project and setting its initial course, and James Buchanan, for taking over and seeing it through to the end. Their extensive knowledge of fusion research was a constant source of drive and perspective.

A big thank-you goes to my InFoMM cohort. This journey would have been far duller, and far harder, without the countless chats we shared.

To my parents and my brother: thank you for your unwavering support and for always being there. I am forever grateful to you all.

To my wife, Giulia: thank you for your endless patience and for standing by me through the hardest times. I truly could not have done this without you.

Finally, to anyone I forgot to mention, thank you all the same. Luckily for me, the odds of you ever reading this thesis are comfortably low.

Abstract

This thesis studies the numerical integration of chaotic dynamical systems over long time spans, a core problem in fields such as fusion energy research, climate modelling, and fluid dynamics. Standard time-parallel algorithms work well for linear systems or over short time intervals, but perform poorly on long periods dominated by chaos.

To address this problem, we develop a framework for finite-time convergence based on the theory of contraction mappings. This framework uses an *outer–inner ball analysis* to give explicit bounds on the number of iterations needed to reach a chosen tolerance, thereby going beyond traditional asymptotic results. We also define a *proximity function* to act as a specialized stopping rule that enforces short-term accuracy while accommodating the natural long-term divergence of trajectories.

Next, we introduce the *moving-window* (MoWi) algorithm for the robust long-term integration of chaotic systems. MoWi splits the overall time domain into a sequence of overlapping windows. It then runs a time-parallel subroutine within each window. Through our framework, a rigorous *tracking analysis* shows that the starting guess for each window lies inside the outer ball of convergence of the time-parallel subroutine. This guarantees that the subroutine converges within a fixed number of iterations.

Lastly, we propose three adaptive variants (AMoWi) to handle systems with time-varying dynamics. These variants dynamically adjust the window length, the shift between windows, and the proximity function weights. We test these methods against the Lorenz, Lorenz–96, and Kuramoto–Sivashinsky equations. The results show that MoWi and its variants can outperform standard time-parallel solvers, ensuring scalable speedups while preserving the true statistical behaviour of the dynamics.

Contents

List of Figures

Chapter 1

Introduction

How can we efficiently run time-parallel methods on chaotic dynamical systems over long time domains? This thesis seeks to answer that question.

The main motivation behind our work is tokamak research. Here, predicting plasma behaviour relies heavily on simulating turbulent transport [35]. This is a hard task because it involves a wide range of timescales, stretching from picoseconds for electron gyro-frequencies to several seconds for macroscopic transport processes [96]. Furthermore, the underlying equations are highly nonlinear. As a result, today's software can take weeks or even months to simulate just a few hundred seconds of physical time, depending on resolution and physical fidelity. Physicists are therefore looking for new algorithms to speed up plasma simulations.

Although tokamak physics in particular motivates us, this work focuses on simpler benchmark models that show the same chaotic behaviour. We do this for two reasons. First, state-of-the-art nonlinear gyrokinetic codes are huge. Developing and testing a new time-parallel framework straight on top of such a complex architecture is impractical [119]. A simplified setting lets us build a clear proof of concept. We can then thoroughly analyse how time parallelization interacts with chaotic dynamics, identifying the key mathematical mechanisms that drive performance. Second, our methods are not limited to plasma physics. In fact, they can be applied to any field ruled by chaotic systems, such as weather forecasting and climate modelling [85].

The contributions of this thesis are as follows:

- In Chapter 3, we present a new convergence framework for time-parallel algorithms that applies to nonlinear chaotic systems. Using the theory of contraction mappings, we provide explicit convergence guarantees that hold in finite time, rather than just asymptotically. In Section 3.4, we introduce the *proximity function* as a new stopping rule to handle the sensitive dependence on

initial conditions that defines chaos. This sensitivity poses a major challenge for time-parallel methods like Parareal that seek to minimize discontinuities between time chunks.

- In Chapter 4, we introduce the *moving-window* (MoWi) algorithm, a new approach to integrating chaotic systems over long time domains. Because tracking a single chaotic trajectory over long periods is numerically ill-posed [13], MoWi sequentially samples distinct trajectories from the invariant distribution on the attractor [37], assuming one exists. It splits the timeline into overlapping windows and solves each with a time-parallel subroutine. The overlap provides a high-quality initial guess for the next window. This allows MoWi to capture the system's true statistical behaviour while speeding up the simulation. We also identify when this approach outperforms traditional implementations of the underlying time-parallel subroutine.

- In Chapter 5, we propose three adaptive MoWi variants (AMoWi-1, AMoWi-2, and AMoWi-3). They dynamically adjust their parameters to stay efficiency as the system's dynamics change. Local chaoticity and divergence rates can vary sharply across the attractor [97]. Our adaptive schemes are built to respond to these variations in real time.

We wrote all our tests in open-source Julia code [12].

The rest of this chapter introduces nuclear fusion, tokamaks, and turbulence. It then reviews the state of the art in time parallelization. We end with a summary of the thesis structure, showing how each chapter builds upon the last.

1.1 Motivation

Many regard nuclear fusion as a promising alternative to conventional energy sources. Unlike fission, it carries no risk of runaway meltdown, produces waste with short-lived radioactivity (with half-lives of roughly 100 years), and does not rely on enriched uranium or plutonium, which can be repurposed for nuclear weapons. Unlike solar or wind, fusion reactors can run continuously to provide a steady supply of energy. The leading design for a fusion reactor is a magnetic confinement device known as tokamak. Yet, making fusion work at scale has proven exceptionally challenging. Despite decades of research, no tokamak has yet reached *breakeven*, that is, the threshold where the energy produced by fusion reactions matches the external heating needed to sustain

the plasma. Even though the core physics is largely understood, major engineering and practical hurdles remain.

One major challenge is plasma turbulence. Microscale drift-type instabilities cause heat and particles to leak from the reactor core across the magnetic field lines much faster than expected, sharply lowering efficiency [35, Section 2.1]. Because the details of this process are still unclear, researchers rely on large-scale numerical simulations of the governing differential equations, typically using gyrokinetic models and magnetohydrodynamics [35, Section 2.2]. However, the highly nonlinear nature of plasma turbulence, combined with the huge gap in timescales, leads to very long wall-clock times for the simulations of interest [119]. To speed them up, researchers must turn to parallel computing.

Today, speedups in computing come mainly from increased parallelism, rather than higher clock frequencies, as shown in Figures 1.1 and 1.2. Single-thread performance has stalled due to the physical limits of silicon and heat dissipation. Therefore, performance gains now stem mostly from packing more cores onto chips [66]. This hardware shift is especially relevant as we enter the age of exascale supercomputing, where machines hold millions of cores [33]. This in turn has driven the development of scalable parallel algorithms that can better harness these resources to reduce the time-to-solution while keeping power consumption in check.

The first approach used to weave parallelism into the numerical solution of differential equations is to parallelize computations using space-decomposition methods [107]. Here, the physical domain is split into subdomains and each core takes up a specific patch. The computational workload per core scales with the patch's volume, whereas the communication overhead scales with its surface area. As the number of cores grows, the volume shrinks faster than the surface area. Once communication costs overtake computation, the parallel efficiency saturates, typically around 10^5 to 10^6 cores [124].

Modern supercomputers, however, have so many cores that space parallelization often saturates long before all parallel resources can be used. This is where time parallelization comes in. Over the past few decades, extensive research into time-parallel algorithms has shown them to yield significant speedups over traditional serial time-stepping [48]. By combining time and space parallelism, one can achieve full space-time parallelization. Hence, time-parallel algorithms are highly sought-after for large-scale simulations in fields such as weather and climate modelling [115], plasma physics [119], and fluid dynamics [24], where both fine time grids and massive parallelism are needed.

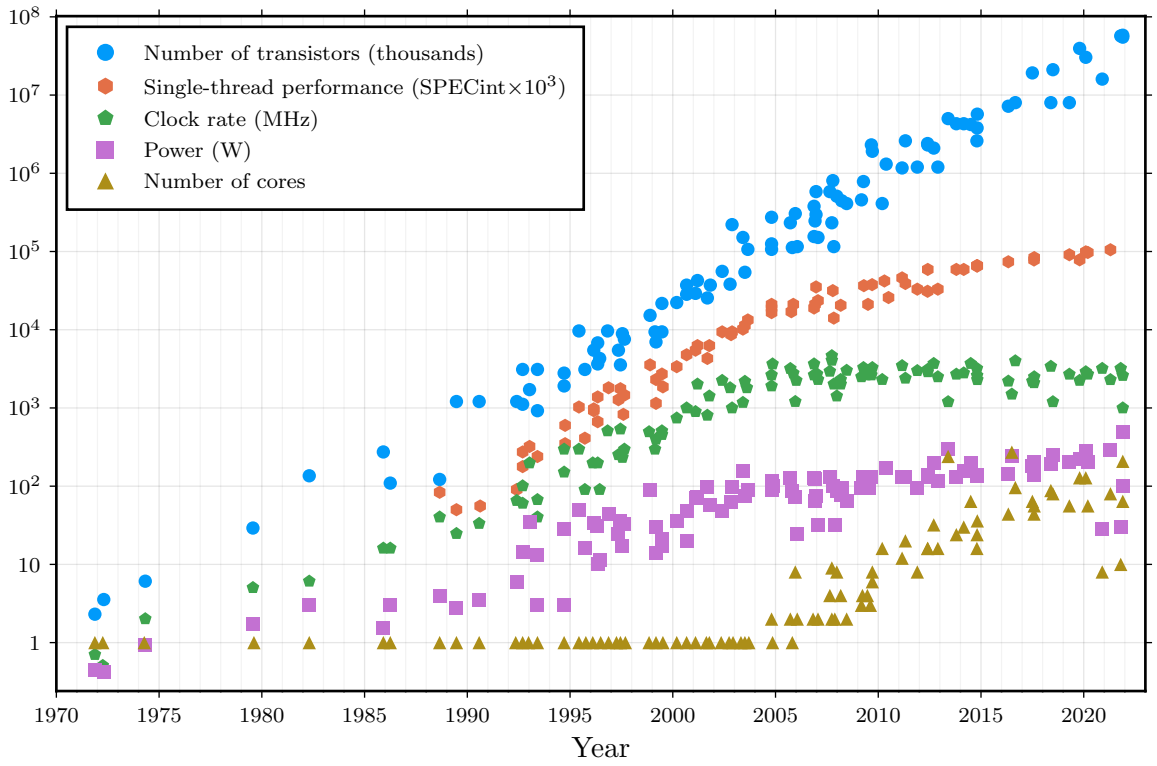

Figure 1.1: Technological advances of microprocessors over time. Data from Horowitz et al. (1986-2010) and Rupp (2011-2022) [114]. Single-thread performance is tracked using SPEC benchmark results. The breakdown of Dennard scaling (whereby power density stays constant as transistors shrink) around 2004, paired with the slowdown of Moore's law (whereby the number of transistors on a chip doubles roughly every two years) around 2015, has starkly hampered performance gains: yearly improvements fell from 52% between 1986 and 2003 to a mere 3.5% since 2015. Because voltage could no longer scale down with transistor size, higher clock speeds would have caused power consumption and heat to surge to unsustainable levels. Thus, around 2005, rising power and silicon costs led chip makers to adopt multicore architectures with lower clock rates per core, rather than chasing faster clock rates. This approach has since become the standard. For an in-depth look into the history of microprocessors, see [66].

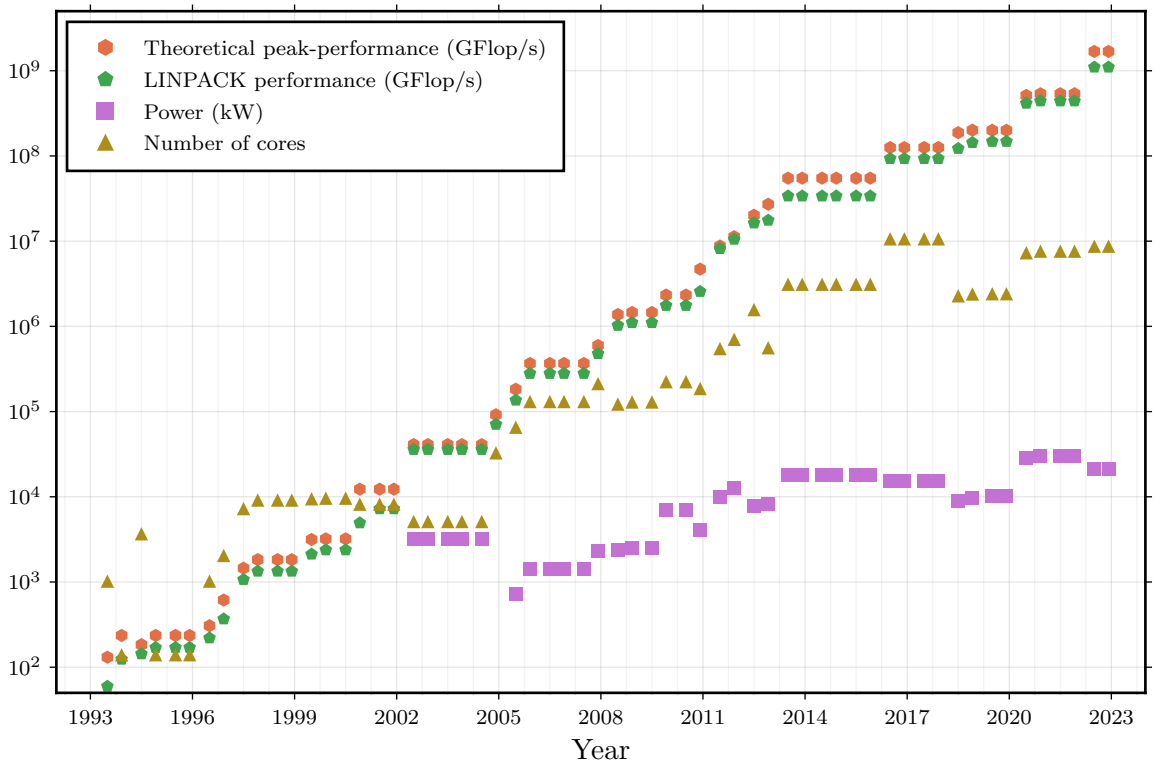

Figure 1.2: Technological advances in supercomputers over time. Data drawn from the TOP500 list (November 2022) [34, 131]. Theoretical peak performance is computed by multiplying the machine's clock rate (in cycles per second) by the number of floating-point operations per cycle. The LINPACK benchmark provides a truer gauge of sustained performance. It tracks how quickly a computer solves a dense system of linear equations using LU factorization with partial pivoting. Since the early 1990s, supercomputers have shifted towards massively parallel architectures, scaling from hundreds to millions of processing cores. Yet, a data centre can supply only so much power: the goal is to build the fastest machine possible within a fixed energy budget.

As we discuss in Section 1.5, many time-parallel algorithms exist, often built upon deeply different designs. Yet, they all share a common principle: by introducing redundant computations, the problem can be recast into a parallelizable form. Although this may seem counterintuitive, it lets us harness idle processing cores that would otherwise sit unused. In short, we trade extra work per core to shrink the overall wall-clock time. Crucially, the algorithmic steps are handled so that information flows forwards along the timeline, ensuring that the causality principle – that the solution at each time step depends on the outcome of the previous steps – is satisfied.

1.2 Nuclear fusion

Fusion is the physical process where two or more light atomic nuclei join to form a heavier one. The final products have a slightly smaller total mass than the starting nuclei. This mass defect is released as energy. The most promising candidates for fusion are deuterium and tritium [80], two isotopes of hydrogen: deuterium is abundant in the Earth's oceans [70] and tritium can be bred from lithium [141]. However, deuterium–tritium (D–T) fusion does not happen naturally on Earth. Extremely high temperatures and densities are necessary to overcome the strong electrostatic repulsion between the positively charged nuclei; so high, in fact, that the fuel breaks down into a plasma, that is, a fully ionized gas made up of bare ions and free electrons.

To grasp the scale of the physical quantities at play, take the following example from [140]. Let T_i and n_i be the temperature and density of the ion species i in the plasma. We define τ_E as the *energy-confinement time*, the mean time it takes for processes like radiation, conduction, convection, and neutron flux to drain the plasma of its stored energy. A larger τ_E means better confinement. The ultimate goal for a fusion reactor is *ignition*, the point at which the energy generated by fusion reactions balances all losses without any external heating. According to [140], for a product $n_i\tau_E \approx 1.5 \times 10^{20}\,\mathrm{s/m^3}$, the required ignition temperature is $T_i \approx 3.5 \times 10^8\,\mathrm{K}$: this is roughly twenty times hotter than the Sun's core. The first milestone, however, is *breakeven*, the point where heating by fusion-born alpha particles sustains the plasma without external input. By supplying external power to offset transport and radiation losses, breakeven can be achieved at the lower value $n_i\tau_E \approx 6 \times 10^{19}\,\mathrm{s/m^3}$. This is known as Lawson's criterion [80].

We need some form of confinement to hold these extreme temperatures (and pressures). In stars, gravity provides this confinement. On Earth, we must turn to other methods: one of the most promising is magnetic confinement, used in devices

such as tokamaks. These use strong magnetic fields to hold the plasma long enough for fusion reactions to happen.

1.3 Tokamaks

First developed in the late 1950s in the Soviet Union, tokamaks[1] are hollow, toroidal devices designed for the magnetic confinement of plasma. For details on how tokamaks work, see [137] and references therein. In short, a system of magnetic coils, known as toroidal and poloidal field coils, traps the plasma within the core of a tokamak. An electric current driven through the plasma yields both the starting heat to warm and ionize the deuterium and tritium gas, and the poloidal field needed for its confinement. Further heating can be achieved by feeding deuterium at high speed into the plasma and by inducing high-frequency currents at the resonance frequencies of the charged particles, where energy absorption is at its highest.

One of the main challenges to maintaining the plasma stable over long times is that the toroidal magnetic field is stronger on the inner side of the tokamak than on the outer edge, decreasing as the inverse of the distance from the symmetry axis. This field gradient drives ions and electrons to drift in opposite vertical directions. The resulting charge separation breeds an outward drift that pushes the whole plasma toward the outer walls [22]. To mitigate power losses and structural damages, modern tokamaks employ a *divertor*, which strips cold, impure plasma away from the hot core, easing the heat load. The divertor reshapes the magnetic field to form a boundary known as the *separatrix* [96]. This splits the cross-section of the tokamak into two zones: an inner region of closed flux surfaces where the plasma is hot and confined, and an outer region called the *scrape-off layer* where the plasma flows quickly to the walls and cools down. The inner plasma flows along closed helical magnetic-field lines and stays in the core, while particles crossing the separatrix are swept into the scrape-off layer and funnelled towards the divertor, thereby reducing stress to the main walls [96].

Heat and particle transport in a tokamak flows along two main paths: parallel and perpendicular to the magnetic-field lines. Of these, the cross-field (perpendicular) transport sets the confinement time, and thus the device's ability to make fusion power. Experiments show that the outward radial drift of charged particles, which drives heat losses, is up to two orders of magnitude higher than predicted by neoclassical transport

[1]The word "tokamak" comes from the Russian **то**роидальная **ка**мера с **ма**гнитными **к**атушками (BGN/PCGN: toroidalnaya kamera s magnitnymi katushkami), meaning "toroidal chamber with magnetic coils".

theory[2] [83]. This gap is known as *anomalous transport*. It springs from chaotic $E \times B$ vortices sweeping large swathes of particles across the magnetic field [22]. The turbulence is driven by drift-type micro-instabilities, tied to steep temperature and density gradients in the plasma, namely trapped-electron modes and ion-temperature-gradient modes [35]. Unravelling this turbulent transport is key to building better fusion reactors, which highlights the need for highly accurate numerical simulations.

1.4 Turbulence

Although the presence of electric currents and magnetic fields make plasmas harder to study, several features of plasma turbulence can be understood within the framework of hydrodynamics. Therein, fluid motion is described by the Navier–Stokes equations, a coupled set of nonlinear partial differential equations. From these equations, it is known that for sufficiently high values of the Reynolds' number – a dimensionless measure of the ratio of inertial to viscous forces – the flow becomes turbulent, forming unstable eddies that transfer energy from large to smaller scales until viscosity is large enough to dissipate it. This phenomenon has been known at least since the times of Leonardo da Vinci[3], but was first formally formalized in 1941 by Kolmogorov, building on an earlier idea of Richardson [112], for an incompressible fluid under conditions of homogeneity and isotropy [74, 75]. After some adjustments, Kolmogorov's phenomenological arguments also apply to magnetohydrodynamic (MHD) turbulence [134]. However, theoretical results of this kind are rare and tricky to obtain, making numerical experiments indispensable for advancing our understanding of turbulence.

Simulating turbulence presents two challenges. The first is that the computational cost of solving the governing equations grows with the Reynolds number. For instance, the number of degrees of freedom – and thus the number of equations to solve – scales as $\mathrm{Re}\,(1 + \ln \mathrm{Re})^{1/3}$ in two dimensions, with Re being the Reynolds' number, and as $\mathrm{Re}^{9/4}$ in three dimensions [132, 133]. Comparable estimates hold for incompressible MHD turbulence [134]. This issue has been addressed by simplifying the physical models used and by parallelizing the simulations in space.

[2] Neoclassical theory assumes a quiescent plasma. Here, particles cross field lines only through Coulomb collisions, paired with the wide geometric drifts caused by the curved toroidal magnetic field [67]. In a hot, reactor-grade plasma, such collisions are rare; the main transport mechanism is instead collective turbulent flow, which is far more effective at transporting energy.

[3] Da Vinci thought that turbulent flows could be divided into three areas: "doue la turbolenza dellacqua si genera, doue la turbolenza dellacqua simantiene plugho, doue la turbolenza dellacqua siposa", meaning "where the turbulence of water is generated, where it persists, where it settles" [26].

The second challenge is that turbulent flows are chaotic, exhibiting sensitive dependence on initial conditions: arbitrarily small differences in initial conditions can lead to exponentially diverging trajectories over time. Long-term prediction is therefore impossible: running the same numerical simulation on two different machines[4] will eventually yield diverging solutions.

Nevertheless, in dissipative systems such as tokamaks, asymptotic trajectories revolve around a shape in phase space known as an attractor. Due to phase contraction, this attractor has vanishing volume, meaning that phase-space volumes shrink under the flow. When the attractor has a fractal structure – locally the product of a continuous manifold and a Cantor set – it is called, for historical reasons, a strange attractor. Assuming that turbulent systems are ergodic, such that points at any time show the same statistical properties "at saturation", numerical solutions are not "worthless because even when they are different, they are statistically identical", a consequence of "the finite amount of energy available to the system" [119]. Thus, simulations have well-defined statistical properties, which are often what we care about.

1.5 Literature review of time-parallel methods

In this section, we review the state of the art in time parallelization. The goal is to solve initial-value problems (IVPs), that is, systems of ordinary differential equations (ODEs) coupled with initial conditions. These systems often arise by discretizing one or more partial differential equations (PDEs) using the method of lines. IVPs can be solved using one-step methods, such as Runge–Kutta methods, or multistep methods, such as Adams methods. One-step methods allow for the construction of higher-order schemes with fewer stability issues than multistep methods and are generally easier to implement when adaptive time-step control is required. Multistep methods, on the other hand, do not suffer from order reduction when applied to stiff ODEs, unlike one-step methods. Both approaches are inherently sequential, such that we must compute one or more steps at a time. By contrast, time-parallel methods allow us to parallelize computations along the time dimension.

Interest in the field of time parallelization has grown steadily over the past few decades, as shown by the sharp rise in publications on the subject [1]. This trend is fuelled by advances in computing power, parallel hardware, and new algorithms.

[4]This holds true even on the same machine. Floating-point arithmetic is non-associative, so minor changes – such as varying the number of processors (which alters the summation order in MPI reductions) or changing compiler optimizations – introduce microscopic round-off differences.

Gander [48] classifies time-parallel techniques into four broad categories: shooting methods, waveform-relaxation methods, multigrid methods, and direct methods. This classification is widely adopted [100], although many algorithms combine elements from multiple categories, and some can even be reinterpreted to fit into another group. It is therefore best to think of these categories as overlapping rather than strictly distinct. In the following sections, we look at a range of methods that have been proposed so far. Our aim is not to be exhaustive but to give a flavour of the main features of these approaches, focusing on stability, accuracy, and efficiency. For a comprehensive review, see [58].

1.5.1 Shooting methods

Shooting methods split the time domain into time chunks. Approximate initial conditions are assigned at the start of each chunk and distributed across processors to be integrated in parallel.

Nievergelt's 1964 method [98] was a direct technique that helped pave the way for multiple-shooting methods. Although not a multiple-shooting method itself, its core idea and structure influenced their development, which is why it is included here. The algorithm consists of four steps: (1) make a rough prediction of the solution at the start of each time chunk; (2) compute accurate trajectories starting from a number of initial conditions in a neighbourhood of each rough prediction; (3) interpolate these trajectories with the rough prediction; and (4) use the endpoint of the interpolated curve as a refined estimate for the initial condition of the next chunk. An illustration of this method is shown in Figure 1.3. Assuming that function evaluations are much more time-consuming than searching and interpolation, the theoretical speedup of this algorithm is proportional to the inverse of the number of time chunks. This is a property shared by many time-parallel methods. For linear equations, interpolation introduces no error, so the overall accuracy matches that of the underlying time-stepping scheme. For nonlinear equations, interpolation adds error, but balancing the number of trajectories and the chunk size can keep interpolation errors comparable to discretization errors. More recently, Barker [8] implemented Nievergelt's method on a general-purpose GPU, showing its minimal communication cost. However, floating-point operation counts and storage requirements remain high, hindering its applicability to large-scale problems.

Nievergelt's work inspired Bellen and Zennaro [9] and Chartier and Philippe [20] to recast the problem in fixed-point form and apply Newton's method. The resulting algorithm has quadratic local convergence and, for ODEs, stops in a finite number

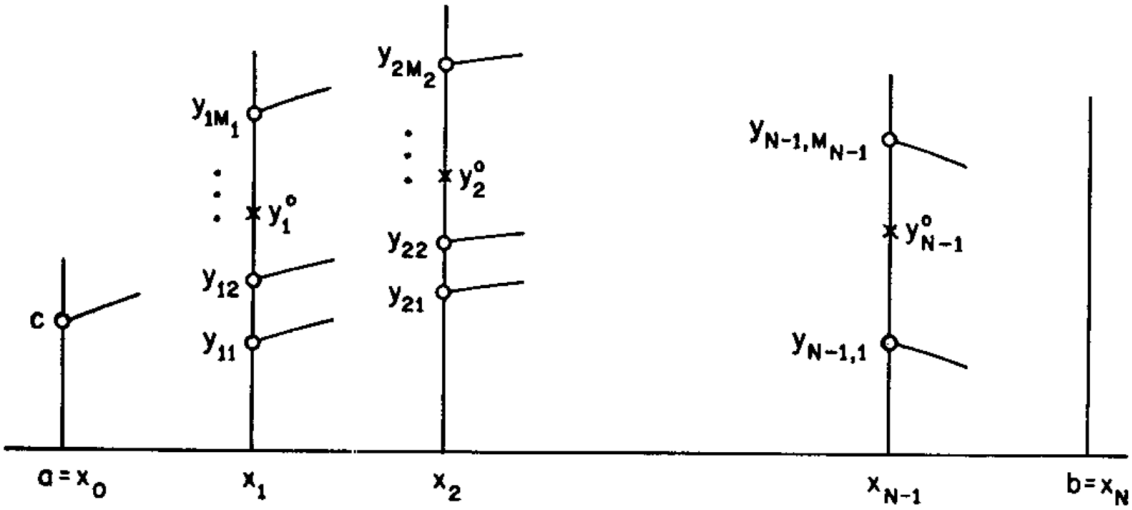

Figure 1.3: Original drawing by Nievergelt [98], demonstrating his method when solving a first-order ODE $y' = f(x, y)$ over N chunks, with initial condition $y(a) = c$, rough predictions $\{y_1^0, y_2^0, \ldots, y_{N-1}^0\}$, and M_i trajectories beginning at points $\{y_{i,1}, y_{i,2}, \ldots, y_{i,M_i}\}$ around each y_i^0.

of steps. Computing the Jacobian, however, can be prohibitively expensive for large systems. To mitigate this, Saha, Stadel, and Tremaine [118] proposed approximating the Jacobian terms using finite differences on a simplified physical model.

Along similar lines but independently of previous work, Lions, Maday, and Turinici [84] introduced the Parallel-in-Real-Time (Parareal) algorithm in 2001, which has attracted widespread attention for its simplicity and broad applicability. Parareal is an iterative procedure that combines a cheap but inaccurate (coarse) solver executed serially over the whole time domain, with an accurate but expensive (fine) solver applied in parallel to each time chunk. A detailed description of Parareal is provided in Section 2.7, and its first iteration is shown in Figure 1.4. Unlike Nievergelt's method, Parareal is an iterative process that refines the solution in parallel. Several authors have studied the convergence properties of Parareal, showing: superlinear and linear convergence for linear ODEs on bounded and unbounded time domains, respectively, and superlinear convergence for the one-dimensional heat equation [59]; superlinear convergence for nonlinear ODEs [53]; convergence for linear parabolic PDEs with constant coefficients [7]; convergence (with numerical experiments) for the heat equation with space- and time-dependent coefficients [117]. However, Parareal (like other time-parallel algorithms) is either unstable or inefficient for hyperbolic problems, likely due to phase errors between the fine and coarse solvers [116]. Modifications such as the Parallel Implicit Time Algorithm (PITA) [43] and its interpretation as a

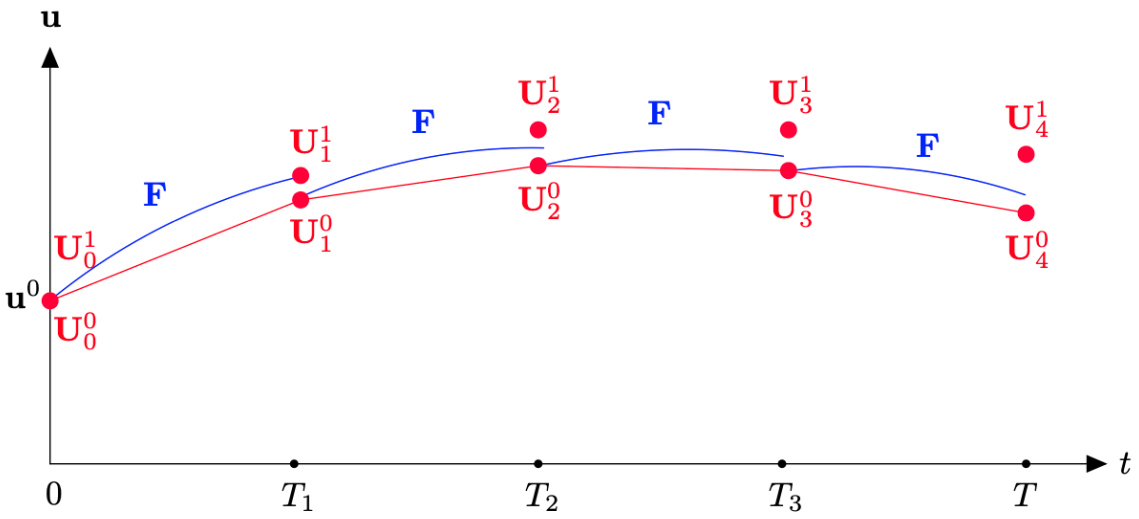

Figure 1.4: First iteration of Parareal, from [58]. The initial guess, computed with the coarse solver, is shown in red, while the first-iteration fine solutions, computed with the fine solver in parallel, are shown in blue.

Krylov–subspace method [27, 44, 59, 115] have been proposed to address this issue, but they can be memory intensive. Furthermore, Parareal has a smaller stability region than that of the fine solver it uses [59], and the sequential coarse solver limits the parallel efficiency[5] to the inverse of the number of iterations [89], which is a drawback for large-scale applications. To alleviate this, tasks in Parareal can be rescheduled through pipelining [6, 39].

1.5.2 Waveform-relaxation methods

Waveform-relaxation methods, originally developed in the context of circuit design, are closely related to Picard's method of successive approximations. These iterative schemes solve time-dependent problems by starting with an initial guess of the solution over the time domain and refining it progressively across the whole domain at once. Although they can, in principle, be applied directly, performance improves significantly when the space domain is decomposed into subdomains. In this configuration, local solutions are computed in parallel while neighbouring subdomains exchange (partial) information. These methods show superlinear convergence for ODEs but converge slowly when applied to PDEs via the method of lines [94]. Hence, Schwarz waveform-relaxation methods, which combine classical waveform-relaxation with an overlapping

[5]The efficiency of a parallel algorithm is defined as the ratio between the cost of the serial solution over the cost of the parallel solution, further divided by the number of processors available to the algorithm.

decomposition, are more commonly used for PDEs. For diffusive problems, both linear [60] and nonlinear [49], these methods converge faster asymptotically than their classical counterparts and terminate in a finite number of steps for linear hyperbolic systems [47]. However, convergence worsens as the time domain grows, and they often require many iterations to resolve the initial time steps. Waveform Relaxation with Adaptive Pipelining (WRAP) [78] addresses this issue by adaptively reducing the size of the time domain when convergence slows down and selecting only the iterates for which updating is necessary.

As mentioned, Schwarz waveform-relaxation relies on overlapping subdomains, which introduces redundant computations. This overlap can be avoided by recognizing that these methods work well with high frequencies but struggle to dampen low frequencies. By replacing the standard Dirichlet and Neumann transmission conditions with absorbing boundary conditions at the subdomain boundaries, optimized Schwarz waveform-relaxation handles the low frequencies better and speeds up the transfer of information. Furthermore, the expensive non-local operators in these boundary conditions can be approximated using local information, making them as cheap to compute as the standard boundary conditions. This approach removes the need for overlap and achieves faster convergence than classical Schwarz waveform-relaxation algorithms [47, 50].

1.5.3 Multigrid methods

Multigrid methods, unlike Parareal's two-level setup, use many coarsening levels. Their main component is a smoother, applied a few times to reduce high-frequency errors, enabling the projection of the problem onto a coarser grid.

The first multigrid method in this context was a space–time-parallel procedure proposed by Hackbusch in 1985 [62]. He observed that a naïve application of the Jacobi smoother in time caused multigrid to underperform or even diverge. Horton and Vandewalle [71] addressed this issue by using various strategies (line smoothers, adaptive semi-coarsening, forward-in-time prolongation operators) that resulted in reasonable contraction rates and mesh independence for the heat equation.

In 2012, Emmett and Minion introduced the Parallel Full Approximation Scheme in Space and Time (PFASST) method [40]. PFASST combines iterations from Parareal with those of Spectral Deferred Correction (SDC) [36] and Full Approximation Storage (FAS) [16]. In SDC, an ODE is rewritten into its Picard integral form, an approximate solution is obtained through quadrature on the discretized domain, and then corrected to a higher order of accuracy by solving a sequence of error equations. FAS extends

multigrid to nonlinear problems. PFASST is more efficient than Parareal because it uses a hierarchy of space-time discretizations rather than just two, and it has been successfully applied to small- and large-scale linear and nonlinear problems, e.g. [40, 124]. Bolten, Moser, and Speck [14, 15] analysed PFASST as a multigrid method by studying its iteration matrix for linear problems. They found that PFASST converges linearly for linear PDEs (occasionally superlinearly) and performs better on diffusive equations than on advective ones. Yet, the number of iterations required for convergence often grows with the number of time steps. This happens because, although PFASST satisfies the approximation property of multigrid methods, it does not satisfy the smoothing property, which requires the iterative solver to damp high-frequency error components far more strongly than low-frequency ones. SDC damps both at roughly the same rate, so PFASST lacks true algorithmic scalability: as resolution increases, it takes more iterations to converge. To be considered a true multigrid method, its smoother needs an additional damping parameter, which, however, reduces its convergence speed.

Another increasingly popular method is Multigrid Reduction in Time (MGRIT) [41], an extension of Parareal that combines F-smoothing with C- and F-smoothing in what is called FCF–smoothing[6] to make the algorithm multilevel and scalable [57]. Falgout et al. [41] showed that MGRIT is faster than Parareal when sufficient parallel resources and efficient communication networks are available, as required for large-scale applications. MGRIT can be extended with FAS to handle nonlinear problems, but the underlying Newton solver progressively becomes more expensive at coarser time levels. Falgout et al. [42] mitigated this issue for a nonlinear diffusion equation by improving the initial guess, employing spatial coarsening, skipping the inaccurate early iterates, and optimizing the number of levels. Finally, MGRIT has been analysed in both two-level [32, 57] and multilevel [123] settings. It shows superlinear convergence for parabolic equations and for hyperbolic problems with strongly diffusive components, often outperforming Parareal, but tends to diverge for purely hyperbolic problems. Some remedies have been proposed; for example, Howse et al. [72] successfully solved the variable-coefficient linear advection equation and the inviscid Burgers equation (with and without shocks) using adaptive spatial coarsening.

1.5.4 Direct methods

These methods are direct in the sense that they do not use iterations like the methods we have discussed so far do.

[6] F- and C-smoothing refer to fine and coarse integration, respectively.

Predictor-corrector methods fall into this category. They use a prediction step to extrapolate a new point and a correction step to refine the prediction using a different stepping scheme. Their speedup is not fully proportional to the number of processors used, which limits them to small-scale parallelism. Moreover, they do not preserve the stability and accuracy of the underlying time-stepping scheme. Examples include the method of Miranker and Liniger [95], Parallel Runge–Kutta (PRK) methods [65, Section II.11], and Revisionist Integral Deferred Correction (RIDC) methods [23]. RIDC methods, in particular, pipeline the computations of SDC with an appropriate time lag to compute the corrections in parallel. See [23] for a convergence and stability analysis of RIDC methods and [99] for a comprehensive review.

Sheen, Sloan, and Thomée [120] proposed a method based on Laplace-transforming the problem in time, solving a sequence of independent subproblems at quadrature nodes in parallel, and then inverting to recover the full solution. This approach is restricted to problems where the Laplace transform applies, that is, linear problems with time-independent coefficients. Somewhat related to this is Paraexp [51], a time-parallel method originally developed for linear problems using an overlapping decomposition of the time domain. Paraexp splits the problem into its homogeneous and inhomogeneous components and solves it in parallel using an approximation of the matrix exponential. Speedup is achieved because there exist many near-optimal approximations of the matrix exponential, such as those based on Krylov–subspace methods or rational-function expansions. Extensions of Paraexp to nonlinear problems are found in [52, 76]. Neither approach requires coarsening in time, but both perform poorly for hyperbolic problems and are limited to problems with constant coefficients.

Cyclic Reduction [48] is another standard parallel method that has been used in this context. Its steps can be executed in parallel to recursively halve the size of the matrix system obtained from discretization until a two by two system remains, which can be solved directly. Cyclic Reduction can also be interpreted as a multigrid method.

All-at-once (or ParaDiag) methods combine approximate solutions into a single monolithic system. Maday and Rønquist [90] proposed a method that decomposes the resulting coefficient matrix into space and time components using Kronecker products, allowing the time component to be diagonalized and the system to be solved in parallel if the time steps are distinct. For a detailed analysis of the truncation error on geometric time grids, the round-off error due to diagonalization, and associated trade-offs, see [54–56]. Alternatively, McDonald, Pestana and Wathen [93] introduced a method that relies on circulant preconditioning for the asymmetric block Toeplitz

coefficient matrix arising from the discretization of the all-at-once formulation of the problem on a uniformly spaced time grid. This preconditioner can be efficiently diagonalized via a recursive application of the Fast Fourier Transform, which is known for its highly optimized implementations. Experimental results show that this approach yields a small number of iterations independent of the number of time steps for ODEs, when used in conjunction with MINRES or GMRES, and nearly independent for PDEs. Because the formulation on which it is based is reasonably robust to temporal perturbations, this method can be extended to non-uniform time-stepping schemes. However, it requires a relatively large number of iterations to solve the wave equation, depending on the degree of dissipation of the underlying time-stepping scheme, and it is sensitive to the choice of initial conditions [61].

1.6 Parareal in fusion research

When applied to turbulent problems, the exponential divergence of trajectories is expected to deteriorate the convergence of time-parallel methods. This is especially true under strong turbulence: Steiner et al. [126] found that Parareal becomes unstable when the Reynolds number is high and the nonlinear convection term dominates. Despite this, Samaddar, Newman, and Sánchez [119] successfully applied a carefully tuned implementation of Parareal to a fully developed turbulent system modelling drift-wave turbulence in the plasma core. Specifically, the authors studied a turbulence model known as the dissipative trapped-electron mode (DTEM) in a doubly periodic slab geometry, which is widely used in the fusion community. The DTEM model extends the Navier-Stokes equations by incorporating two nonlinearities that control the transfer of energy between modes: the $E \times B$ nonlinearity, which drives a non-local cascade of energy towards the high-frequency modes, and the polarization nonlinearity, which drives a local cascade towards the low-frequency modes. Using the method of lines, space was discretized using a pseudo-spectral code with approximately 10^5 degrees of freedom, while Parareal handled the time dimension. In the setup of Parareal, the fine solver was a Jacobi-preconditioned Newton–Krylov implicit solver and the coarse solver was a fourth-order Runge–Kutta scheme with a time step that is eight times larger than that of the fine solver. The computations of Parareal were pipelined for efficiency using an event-based approach [39].

As noted in [119], jumps between Parareal time chunks are bounded by the system's free energy. By choosing a convergence criterion based on statistical properties rather than pointwise continuity at chunk boundaries, the algorithm was able to replicate

the statistical behaviour of turbulence, accurately capturing statistics and bulk effects on large scale modes. This observation aligns with the intuitive picture outlined in Section 1.4 and motivates the proximity function we introduce in Section 3.4 and the moving-window algorithm we develop in Chapter 4. As a result, convergence hinges on two factors: the cut-off value of the chosen spatial scale and the size of the time chunks. According to the analysis in [110, 111], this is due to the way the nonlinear terms in the model transfer energy. The $E \times B$-driven cascade transfers energy from low- to high-frequency modes, where it is dissipated. Because this transfer can interfere with the Parareal correction, convergence depends on how much energy is transferred, which in turn depends on the cut-off frequency and chunk size. A similar phenomenon occurs for the polarization nonlinearity, although convergence quality appears to depend on the magnitude of the energy cascade rather than its sign. Nevertheless, the whole spatial spectrum converges due to the presence of strong nonlinear couplings between mode pairs, as shown by simulations. In conclusion, Parareal achieved parallel speedups of up to 10, which is a significant improvement in this context.

1.7 Outline

Having established why it is important to speed up tokamak simulations and highlighted how chaos can be both an obstacle and an opportunity, we now outline the structure of the remaining chapters in this thesis.

In Chapter 2, we present the theoretical background underpinning our work. We begin with initial-value problems for systems of ordinary differential equations and explain how chaos arises in these contexts through sensitive dependence on initial conditions, Lyapunov exponents, and attractors. We also introduce the numerical tools used throughout this thesis: Runge–Kutta methods for serial time-stepping and the Parareal algorithm [84]. As discussed in Section 1.5, different time-parallel algorithms break the sequential nature of time-stepping in different ways. Some, such as Parareal, employ a predictor-corrector scheme. We focus on Parareal because it is the most widely studied method and serves as a natural entry point into time-parallel techniques. This choice was deliberate to maximize clarity and emphasize the impact of the new theoretical framework, which is the main contribution of this thesis.

In Chapter 3, we introduce this framework for analysing time-parallel methods applied to nonlinear and potentially chaotic problems. This framework is built on the well-established theory of contraction mappings and introduces new concepts such as

outer and inner balls and a proximity function to measure (statistical) convergence in chaotic systems while allowing for trajectory divergence. We derive explicit convergence rate estimates for Parareal and validate them through numerical experiments on toy problems. This framework lays the foundation for the work that follows.

In Chapter 4, we present the moving-window (MoWi) algorithm, a meta-algorithm that splits the time domain into overlapping windows and uses time-parallel integration within each window. Doing so, MoWi can reuse partial solutions and control how they propagate forwards. MoWi is designed to sample initial conditions from the invariant distribution of a chaotic system's global attractor without allowing local chaos to derail convergence. Using the framework developed in Section 3.2, we show that if each window is solved to a suitable tolerance, MoWi stays on track in phase space. This tracking ability allows MoWi to outperform a direct application of Parareal over the full time domain in cases where Parareal struggles or stalls. Importantly, MoWi is a flexible, black-box meta-algorithm that can wrap around any time-parallel method, not just Parareal.

In Chapter 5, we extend MoWi with three adaptive strategies:

1. AMoWi-1, in which we adapt the window length (stretching or shrinking);
2. AMoWi-2, in which we adapt the window shift (leaping or hopping);
3. AMoWi-3, in which we adapt the weighting factor in the proximity function (zooming in or out).

These strategies let MoWi handle changes in the local Lyapunov exponent across time and space. When a trajectory comes into a region of the attractor with a high local Lyapunov exponent, the flow becomes more expansive and unpredictable, causing the fine and coarse solvers to diverge more quickly and driving up the effective convergence factor for that window. If this factor grows too large, the solver fails to converge within its fixed iteration budget. We provide formal proofs that a robust recovery mechanism exists: by taking a corrective action, convergence can always be restored in a predictable, finite number of restarts. These results show that the adaptive control problem is solvable in principle. Hence, MoWi remains on track and maintains meaningful speedups. To support this, we demonstrate these strategies on the Kuramoto–Sivashinsky equation, chosen for its spatio-temporal chaos and as a natural next step in complexity.

Finally, in Chapter 6, we summarize the main findings, discuss limitations, and highlight open questions. We suggest directions for future research, including combining

adaptive strategies and integrating MoWi with other time-parallel methods like MGRIT, PFASST, or ParaDiag. We also consider extensions to production plasma codes and applications beyond fusion, as well as potential intersections between MoWi and modern machine-learning approaches.

Chapter 2

Prerequisites

The mathematical foundation of this thesis rests on key concepts from the theory of ordinary differential equations and dynamical systems, particularly in the context of chaos. While a fully rigorous and general theory of chaos remains an active area of research [143], many practical insights can be drawn from foundational notions such as sensitive dependence on initial conditions, Lyapunov exponents, and attractors.

In this chapter, we introduce these core concepts. Our aim is not to be exhaustive but to give enough detail for the reader to follow our methodology developed in later chapters, especially Chapters 3 and 4. We also review the main numerical tools employed throughout this work: Runge–Kutta methods, which serve as our serial solvers, and the Parareal algorithm, which we use both as our time-parallel algorithm of choice in Chapter 3 and as a subroutine within MoWi in Chapter 4.

2.1 Basics

We begin with basic notation for Euclidean spaces, norms, and balls. For any integer $d \geq 1$, let $\mathbb{R}^d$ denote the d-dimensional Euclidean space. A vector $x \in \mathbb{R}^d$ has components $[x_1, \ldots, x_d]$. Throughout this thesis, we typically take its vector norm $\|x\|$ to be the p-norm:

$$\|x\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{1/p}, \text{ for } p \geq 1. \tag{2.1}$$

As $p \to \infty$, the p-norm approaches the ∞-norm:

$$\|x\|_\infty = \max_{i=1,\ldots,d} |x_i|. \tag{2.2}$$

We use the term p-norm broadly, including this limiting case. For a matrix $A \in \mathbb{R}^{d\times d}$, the induced operator norm $\|A\|$ with respect to a given vector norm is

$$\|A\| = \sup_{\|x\|=1} \|Ax\|. \tag{2.3}$$

Note that the choice of norm does not change the qualitative results, since the analysis takes place in a finite-dimensional vector space, but it does affect the quantitative details because the key constants depend on the norm. In practice, our theory is written for a general p-norm, while we use the standard Euclidean 2-norm in the numerical experiments, which is the usual choice in this context.

Next, we define the distance between sets and also the neighbourhoods of sets and their boundaries. Let

$$\operatorname{dist}(y, \mathcal{X}) = \inf_{x\in\mathcal{X}} \|x - y\|, \tag{2.4}$$

be the (semi-)distance of a point y from a set $\mathcal{X}$. Then we define the distance between two sets[1] as

$$\operatorname{dist}(\mathcal{Y}, \mathcal{X}) = \sup_{y\in\mathcal{Y}} \operatorname{dist}(y, \mathcal{X}). \tag{2.5}$$

This allows us to define a neighbourhood as

$$\mathcal{N}_\epsilon(\mathcal{X}) = \{y \in \mathbb{R}^d \,:\, \operatorname{dist}(y, \mathcal{X}) < \epsilon\}, \tag{2.6}$$

and its boundary as

$$\partial\mathcal{N}_\epsilon(\mathcal{X}) = \{y \in \mathbb{R}^d \,:\, \operatorname{dist}(y, \mathcal{X}) = \epsilon\}. \tag{2.7}$$

Note that if $\operatorname{dist}(\mathcal{Y}, \mathcal{X}) \le \epsilon$ then $\overline{\mathcal{Y}} \subseteq \mathcal{N}_\epsilon(\overline{\mathcal{X}})$, so that $\operatorname{dist}(\mathcal{Y}, \mathcal{X}) = 0$ implies $\overline{\mathcal{Y}} \subseteq \overline{\mathcal{X}}$.

We also define open and closed balls in $\mathbb{R}^d$. The open ball of radius ϵ centred at x (with respect to norm $\|\cdot\|$) is

$$B_\epsilon(x) = \{y \in \mathbb{R}^d \,:\, \|x - y\| < \epsilon\}, \tag{2.8}$$

and its boundary

$$\partial B_\epsilon(x) = \{y \in \mathbb{R}^d \,:\, \|x - y\| = \epsilon\}. \tag{2.9}$$

Note that $B_\epsilon(x) = \mathcal{N}_\epsilon(\{x\})$ and $\partial B_\epsilon(x) = \partial\mathcal{N}_\epsilon(\{x\})$.

These definitions of balls and neighbourhoods will be central to our convergence analysis in Section 3.2.

[1] This distance is not symmetric; a symmetric variant would be the Hausdorff distance. For our aims, though, the one-sided form is enough. This is because (1) when a set of trajectories $\mathcal{X}$ is attracted to a set $\mathcal{A}$, we mean that the distance from $\mathcal{X}(t)$ to $\mathcal{A}$ tends to zero as $t \to \infty$; the reverse direction plays no role, and (2) the key inclusion condition for MoWi demands that the image of the inner ball under the system's evolution lie wholly inside the next outer ball, which is again a one-sided containment statement.

2.2 Initial-value problems

Many transport mechanisms in tokamak plasmas can be described by systems of nonlinear partial differential equations (PDEs) coupled with initial and boundary conditions. These PDEs are commonly approximated via the method of lines (spatial discretization followed by time integration), yielding systems of ordinary differential equations (ODEs) coupled with initial conditions. In this thesis, we focus on solving these systems, known as Cauchy or initial-value problems (IVP).

Any ODE of finite order can be rewritten as a system of first-order ODEs through a suitable change of variables. Most general-purpose ODE solvers assume that the system is in first-order form. Thus, an ODE is typically written as

$$u'(t) = f\big(t, u(t)\big), \tag{2.10}$$

where t is time, u is the solution evolving in phase space, and f is the right-hand side derivative, often arising from the spatial discretization of a PDE. We do not directly address PDEs in this thesis, but we assume that, under a suitable discretization, the phase space (and thus the function f) retains enough structure from the PDE that the numerical results remain physically meaningful. To complete the IVP specification, we must declare an initial condition u_0[2] such that

$$u(0) = u_0. \tag{2.11}$$

An ODE is called autonomous if f does not explicitly depend on t. Any non-autonomous system can, however, be reformulated as an autonomous system by adding a new state variable representing time[3]: $\hat{u}(t) = [u(t), t]$. Also, when f is complex-valued, an ODE can be written in real form by considering the real and imaginary parts separately. For these reasons, we focus on real-valued autonomous ODEs.

So, throughout, we take an IVP to be

$$\begin{aligned} u'(t) &= f\big(u(t)\big), \\ u(0) &= u_0, \end{aligned} \tag{2.12}$$

where $d \geq 1$, $t \in \mathbb{R}$, $u : \mathbb{R} \to \mathbb{R}^d$, $f : \mathbb{R}^d \to \mathbb{R}^d$, and $u_0 \in \mathbb{R}^d$. Note that, for most real problems, especially those that come from PDEs, nonlinear IVPs are non-integrable, that is, one cannot solve the IVP (2.12) in closed form. We follow [127] to define the basic concepts related to existence and uniqueness of a solution for the IVP (2.12).

[2]We take the initial time to be $t = 0$, to simplify the notation even further.

[3]The transformation does change the system's dimension and Lyapunov spectrum, but in a simple, well-understood way: the new system inherits the original d exponents and gains one extra exponent equal to zero. The system's chaotic nature, set by its positive exponents, is thus untouched.

Definition 2.2.1 ([127, Definition 2.1.2])**.** We say that the IVP (2.12) defines a dynamical system on a subset $\mathcal{X} \subseteq \mathbb{R}^d$ if, for every $u_0 \in \mathcal{X}$, there exists a unique solution $u(t) \in \mathcal{X}$ for all $t \geq 0$.

According to the classical theory of ODEs, a sufficient (though not necessary) condition for the local existence and uniqueness of a solution to the IVP (2.12) is that f is Lipschitz. This assumption often extends nicely to studying the behaviour of numerical approximations.

Definition 2.2.2 ([127, Definition A.l])**.** A function $f : \mathbb{R}^d \to \mathbb{R}^d$ is said to be Lipschitz (continuous) on $\mathcal{X} \subset \mathbb{R}^d$ if there exists a constant Λ_f such that

$$\|f(u) - f(v)\| \leq \Lambda_f \|u - v\|, \quad \forall u, v \in \mathcal{X}, \tag{2.13}$$

for some norm $\|\cdot\|$ on $\mathbb{R}^d$. If f is Lipschitz on $\mathbb{R}^d$, then f is said to be globally Lipschitz (continuous). If f is Lipschitz on every bounded subset of $\mathbb{R}^d$, then f is said to be locally Lipschitz (continuous).

Theorem 2.2.3 ([127, Theorem 2.1.3])**.** Let $f : \mathbb{R}^d \to \mathbb{R}^d$ be globally Lipschitz on $\mathbb{R}^d$. Then there exists a unique solution $u(t)$ to the IVP (2.12) for all $t \in \mathbb{R}$. Following Definition 2.2.1, this means that the IVP (2.12) defines a dynamical system on $\mathbb{R}^d$.

The scope of Theorem 2.2.3 is that, in general, it is necessary to assume that the vector field is globally Lipschitz, otherwise singularities may develop in the solution in a finite time. On the other hand, under local Lipschitz continuity, one can use the method of successive approximations – also known as Picard's iteration [128] – to show existence and uniqueness of a solution to the IVP (2.12) for all $u_0 \in \mathcal{X}$ and $t \in [0, T(u_0))$, where $T(u_0)$ denotes the time at which the solution leaves the set $\mathcal{X}$, and may be either finite or infinite.

Theorem 2.2.4 ([127, Theorem 2.1.5])**.** Let $f : \mathbb{R}^d \to \mathbb{R}^d$ be locally Lipschitz. Define $\mathcal{X} = \{u \in \mathbb{R}^d : \|u - u_0\| \leq U\}$ and let $M = \max_{u \in \mathcal{X}} \|f(u)\|$. Then there exists a unique solution of the IVP (2.12) on the interval $|t| \leq U/M$ which satisfies $u(t) \in \mathcal{X}$.

Furthermore, if there does not exist a unique solution on the interval $t \in [0, \infty)$ then there exists $T > 0$ such that $\|u(t)\| \to \infty$ as $t \to T$ (from below). Hence, provided that no solution of the IVP (2.12) becomes unbounded in a finite time, it defines a dynamical system on $\mathbb{R}^d$.

We can now define the evolution semigroup. This, together with its action, will be important in Sections 2.3 and 2.4.

Definition 2.2.5 ([127, Definition 2.1.6])**.** Given a dynamical system generated by the IVP (2.12), we define the evolution semigroup $S(\cdot, t) : \mathcal{X} \to \mathcal{X}$ to be the operator such that $u(t) = S(u(0), t)$ for all $t \geq 0$. This operator has the properties:

- $u(t_1 + t_2) = S\big(u(t_2), t_1\big) = S\big(u(t_1), t_2\big)$ for all $t_1, t_2 \geq 0$,
- $S(\cdot, 0) \equiv I$, the identity on $\mathbb{R}^d$.

One often prefers the notation $u(t) = S(t)u_0$. We do too, as it makes it obvious that the evolution semigroup is just a convenient notation for advancing a solution through t.

Definition 2.2.6 ([127, Definition 2.1.8])**.** We define the action of the evolution semigroup on a bounded set $\mathcal{X} \subset \mathbb{R}^d$ by

$$S(t)\mathcal{X} = \bigcup_{\mathcal{Y} \in \mathcal{X}} S(t)\mathcal{Y} \tag{2.14}$$

2.2.1 Continuous dependence on initial conditions

In what follows, we assume that $u(t)$ and $v(t)$ be two solutions of the IVP (2.12) with initial conditions u_0 and v_0, respectively.

Definition 2.2.7 ([127, Definition 2.1.11])**.** We say that $u(t)$ depends continuously on u_0 if for all $\epsilon > 0$ there exists $\delta > 0$ such that $v(t) \in B_\epsilon(u(t))$ for all $v_0 \in B_\delta(u_0)$. If this holds true for all $u_0 \in \mathbb{R}^d$, then we also say that the dynamical system defined by the IVP (2.12) is continuous with respect to the initial data.

An important consequence of f being Lipschitz is that the dynamical system will be continuous with respect to initial data.

Theorem 2.2.8 ([65, Theorem 10.2], [127, Theorem 2.1.12])**.** Suppose that the IVP (2.12) defines a dynamical system on $\mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}^d$ is locally Lipschitz. Let $\mathcal{X} \subset \mathbb{R}^d$ be a bounded set with the property that $u(t), v(t) \in \mathcal{X}$ for $t \in [0, T]$. Let Λ_f be the Lipschitz constant of f on $\mathcal{X}$. Then

$$\|u(t) - v(t)\| = \|S(t)u_0 - S(t)v_0\| \leq e^{\Lambda_f t}\|u_0 - v_0\|, \quad \forall t \in [0, T]. \tag{2.15}$$

Hence, the dynamical system is (Lipschitz) continuous with respect to initial data.

In the context of numerical methods, this implies that the solution depends continuously on errors in the initial condition, with the error growing (at worst) with rate Λ_f. In practice, however, this holds only when the function f is known exactly: in the context of PDEs, where the right-hand side is often an approximation of the true spatial operators, discretization introduces an additional source of error that must also be taken into account [65, Theorem 10.2].

Furthermore, Λ_f can be very large in practice, as is often the case with stiff or chaotic problems. If f is continuously differentiable, so that its Jacobian $Df = \partial f/\partial u$ exists and is continuous, then the Lipschitz constant of f can be expressed in terms of its Jacobian. This is shown in the following theorem, the proof of which relies on the Mean-Value Theorem.

Theorem 2.2.9 ([65, Lemma 7.2])**.** Let $\|Df(u)\| \leq \Lambda_f$ for all $u \in \mathbb{R}^d$. Then f is Lipschitz continuous on each convex, compact subset with constant Λ_f. In particular, Λ_f can be tightened to

$$\Lambda_f = \sup_{u \in \mathbb{R}^d} \|Df(u)\|. \tag{2.16}$$

2.3 Chaos

Since turbulence is a chaotic phenomenon, we should establish a clear definition of it. There are many mathematical notions of chaos in the literature. A brief and concise presentation of them can be found in [81]. None of these definitions are equivalent, but there are various implications. Within the context of our research, we only need to highlight a few key aspects, which we will do in the forthcoming sections.

2.3.1 Sensitive dependence on initial conditions

One of the hallmarks of chaos is sensitive dependence on initial conditions. Its roots can be traced back to the early findings of Maxwell [92], Poincaré [106], and Hadamard [63], but it was famously brought to prominence by Lorenz in his 1963 work on atmospheric convection [85]. The formal definition follows that of Definition 2.2.7 and is as follows.

Definition 2.3.1 (SDIC)**.** A dynamical system defined by the IVP (2.12) has sensitive dependence on initial conditions (SDIC) if there exists a constant $\delta > 0$ such that for all $v_0 \in \mathcal{X}$ and every $\epsilon > 0$, there exists a point $u_0 \in X$ such that $v_0 \in B_\epsilon(u_0)$ but $v(t) \notin B_\delta(u(t))$ for $t \geq 0$.

In other words, no matter how small your neighbourhood around u_0, you can find a point v_0 whose orbit eventually diverges from that of u_0 by at least δ. The number δ is not unique, since any δ' such that $0 < \delta' < \delta$ works too. This could be rectified by taking the supremum of all such δ, but this is often too hard to compute. In practice, what matters is that some δ exists.

However, Definition 2.3.1 does not specify the rate of such divergence. As a consequence, physicists often prefer to characterize SDIC with the following stronger definition.

Definition 2.3.2 (λ-SDIC)**.** A dynamical system defined by the IVP (2.12) has λ-SDIC if there exists $\lambda > 0$ such that for almost all[4] points $u_0 \in \mathcal{X}$, for all $\epsilon > 0$ there exists $t > 0$ and $C > 0$ such that $Ce^{\lambda t}\|u_0 - v_0\| \leq \|u(t) - v(t)\| \leq C^{-1}e^{\lambda t}\|u_0 - v_0\|$ for almost all points $v_0 \in B_\epsilon(u_0)$.

Definition 2.3.2 is the most commonly employed and practical method for distinguishing between regular and chaotic motion among physicists. In this context, λ is the parameter quantifying the sensitive dependence on initial conditions, and it is identified with the maximal Lyapunov exponent. If $\lambda > 0$, then orbits separate exponentially on average. We will treat Lyapunov exponents in the next section.

2.3.2 Lyapunov exponent

Lyapunov exponents (LEs) describe the rate of expansion and contraction of neighbouring trajectories in phase space, thus playing a central role in the study of chaos, stability, and predictability in dynamical systems. The study of LEs originated from the work of Lyapunov on the stability of solutions of differential equations [88]. Since then they have been extensively used for studying dynamical systems. Conditions for their existence have been characterized by Oseledets [102] and the connection between them and chaotic behaviour was developed by Pesin [105]. The value for the maximal LE is a signature of chaotic or regular behaviour, because for a chaotic orbit at least one LE is positive, implying exponential divergence of nearby orbits [37].

Consider the IVP (2.12) and its solution $u(t)$. We want to describe the expansion and contraction of trajectories near $u(t)$. Hence, it is natural to study the linearization of (2.12) around $u(t)$, which gives rise to a variational system for infinitesimal

[4]In this context, "almost all" implies that the probability of randomly choosing an initial condition where this property fails is zero. Theoretically, this excludes specific non-chaotic sets like unstable periodic orbits. Practically, these exceptions are irrelevant: finite precision inevitably pushes numerical simulations off such unstable paths onto typical chaotic trajectories.

perturbations $\delta u(t)$,

$$\begin{aligned} \delta u'(t) &= J(t)\delta u(t), \\ \delta u(0) &= \delta u_0, \end{aligned} \tag{2.17}$$

where $J(t) = Df(u(t))$ is the Jacobian of the right-hand side of (2.12).

Definition 2.3.3. We define

$$\lambda = \limsup_{t\to\infty} \frac{1}{t} \log \|\delta u(t)\|, \tag{2.18}$$

to be the (maximal) Lyapunov exponent of the IVP (2.12).

Note that, despite the norm $\|\cdot\|$ being in (2.18), the LE does not depend on it [38]. See [10, 11] for more details.

Apart from using it as a criterion for the chaoticity or the regularity of an orbit, the maximal LE is extremely important in the context of numerical integration because it determines the average time after which the propagation of rounding errors outgrows any numerical prediction. To see why, let δ be a perturbation for the state of a chaotic system. This perturbation grows exponentially in time like $\exp(\lambda t)$, reaching a size Δ after a time

$$\frac{1}{\lambda} \log\left(\frac{\Delta}{\delta}\right). \tag{2.19}$$

Given that (2.19) is in units of time, the maximal LE is dimensionally equivalent to a frequency, and its inverse is often identified as the predictability time or the turbulent timescale of a system.

However, since λ is defined asymptotically, (2.19) is just an approximate relation, and thus the maximal LE may not be accurate for describing short-term behaviour. For this reason, practitioners often prefer finite-time alternatives to λ, such as the finite-time Lyapunov exponent (FTLE), which is defined as

$$\lambda_t(u_0) = \frac{1}{t} \log\left(\frac{\|\delta u(t)\|}{\|\delta u(0)\|}\right), \tag{2.20}$$

where we explicitly note the dependence on the initial condition u_0. This is a quantity akin to λ that describes the local divergence rate of trajectories. It is useful because, in general, nearby trajectories do not diverge at the same rate: for instance, Nese [97] studied how λ_t varies on the Lorenz attractor and found that it varies considerably with time, i.e. with respect to the location of a system in phase space. The FTLE has proved to be of critical importance in applications such as weather forecasting [73, 142, 144]. Finally, λ_t converges to λ in the limit $t \to \infty$. We will make use of this definition in Section 3.4.

2.4 Attractors

We have so far presented a cursory treatment of chaos primarily to motivate the idea of complicated sets supporting chaotic dynamics. Such sets are often called strange attractors. In this section, we follow [127] and consider general sets which are attracting for solutions of the IVP (2.12).

We do not impose restrictions on the possible dynamics within the attracting set. We only assume the existence of a compact set possessing some form of attractivity. This will be useful Chapter 4.

Definition 2.4.1 ([127, Definition 2.2.3])**.** A set $\mathcal{Y}$ is said to be forward (backward) invariant (under S) if $S(t)\mathcal{Y} \subseteq \mathcal{Y}$ ($S(t)\mathcal{Y} \supseteq \mathcal{Y}$) for all $t \geq 0$. If $\mathcal{Y}$ is both forward and backward invariant (under S), so that $S(t)\mathcal{Y} \equiv \mathcal{Y}$ for all $t \geq 0$, then $\mathcal{Y}$ is said to be invariant under S.

An attractor is a set in phase space to which trajectories converge as time grows. Here, we use the following general definition.

Definition 2.4.2 ([127, Definition 2.7.1])**.** A set $\mathcal{A}$ attracts a set $\mathcal{X}$ under $S(t)$ if, for any $\epsilon > 0$, there exists T such that $S(t)\mathcal{X} \subset \mathcal{N}_\epsilon(\mathcal{A})$ for all $t \geq T$. A compact invariant set $\mathcal{A}$ is an attractor if it attracts an open neighbourhood of itself. A global attractor is an attractor which attracts every bounded set in $\mathbb{R}^d$.

An attractor which is not a global attractor is sometimes called a local attractor.

Attractors can have one (or more) positive LE, corresponding to a direction in which the system experiences a repeated stretching and folding that decorrelates nearby orbits on the attractor.

Many systems arising in practical applications (fluid flows, reaction-diffusion problems, plasma transport) exhibit dissipativity. A classical, though insufficient, definition for dissipativity is that the Jacobian determinant of the vector field in (2.12) is negative, i.e.

$$\det\left(\left[\frac{\partial f_j}{\partial x_k}\right]_{j,k=1,\ldots,n}\right) < 0. \tag{2.21}$$

Equality means that the system conserves phase space volumes, which is a property of Hamiltonian systems. Condition (2.21) indicates a "dissipation" of phase space volumes, but is not enough to ensure a bounded attractor. For instance, the two-dimensional system

$$x_1' = \frac{x_1}{2}, \quad x_2' = -x_2, \tag{2.22}$$

has unbounded solutions, despite having a negative Jacobian determinant. A more robust way for a dynamical system to be dissipative is if all trajectories eventually enter and remain confined in some bounded set. Formally, we have the following.

Definition 2.4.3 ([127, Definition 2.8.7])**.** A dynamical system on $\mathbb{R}^d$ is said to be dissipative if there is a bounded, forward invariant set $\mathcal{X}$ with the property that, for any bounded set $\mathcal{Y} \subseteq \mathbb{R}^d$, there exists $T \geq 0$ such that $S(t)\mathcal{Y} \subseteq \mathcal{X}$ for all $t > T$. The set $\mathcal{Y}$ is called an absorbing set.

Absorbing sets need not be unique. Once an absorbing set has been identified, one may define the global attractor of a dynamical system with the following result.

Theorem 2.4.4 ([127, Theorem 2.8.13])**.** A dissipative dynamical system with absorbing set $\mathcal{Y}$ has global attractor

$$\mathcal{A} = \overline{\bigcap_{t \geq 0} S(t)\mathcal{Y}}. \tag{2.23}$$

In practice, one often finds that dissipative systems satisfy the condition

$$\exists \alpha \geq 0, \beta > 0 \ : \ \langle f(u), u \rangle \leq \alpha - \beta \|u\|^2, \quad \forall u \in \mathbb{R}^d, \tag{2.24}$$

where the norm is that induced by the inner product. This leads to

$$\|u(t)\|^2 \leq \|u_0\|^2 e^{-2bt} + \frac{a}{b}\left(1 - e^{-2bt}\right). \tag{2.25}$$

Hence the open ball $B_r(0)$ with $r = \sqrt{\alpha/\beta + \epsilon}$ is an absorbing set for any $\epsilon > 0$ and any given initial data [127, Theorem 2.8.8].

We conclude with some remarks:

- When extending these ideas to PDEs (and thus infinite-dimensional phase spaces), the discussion becomes more complicated. The long-term behaviour of the system may depend on the chosen norm, and even with a fixed norm, compactness is not guaranteed for closed and bounded sets, such that even if we find some bounded absorbing set $\mathcal{Y}$, as above, we have no guarantee that $\cap_{t \geq 0} S_t(\overline{\mathcal{Y}})$ will be non-empty. Thus, proving the existence of a global attractor usually requires additional smoothing properties. Nevertheless, many PDEs of physical interest (like the 2D Navier-Stokes equations, certain reaction-diffusion systems, Kuramoto–Sivashinsky) have been shown to possess global attractors of finite dimension.

- Numerical approximations of attractors are often obtained by integrating a system over a long time interval and plotting the resulting trajectory after discarding the initial transient phase. However, as $T \to \infty$, the error bound for the numerical solution becomes unbounded, suggesting this method may not accurately approximate the attractor. Despite this, in practice, different numerical methods with sufficiently small step sizes tend to produce similar attractor plots, implying that these numerical solutions approximate the true attractors well. Stuart and Humphries [127, Chapter 7] have demonstrated that, at least in the context of classical techniques like Runge–Kutta methods and linear multistep methods, a numerical approximation of a dynamical system with an attractor $\mathcal{A}$ also possesses a numerical attractor $\mathcal{A}_h$ if the step size h is sufficiently small. Furthermore, as $h \to 0$, $\mathcal{A}_h$ converges to $\mathcal{A}$ in a set-theoretic sense.

- Finally, strange attractors and the associated exponential divergence of trajectories underscore the challenges for time-parallel algorithms: although each subinterval can be solved in parallel, the sensitive dependence on initial conditions can require careful handling of interface information. In Chapter 4, we propose a new strategy for time-parallel integration specifically tailored to such dissipative systems, aiming to exploit their attractor properties while avoiding the pitfalls of unbounded error growth over long time horizons.

2.5 Examples

To highlight the key aspects of our analysis and algorithm, we work with toy problems. In two dimensions, chaotic behaviour requires studying forced or non-autonomous systems like the Duffing equation. According to the Poincaré–Bendixson Theorem, a solution to an autonomous two-dimensional system that remains bounded for all future time is either periodic or approaches a periodic solution.

In contrast, three-dimensional autonomous systems can exhibit chaotic behaviour. The most famous example of deterministic chaos is the Lorenz system, which is believed to model the complexity seen in more realistic systems such as those arising in hydrodynamics, weather prediction, and chemical reactions. In this work, we also examine the Lorenz-96 equations and the Kuramoto–Sivashinsky equations.

Many well-known systems, including the Lorenz equations, are dissipative. Dissipative systems often arise from spatial discretization of partial differential equations (PDEs). Examples include the Cahn–Hilliard equation, the two-dimensional

Navier-Stokes equations, the complex Ginzburg–Landau equation, and the Kuramoto–Sivashinsky equation. These systems satisfy an infinite-dimensional analogue of the condition (2.24) [130].

2.5.1 Logistic equation

Consider the equation

$$u'(t) = u\,(1-u). \tag{2.26}$$

Take $u(0) = u_0$. Then (2.26) has solution

$$u(t) = \frac{u_0}{u_0 + (1-u_0)\,e^{-t}}, \tag{2.27}$$

and hence defines a dynamical system on $[0,\infty)$. The set $\mathcal{X} = [0,1]$ is invariant for this problem. Take $u_0 \in \mathcal{X}$. Then $u(t) \in \mathcal{X}$, which implies $\mathcal{X} \supseteq S(t)\mathcal{X}$. Also,

$$u_0 = \frac{u}{u + (1-u)\,e^{t}}. \tag{2.28}$$

So, if $u(t) \in \mathcal{X}$ then $u_0 \in \mathcal{X}$, which implies $\mathcal{X} \subseteq S(t)\mathcal{X}$.

2.5.2 Lorenz equations

First introduced by Lorenz in 1963 [85], this system is a canonical example of deterministic chaos that shares key features with tokamak turbulence. Its equations describe the convective motion of a fluid between two parallel infinite horizontal plates in two dimensions, where the lower plate is heated and the upper one is cooled. This model is derived from a Fourier–Galerkin truncation of a Boussinesq approximation of the Navier-Stokes equations coupled with thermal convection. As such, the resulting system has some commonalities with the 2D Navier-Stokes equations [45]. The equations are

$$\begin{aligned} x' &= \sigma(y-x), \\ y' &= x(\rho - z) - y, \\ z' &= xy - \beta z, \end{aligned} \tag{2.29}$$

with $\sigma, \beta > 0$ and $\rho > 1$. The variable x is proportional to the intensity of the convective motion, y to the temperature difference between the rising and falling parts of the fluid, and z to the deviation of the temperature profile from its equilibrium, which increases linearly with height. Since the equations are polynomials, they are locally Lipschitz. Lorenz realized that for the canonical parameter values $\rho = 28$, $\sigma = 10$, and $\beta = 8/3$, this system suffers from sensitive dependence on initial conditions.

Also, a combination of results culminating with the work of Tucker [135] shows that for the same parameters, the system (2.29) admits a chaotic attractor that is uniformly hyperbolic, except for a singularity due to the attractor containing the point of equilibrium identified by $x = y = 0$ and $z = -\rho$. Such an attractor supports an invariant probability measure with a positive LE, $\lambda \approx 0.91$.

2.5.3 Lorenz-96 equations

Lorenz designed this system in 1996 as a test problem for numerical weather prediction [87]. It describes the evolution of an atmospheric quantity of interest (e.g., temperature, pressure or vorticity) on a circle of latitude of the Earth, represented by a periodic lattice of N sites i, and undergoing advection, dissipation, and external forcing, respectively simulated by a quadratic, a linear, and a constant term. This system was not designed to be realistic, as the aim of Lorenz was just to study fundamental issues regarding the predictability of the atmosphere. It belongs to the class of forced dissipative systems with quadratic nonlinear terms and can be considered as one of the simplest of them. For $i = 1, \ldots, N$, the equations are

$$\begin{aligned} x_i' &= x_{i-1}\left(x_{i+1} - x_{i-2}\right) - x_i + F, \\ x_{i-N} &= x_{i+N} = x_i, \end{aligned} \tag{2.30}$$

where $N \in \mathbb{N}$ is the dimension and $F \in \mathbb{R}$ is the forcing parameter. Their dynamics are completely described by N and F, which in fact acts as a bifurcation parameter when $N \geq 4$. In particular, $N = 40$ and $F = 8$ are standard choices in the literature. Clearly, $x_i = F$ for all i is an equilibrium solution for all N and F. Such a solution is stable for $-1/2 < F < 8/9$, and, as F increases, the system goes through multiple bifurcations, exhibiting spatial and temporal periodicity, until solutions become chaotic [136]. Also, due to the symmetry of the model, all variables should have the same statistical behaviour, so that correlations between two variables x_i and x_j only depends on the difference $i - j$ of their indices. These reasons, together with the fact that the system has multiple positive Lyapunov exponents, have made (2.30) quite popular for the study general aspects of chaos, without regard to its phenomenological justification [46].

2.5.4 Kuramoto–Sivashinsky equation

The Kuramoto–Sivashinsky (KS) equation is one of the most studied models of spatio-temporal dynamics in spatially extended systems. In one space dimension, it is written

as

$$u_t = -uu_x - u_{xx} - u_{xxxx}, \tag{2.31}$$

for $x \in [0, L]$. It has been independently derived in the context of several extended physical systems driven far from equilibrium by intrinsic instabilities, including instabilities of dissipative trapped ion modes in plasmas [79]; to describe unstable drift waves in plasmas [25]; in the context of angular phase-turbulence for a system of reaction-diffusion equations modelling the Belousov–Zhabotinsky reaction [77]; in modelling small thermally diffusive, interfacial instabilities in laminar flame fronts [122]; in the context of small fluctuations from a reference Poiseuille flow in fluid films on inclined surfaces [121]; to describe long waves on the interface between two viscous fluids [69].

Mathematically, the second-order derivative term, due to its sign, acts as an energy source, and thus has a destabilizing effect. The nonlinear term uu_x, however, transfers energy from low to high wave numbers, where the hyper-dissipative, stabilizing fourth-order derivative term dominates. The interplay between these contrasting features gives rise to a very rich dynamical behaviour. It has been proven that, for all L-periodic initial conditions, (2.31) has a unique smooth solution that depends continuously on its initial datum [129]. The size of the solution is known to be no greater than $\mathcal{O}(L^{3/2})$ [17, 103], although numerical experiments suggest that the optimal exponent is $1/2$ [138]. Other studies have focussed on how the solution changes with the system size L, which plays the role of a bifurcation parameter. For $L < 2\pi$, the trivial solution $u \equiv 0$ is stable. As L increases beyond 2π, the trivial solution becomes unstable, transitioning from intermittent disorder and travelling waves to chaos and turbulence for sufficiently large L.

Although in the turbulent regime they only hold in a time-averaged sense, (2.31) presents several symmetries [28]. For example, one such symmetry is Galilean invariance, i.e. if $u(x, t)$ is a solution then so is $u(x - ct, t) + c$, where c is an arbitrary constant. On periodic domains, we can impose a zero mean condition without loss of generality to remove this invariance. Furthermore, spatial periodicity makes it convenient to work in the Fourier space, which results in replacing (2.31) with a system of infinitely many ODEs.

2.6 Runge–Kutta methods

The two widely used families of numerical methods for solving ODEs are linear multistep methods and Runge–Kutta methods. In this section, we focus on the

latter, discussing key aspects such as accuracy and stability. Although our focus is on Runge–Kutta methods, most of the concepts presented here also apply to linear multistep methods.

A Runge–Kutta method is a kind of one-step method. This means it computes a sequence of approximations u_n of the true solutions $u(t_n)$ at discrete time steps t_n ($n \in \mathbb{N}$). The general form of a one-step method solving (2.12) is

$$u_{n+1} = u_n + h\Phi(u_n) \tag{2.32}$$

where $h = t_{n+1} - t_n$ is the step size, and Φ is a function that approximates the behaviour of the differential equation over the interval. An s-stage Runge–Kutta method with coefficients a_{ij}, b_j and c_i ($i, j = 1, \ldots, s$) is defined by

$$u_{n+1} = u_n + h\sum_{i=1}^{s} b_i f(t_n + hc_i, k_i), \tag{2.33}$$

$$k_i = u_n + h\sum_{j=1}^{s} a_{ij} f(t_n + hc_j, k_j), \tag{2.34}$$

where the k_i are called stages. Note that another, algebraically equivalent definition is

$$u_{n+1} = u_n + \sum_{i=1}^{s} b_i k_i, \tag{2.35}$$

$$k_i = hf\left(t_n + hc_i,\ u_n + \sum_{j=1}^{s} a_{ij} k_j\right). \tag{2.36}$$

The coefficients of a Runge–Kutta method are often conveniently summarized in a Butcher tableau

$$\begin{array}{c|c} c & A \\ \hline & b^\top \end{array} \tag{2.37}$$

given by the matrix $A = (a_{ij})_{i,j=1}^{s}$ and the vectors $b = (b_j)_{j=1}^{s}$ and $c = (c_i)_{i=1}^{s}$. A Runge–Kutta method is called explicit when $a_{ij} = 0$ for all $i \leq j$, such that A is strictly lower-triangular, and thus (2.34) can be solved recursively. Otherwise, (2.34) is a system of nonlinear equations for which there is no general explicit solution formula, and the method is called implicit. For implicit Runge–Kutta methods, we can solve (2.34) using Newton's method or related algorithms, provided that we choose a step size h small enough. For a more detailed discussion of these methods, see the monographs [64, 65].

A useful tool to evaluate the accuracy of a one-step method is that of the local truncation error, which measures how accurately the numerical solution approximates the exact solution in each step. It is defined as

$$\tau_n(h) = \frac{u(t_{n+1}) - u(t_n)}{h} - \Phi(u_n). \tag{2.38}$$

A one-step method is said to be consistent if $\tau = \sup_{n\in\mathbb{N}} |\tau_n| \to 0$ as $h \to 0$. Moreover, the method is said to have order p if its local truncation error satisfies $\tau = \mathcal{O}(h^{p+1})$. A necessary and sufficient condition for a Runge–Kutta method to be consistent, and thus of order $p = 1$, is that $\sum_{i=1}^{s} b_i = 1$, which we assume holds always. The order p of a Runge–Kutta method is determined by its coefficients (A, b, c), which must satisfy a set of algebraic constraints known as the order conditions. These conditions are derived by matching the Taylor series expansion of the numerical solution with that of the true solution. For example, to achieve order $p = 2$, the coefficients must satisfy not only the consistency condition $\sum_{i=1}^{s} b_i = 1$, but also $\sum_{i=1}^{s} b_i c_i = 1/2$. Higher orders require satisfying an exponentially growing number of increasingly complex conditions. In practice, Runge–Kutta methods often use adaptive time-stepping strategies (in which case they are called embedded Runge–Kutta methods [65]) that monitor the local truncation error by computing two solutions of different orders within a single iteration and dynamically adjust the step size h to maintain the desired level of accuracy.

While accuracy measures how well a method approximates the exact solution locally, stability determines how numerical errors behave as the computation progresses. Small errors introduced at each step can build up, and in some cases, make the solution diverge. There are several stability concepts in the literature, the simplest of which is the classical A-stability for linear problems [29]. It is based on Dahlquist's equation $u'(t) = \mu u(t)$, with $\mu \in \mathbb{C}$. A one-step method applied to this problem yields the difference equation $u_{n+1} = \mathcal{R}(z)\, u_n$, where $z = h\mu$ and $\mathcal{R}$ is known as the stability function of the method. For a Runge–Kutta method,

$$\mathcal{R}(z) = 1 + z b^\top (I_s - zA)^{-1} \mathbf{1}_s, \tag{2.39}$$

with $I_s \in \mathbb{R}^{s\times s}$ and $\mathbf{1}_s \in \mathbb{R}^s$ being the identity matrix and the column vector of all ones, respectively. A Runge–Kutta method is called A-stable if $|\mathcal{R}(z)| \leq 1$ whenever $\Re(z) \leq 0$. A natural extension is to consider the linear non-autonomous problem $u'(t) = M(t)\, u(t)$, with $M : [0, \infty) \to \mathbb{R}^{d\times d}$. Setting $Z_{n,i} = hM(t_n + hc_i)$ and Z_n to

be the block-diagonal matrix with entries $Z_{n,1}, \ldots, Z_{n,s}$ on the diagonal, the resulting difference equation becomes $u_{n+1} = \mathcal{R}(Z_n)\, u_n$, where

$$\mathcal{R}(Z_n) = I_d + (b^\top \otimes I_d)\big(I_s - Z_n(A \otimes I_d)\big)^{-1} Z_n(\mathbf{1}_s \otimes I_d). \tag{2.40}$$

However, practical implementations make use of the approximation $M(t_n + hc_i) \approx M(t_n)$, so that (2.40) actually simplifies to

$$\mathcal{R}(Z_n) = I_d + (b^\top \otimes I_d)(I_s - A \otimes Z_n)^{-1}(\mathbf{1}_s \otimes Z_n), \tag{2.41}$$

where $Z_n = hM(t_n)$. We will use (2.41) in the proof of Theorem 3.3.2.

One result that we will make use of in Section 3.3 is the following.

Theorem 2.6.1 ([18, Lemma 319A])**.** Let $f : \mathbb{R}^d \to \mathbb{R}^d$ in (2.12) be a globally Lipschitz function with constant Λ_f. Let $u_0, v_0 \in \mathbb{R}^d$ be two input values to a step with the Runge–Kutta method (A, b, c), using step size $h \le h_0$, where $h_0\Lambda_f\rho(\|A\|) < 1$. Let u_1, v_1 be the corresponding output values. Then

$$\|u_1 - v_1\| \le (1 + h\Lambda^*)\|u_0 - v_0\|, \tag{2.42}$$

where

$$\Lambda^* = \Lambda_f \, \|b^\top\| \, (I - h_0\Lambda_f\|A\|)^{-1}\mathbf{1}. \tag{2.43}$$

Also, note that

$$(1 + h\Lambda^*)^n \le \exp(\Lambda^* hn). \tag{2.44}$$

We conclude by mentioning that numerous extensions of the basic Runge–Kutta framework have been developed to address challenges in numerical integration. Implicit-explicit (IMEX) Runge–Kutta methods combine explicit and implicit schemes to efficiently handle systems with both stiff and non-stiff components [5, 104]. Non-stiff terms are treated explicitly to reduce computational cost, while stiff terms are integrated implicitly to maintain stability.

We will employ an IMEX scheme for solving the Kuramoto–Sivashinsky equation in all experiments. Exponential Runge–Kutta methods are designed for stiff systems where stiffness primarily arises from linear terms. These methods integrate the linear part exactly using the matrix exponential while handling the nonlinear part explicitly [68].

2.7 Parareal

Parareal is one of the most well-known time-parallel algorithms. Its straightforward design has allowed practitioners to study its properties from various perspectives, most notably by interpreting it as a predictor-corrector algorithm, a deferred correction method, a multiple shooting technique, and a multigrid method. In this section, we follow Gander and Vandewalle [59] and present Parareal as an inexact Newton's method.

Consider an IVP of form (2.12) defined over a finite time domain of integration $[0, T]$. Let S denote the evolution semigroup from Section 2.2, so that the solution can be written as $u(t) = S(t)\, u_0$. We divide the time domain into N time chunks $[T_{n-1}, T_n]$ of size $\Delta T_n = T_n - T_{n-1}$ $(n = 1, \dots, N)$ such that $0 = T_0 < \cdots < T_N = T$. To keep it simple, we assume that all time chunks have equal length, i.e. $\Delta T_n = \Delta T = T/N$ for all n. Thus, we obtain N subproblems of form

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t}\{S(t)\, U_{n-1}\} = f\big(t, S(t)\, U_{n-1}\big), \quad t \in [T_{n-1}, T_n], \\ &S(T_{n-1})\, U_{n-1} = U_{n-1}, \end{aligned} \tag{2.45}$$

where we have made explicit the dependence of S on the initial condition U_{n-1} of the n-th subproblem. Our goal is to solve each of the subproblems independently and in parallel, while iteratively reducing the discontinuities between the time chunks, so as to patch together the resulting subsolutions into a unique continuous solution. We cannot hope to achieve perfect parallelism because it would require complete independence among time chunks, which in turn would violate the causality principle. Yet, as we will see, there are compromises that can be made.

As mentioned, we want to obtain a unique continuous solution. Continuity requires $U_0 = u_0$ and $U_n = S(T_n)\, U_{n-1}$ for all $n = 1, \dots, N$[5]. This is equivalent to solving the root-finding problem

$$\mathcal{E}(U) = \begin{bmatrix} U_0 - u_0 \\ U_1 - S(T_1)\, U_0 \\ \vdots \\ U_N - S(T_N)\, U_{N-1} \end{bmatrix} = 0, \tag{2.46}$$

where U is a vector containing all the time chunks' initial conditions, i.e. $U = [U_0^\top, \dots, U_N^\top]^\top$. We can solve (2.46) using Newton's method, which reads as

$$U^k = U^{k-1} - \mathcal{D}\mathcal{E}(U^{k-1})^{-1} \mathcal{E}(U^{k-1}), \tag{2.47}$$

[5]We do not need $n = N$, but it will simplify the notation later on.

with $\mathcal{DE}$ denoting the Jacobian of $\mathcal{E}$. Since

$$[\mathcal{DE}]_{ij} = \delta_{ij} - \frac{\partial}{\partial U_j}\{S(T_i)U\}_{i-1}\Big|_{U_{i-1}}, \tag{2.48}$$

where δ_{ij} is the Kronecker delta, (2.47) can be rearranged into

$$\begin{aligned} &U_0^k = u_0, \\ &U_n^k = S(T_n)\, U_{n-1}^{k-1} + \frac{\partial}{\partial U_{n-1}}\{S(T_n)U\}_{n-1}\Big|_{U_{n-1}^{k-1}} (U_{n-1}^k - U_{n-1}^{k-1}), \quad n = 1, \dots, N. \end{aligned} \tag{2.49}$$

We have thus divided the solution process of the original problem (2.12) into two steps: first solve (2.45) for every time chunk, then impose continuity via (2.49). By assigning each time chunk to a different processor, so that N is also equal to the number of available processors, we can solve each subproblem of (2.45) in parallel. The trade-off is that continuity must be imposed sequentially.

In general, however, S and its derivatives are not available analytically. Instead, Parareal approximates those terms using two solvers: a coarse solver G[6], cheap to compute but inaccurate, to be applied sequentially, and a fine solver F, accurate but expensive, to be applied to each time chunk in parallel. More formally, we define $F(T_{n-1}, T_n, U_{n-1})$ and $G(T_{n-1}, T_n, U_{n-1})$ – which we will shorten to $F(U_{n-1})$ and $G(U_{n-1})$ for convenience – to be the fine and coarse numerical solutions, respectively, of (2.45) at T_n starting from U_{n-1} at T_{n-1}. We then take the approximations

$$S(T_n; U_{n-1}^{k-1}) \approx F(U_{n-1}^{k-1}), \tag{2.50}$$

$$\frac{\partial S}{\partial U_{n-1}}(T_n; U_{n-1}^{k-1})\,(U_{n-1}^k - U_{n-1}^{k-1}) \approx G(U_{n-1}^k) - G(U_{n-1}^{k-1}), \tag{2.51}$$

to obtain the Parareal iteration

$$\begin{aligned} &U_0^k = u_0, \\ &U_n^k = G(U_{n-1}^k) + F(U_{n-1}^{k-1}) - G(U_{n-1}^{k-1}), \quad n = 1, \dots, N. \end{aligned} \tag{2.52}$$

An outline of Parareal is given in Algorithm 1.

In conclusion, two crucial points need to be addressed. Firstly, due to the approximation (2.50), it is not possible to recover the exact analytical solution. Instead, Parareal aims to converge to the fine solution, i.e. the numerical solution obtained by applying the fine solver F across the whole time domain. Therefore, when it comes to

[6]From French *grossier*, which means "coarse".

Algorithm 1 Parareal

```
# coarse guess
U_0^0 = u_0
for n = 1, ..., N do
    U_n^0 = G(U_{n-1}^0)
end for
for k = 1, 2, ... until convergence do
    # parallel step
    for n = 1, ..., N do in parallel
        Û_n^k = F(U_{n-1}^{k-1})
    end for
    # correction step
    U_0^k = u_0
    for n = 1, ..., N do
        U_n^k = G(U_{n-1}^k) + Û_n^k − G(U_{n-1}^{k-1})
    end for
end for
```

solving an IVP, Parareal cannot outmatch the accuracy and stability of the fine solver on which it is based. In particular, this fine solution is also the root

$$U^* = \begin{bmatrix} U_0^* \\ U_1^* \\ \vdots \\ U_N^* \end{bmatrix} = \begin{bmatrix} u_0 \\ F(U_0^*) \\ \vdots \\ F(U_{N-1}^*) \end{bmatrix} \tag{2.53}$$

of the root-finding problem

$$\mathcal{E}_F(U) = \begin{bmatrix} U_0 - u_0 \\ U_1 - F(U_0) \\ \vdots \\ U_N - F(U_{N-1}) \end{bmatrix} = 0. \tag{2.54}$$

Notably, (2.54) mirrors (2.46), with F taking the place of S.

Secondly, let us examine the behaviour of Parareal as the number of iterations increases. After one iteration, the solution converges up to T_1 because $U_1^1 = F(U_0^1) = F(u_0)$. Similarly, the solution converges up to T_n for $n \leq k$ after k iterations. Therefore, the solution converges over the whole time domain after $k = N$ iterations. This holds true regardless of how inaccurate and unstable the coarse solver is. However, converging in N iterations is equivalent to achieving no parallel speedup. The reason is that the total wall-clock time required would be identical to that taken by running the simulation sequentially, or even longer due to the additional time spent for coarse

solving and communication between processors. Therefore, it is imperative to achieve sufficient convergence in $k \ll N$ iterations.

2.7.1 Algebraic interpretation

There is an alternative interpretation of Parareal due to Maday and Turinici [91] which is more algebraic in nature. Their perspective allows us to emphasize different properties from those discussed above.

By casting the root-finding problem (2.46) in operator form, we get

$$A_F U = \begin{bmatrix} I & & & \\ -F & \ddots & & \\ & \ddots & \ddots & \\ & & -F & I \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_N \end{bmatrix} = \begin{bmatrix} u_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = b. \tag{2.55}$$

Defining the operator matrix

$$A_G = \begin{bmatrix} I & & & \\ -G & \ddots & & \\ & \ddots & \ddots & \\ & & -G & I \end{bmatrix}, \tag{2.56}$$

it follows that Parareal is equivalent to the following preconditioned Richardson iteration:

$$U^{k+1} = U^k - A_G^{-1}(A_F U^k - b). \tag{2.57}$$

Iteration (2.57) suggests an important aspect of Parareal. Even though F and G need not be based on the same time-stepping scheme, the action of G should closely approximate that of F. This ensures that A_G^{-1} serves as a good preconditioner for A_G, meaning that the matrix $A_G^{-1}A_F$ is close to the identity matrix and that A_G is easier to invert than A_F. Nevertheless, preconditioning alone is not enough to achieve time parallelization: one must also be able to apply the fine solver in parallel. In this context, this is feasible because A_F is (block) bidiagonal and U^k is known at the k-th iteration.

2.7.2 Efficiency

One important metric to characterize the behaviour of a parallel algorithm is the parallel efficiency η, defined as the ratio between the computational cost of obtaining a

serial solution over that of obtaining a parallel solution, further divided by the number of processors used:

$$\eta = \frac{\text{time taken on 1 processor (serial execution time)}}{p \times \text{time taken on } p \text{ processors (parallel execution time)}}.$$

It should be noted that, although we have used processors as our computing units here, this may not always be the best choice in today's computing landscape. Nowadays, there are many alternatives, such as containers, virtual machines, cores, or a mix of different components. Also, modern machines have complex configurations where many elements beyond just processors affect algorithmic performance. So, the choice of what computing units should be considered depends heavily on the machine's architecture and on the algorithm's implementation. Nevertheless, this assumption does not significantly affect our conclusions. Another point to consider is how we define the serial solution. There are at least two choices: either the execution time of the parallel algorithm on one processor, or the execution time of the best sequential algorithm available. In the context of Parareal, it is straightforward to choose the solution obtained with the fine solver.

Maday gives a detailed discussion of the parallel efficiency of Parareal in [89]. Assume that the number of processors p available to Parareal is equal to the number of time chunks N, that is $p \equiv N$. Also, let C_{S} be the computational cost of obtaining a serial solution and C_{P} be that of obtaining a parallel solution. Neglecting communication overhead, we get

$$\eta = \frac{C_{\mathrm{S}}}{N C_{\mathrm{P}}}. \tag{2.58}$$

Let h be the time step size of the fine solver, ξ be the ratio of the coarse time step size over the fine time step size, and C_F and C_G be the costs associated to one step of the fine and coarse integrations, respectively. The serial cost then becomes

$$C_{\mathrm{S}} = \frac{C_F T}{h}. \tag{2.59}$$

Next, to work out the parallel cost of Parareal, we need to look at its specific implementation. A straightforward implementation would be to first run the coarse solver serially, then send the computed coarse points to the relevant processors, and finally run the fine solver on each processor in parallel. In this scenario, the cost of Parareal becomes

$$C_{\mathrm{P}} = K \frac{C_F \Delta T}{h} + K \frac{C_G T}{\xi h} = C_{\mathrm{S}} \frac{K(1 + \zeta N)}{N}, \tag{2.60}$$

where K is the total number of iterations needed to achieve the desired convergence, and $\zeta = C_G/(\xi C_F)$ is the ratio of the time taken by G to the time taken by F to integrate the same problem over the same time interval. As a result, the parallel efficiency of Parareal becomes

$$\eta = \frac{1}{K(1+\zeta N)}. \tag{2.61}$$

In the best-case scenario, G is so much cheaper to apply than F that we can take the limit for $\zeta \to 0$, so that η decreases as $\mathcal{O}(1/K)$. But even then, the efficiency can at most get up to 50%, which we have when $K = 2$. In practice, the cost of applying G is not negligible, which further restricts this bound. This is an issue not exclusive to Parareal, as it afflicts many time-parallel techniques to different extents, both due to the sequential nature of the problem, which makes time parallelization a difficult task per se, and as a consequence of Amdahl's law, whereby the non-parallelizable portion of a generic parallel algorithm often has a disproportionate effect on its overall speedup [66]. Once again, therefore, it is extremely important to achieve convergence within a few iterations.

One can improve (2.61) by observing that, since the solution has converged at T_n for $n \leq k$ at the k-th iteration, then the fine and coarse integrations only needs to take place over the interval $[T_k, T]$. This observation leads to

$$C_{\mathrm{P}} = \left(\frac{C_F T}{Nh} + \frac{C_G T}{\xi h}\right) \frac{\overbrace{N + (N-1) + \cdots + (N-K+1)}^{K \text{ terms}}}{N} \tag{2.62}$$

$$= C_{\mathrm{S}} \frac{1+\zeta N}{N} \sum_{i=1}^{K} \frac{N-i+1}{N} \tag{2.63}$$

$$= C_{\mathrm{S}} \frac{K(1+\zeta N)}{N} \left(1 - \frac{K-1}{2N}\right), \tag{2.64}$$

such that

$$\eta = \frac{1}{K(1+\zeta N)\left(1 - \frac{K-1}{2N}\right)}. \tag{2.65}$$

Although this results in an increased parallel efficiency, we can obtain a more substantial improvement by carefully reallocating the tasks (fine and coarse integration, correction) of Parareal. For example, Aubanel [6] optimizes the efficiency by creating a computational pipeline for the sequential coarse correction step. Rather than having a single master process compute the entire coarse correction chain, the task is passed from one processor to the next. Once processor $n-1$ has computed its value U_{n-1}^k, it passes this result to the now-idle processor n, which then computes the next coarse

step in the sequence. While the coarse correction itself remains sequential, as each step depends on the previous one, this pipelining strategy uses otherwise idle resources to perform the work, effectively hiding some of the serial cost. Following this approach, one finds that

$$C_{\mathrm{P}} = K\left(1 - \frac{K-1}{2N}\right)\left(C_F\frac{T}{Nh} + C_G\frac{T}{N\xi h}\right) + C_G\frac{T}{\xi h}\left(1 - \frac{1}{N}\right), \tag{2.66}$$

such that

$$\eta = \frac{1}{K\left(1 - \frac{K-1}{2N}\right)(1+\zeta) + \zeta(N-1)}. \tag{2.67}$$

Comparing (2.67) with either (2.61) or (2.65), we can see that the parallel efficiency of the task-scheduled version loses its dependence on K as $N \to \infty$. Although (2.67) still retains its dependence on ζ, this trick enables us to conceivably halt the $1/K$ decay by increasing the number of working processors.

Similar improvements can be obtained in other ways too – see [39] for such an example.

2.7.3 Convergence

We present two important theorems from the literature about the convergence of Parareal. The first result, due to Staff and Rønquist [125] and later generalized by Gander and Vandewalle [59], considers the case of Parareal when applied to Dahlquist's equation

$$\begin{aligned} u'(t) &= \mu u(t), \quad \mu \in \mathbb{C}, \\ u(0) &= 1. \end{aligned} \tag{2.68}$$

Theorem 2.7.1 ([59, Theorem 4.5])**.** Let $\Delta T = T/N$ and $T_n = n\Delta T$ for $n = 0, \ldots, N$. Let F be the exact integrator of (2.68) and G a one-step method with stability function $\mathcal{R}_G(z)$ such that $z = \mu\Delta T$. Then, at the k-th iteration of Parareal, we have

$$\|U^k - U^*\|_\infty \leq g(\mathcal{R}_G(z))|e^z - \mathcal{R}_G(z)|^k \|U^0 - U^*\|_\infty, \tag{2.69}$$

where

$$g(x) \leq \begin{cases} \min\left\{\left(\dfrac{1-|x|^{N-1}}{1-|x|}\right)^k, \binom{N-1}{k}\right\}, & |x| < 1, \\ |x|^{N-k-1}\binom{N-1}{k}, & |x| \geq 1, \end{cases} \tag{2.70}$$

and $\|U^k - U^*\|_\infty = \max_{0\leq n\leq N}|U_n^k - U_n^*| = \max_{1\leq n\leq N}|U_n^k - U_n^*|$.

Theorem 2.7.1 holds even if F is an approximate solver. For example, if the fine solver is composed of L Runge–Kutta steps, each with its own stability function $\mathcal{R}(z)$, then it is enough to substitute e^z with the "stability function" $\mathcal{R}_F(\mu\Delta T) = \mathcal{R}(\mu\Delta T/L)^L$. Clearly, the assumption that the time-step length for G is equal to the length of each time chunk can be generalized similarly.

In [59, Theorem 4.9], Gander and Vandewalle also show that if we fix ΔT and let $N \to \infty$ then the convergence factor in (2.69) reduces to

$$\beta(z) = \frac{|e^z - \mathcal{R}_G(z)|}{1 - |\mathcal{R}_G(z)|}. \tag{2.71}$$

Definition 2.7.2. For the Parareal algorithm using a one-step method as the coarse solver and the exact solution as the fine solver, we define its stability region to be the set

$$\{z \in \mathbb{C} \,:\, |\beta(z)| \leq 1,\, |\mathcal{R}_G(z)| \leq 1\}, \tag{2.72}$$

where β is as defined in (2.71) and $\mathcal{R}_G$ is the stability function of the coarse solver.

In plain words, the stability region of Parareal is the set of points such that the convergence factor β is smaller than 1 everywhere and the coarse solver operates in its region of stability. Note that, as shown in Figure 2.1, the stability region of Parareal shrinks compared to that of its fine solver, due to the additional limitations introduced by the coarse solver. A widespread rule of thumb among practitioners is that the coarse solver affects stability and convergence rate, whereas the accuracy of the final result depends on the fine solver.

Finally, the following theorem due to Gander and Hairer [53] applies to both linear and nonlinear IVPs of form (2.12).

Theorem 2.7.3 ([53, Theorem 1])**.** Let $\Delta T = T/N$ and $T_n = n\Delta T$ for $n = 0, \ldots, N$. Let F be the exact integrator of (2.12) and G an approximate solver of order p that

1. has local truncation error bounded above by $c_3\Delta T^{p+1}$;

2. satisfies the Lipschitz condition

$$\|G(U_n) - G(V_n)\| \leq (1 + c_2\Delta T)\|U_n - V_n\|, \quad \text{for all } U_n, V_n \in \mathbb{R}^d; \tag{2.73}$$

3. can be expanded as

$$F(U_n) - G(U_n) = \sum_{i=p+1}^{\infty} \alpha_i(U_n)\Delta T^i, \tag{2.74}$$

for α_i continuously differentiable.

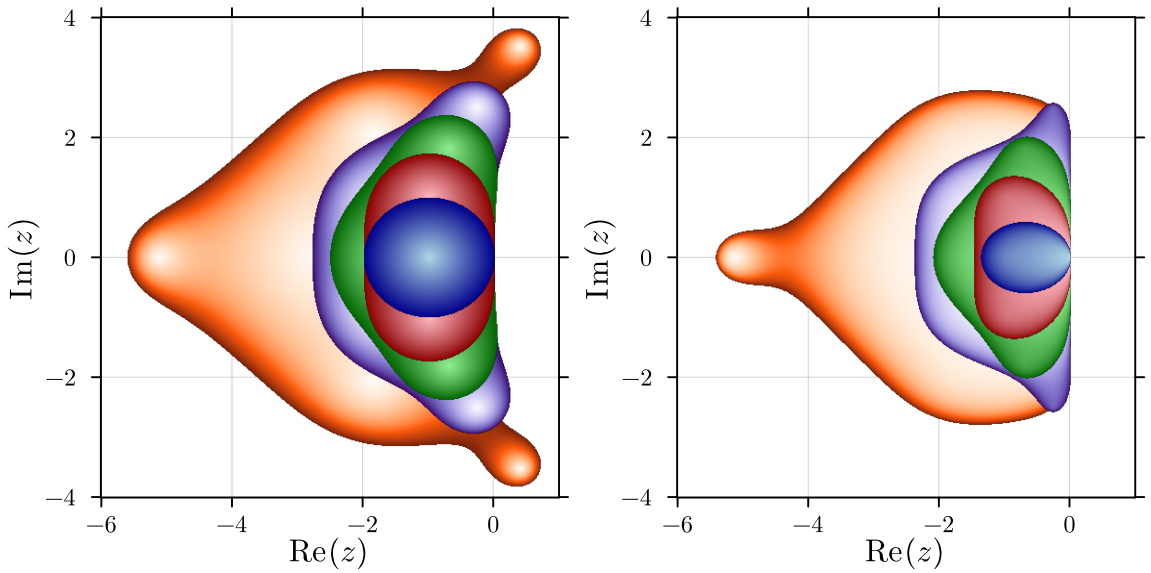

Figure 2.1: Comparison between the stability regions of five explicit Runge–Kutta methods (left) – Explicit Euler (blue), Explicit Midpoint (red), 3rd-order Heun (green), 4th-order Runge–Kutta (purple), 5th-order Butcher (orange) – and the corresponding regions for Parareal (right).

Then, at the k-th iteration of Parareal, we have

$$\|U_n^k - U_n^*\| \leq c_3 c_1^k \frac{T_n^{k+1}}{(k+1)!} e^{c_2(T_n - T_{k+1})} \Delta T^{p(k+1)}, \quad n = 1, \ldots, N, \tag{2.75}$$

where c_1 is a positive constant determined by the Lipschitz constant of α_{p+1}.

2.8 Implementation

To support our theoretical work with practical experiments, we develop a custom time-parallel solver in the Julia programming language [12].

Julia is an open-source language designed for high-performance scientific and technical computing, aiming to combine the ease of writing in dynamically typed languages like Python and Matlab with the performance of statically typed languages like Fortran and C. Leveraging LLVM's JIT compilation framework, Julia achieves single-threaded performance comparable to, and sometimes exceeding, that of C and Fortran (for appropriately written, type-stable code) [2]. Furthermore, Julia has built-in support for parallelism [109]: for insights into Julia's parallel capabilities in communication-intensive simulations, see [113].

This work produces two packages: a serial solver, `NSDERungeKutta.jl` [3], and a proof-of-concept Parareal solver, `NSDETimeParallel.jl` [4]. The goal of this im-

plementation is to provide a tool that is good enough for the numerical tests in this thesis.

However, achieving good parallel speedup has proved to be challenging. As the performance test in Figure 2.2 shows, the obtained speedup is limited. While Parareal itself is not highly scalable, the modest speedup here is mainly due to a synchronisation barrier in our simple implementation: the correction step must wait for all parallel fine solves to finish. We realise that building a production-ready, highly scalable solver is a full research project in itself. Since this thesis focuses on the theory of the algorithms, further software development lies outside its scope.

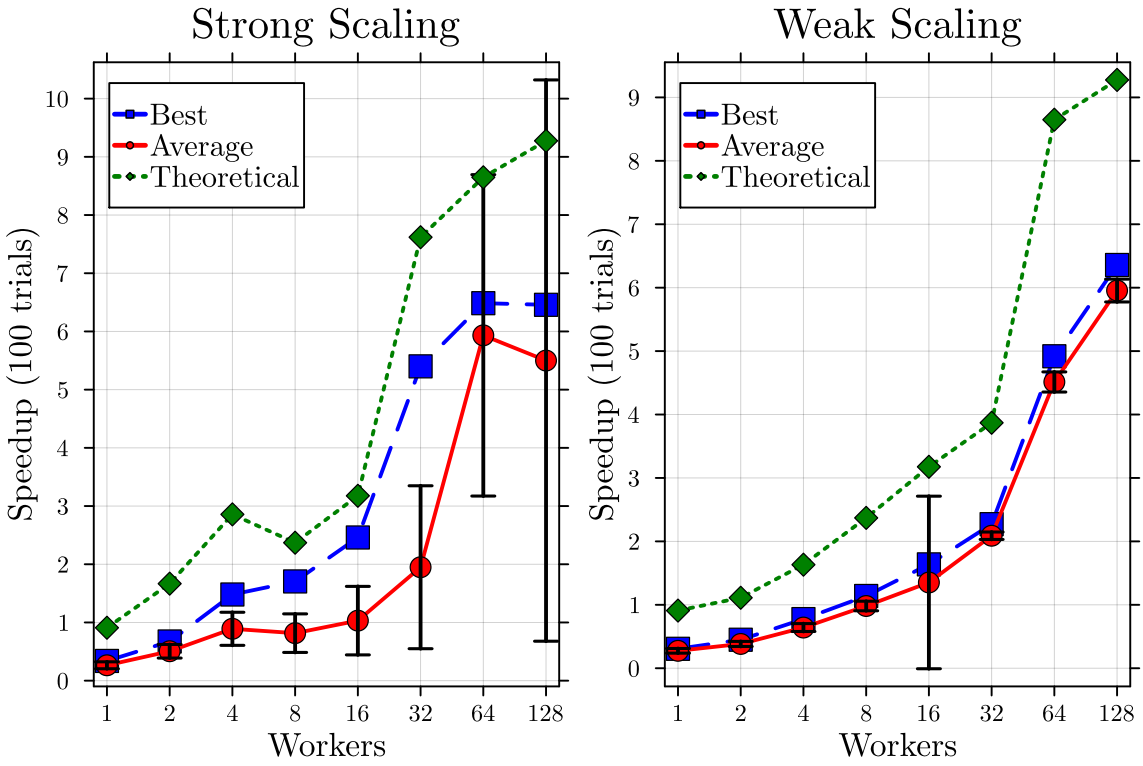

Figure 2.2: Performance of `NSDETimeParallel.jl`, a proof-of-concept Parareal solver that uses Julia's default distributed-computing capabilities (`Distributed.jl`) with different numbers of workers. The plot shows the average and best speedups across 100 trials, with error bars for standard deviations. The speedup is limited since all workers have to finish before moving on the next iteration.

Chapter 3

A new framework for convergence

As seen in Chapter 1, there is a vast body of work on time-parallel methods. However, the convergence analysis is limited to linear problems. In this chapter, we present a new framework for analysing time-parallel algorithms when applied to nonlinear (possibly chaotic) problems. We focus on time-decomposition methods, which we treat as contraction mappings that converge in a finite number of iterations. This is a powerful framework because it turns an asymptotic convergence rate into a finite-time guarantee, directly linking initial error size, convergence rate, and iteration count.

This finite-time perspective matters for two reasons. First, in practice we need to know how many iterations it takes to reach a given tolerance, not just that the algorithm will eventually converge. For time-parallel methods to be efficient, this number must be small compared to the number of processors available. Our framework gives explicit bounds that practitioners can use to predict performance before running costly simulations. Secondly, this view is crucial for when we develop the moving-window algorithm (MoWi) in Chapter 4. MoWi needs guarantees that each window can be solved within a fixed computational budget. The contraction mapping framework provides exactly these guarantees, allowing us to prove that MoWi stays 'on track' even when solving chaotic systems over long time domains.

To demonstrate how this framework applies in practice, we apply it to Parareal. We derive explicit estimates for its convergence rate on nonlinear problems, showing that the convergence factor scales as $\mathcal{O}(h^2)$ for small step sizes h. While our bounds are conservative for moderate h, they become increasingly accurate as h decreases, precisely where they matter most for turbulent simulations, which need fine time resolution. We validate these estimates through numerical experiments on chaotic test problems.

Beyond convergence rates, we tackle another practical challenge in chaotic systems: devising a stopping criterion that measures convergence without knowing the true

solution. We introduce the *proximity function*, a weighted measure that accounts for the exponential divergence of trajectories in chaotic systems by scaling discontinuities according to the Lyapunov exponent. This gives a chaos-aware stopping criterion that cuts down iteration count while keeping the same statistical accuracy, outperforming standard checks in applications where mean or bulk behaviour matters more than pointwise accuracy.

3.1 Generalities

From a general viewpoint, the goal of every time-parallel algorithm is to find a vector $U^* \in \mathbb{R}^{d(N+1)}$ that solves the root-finding problem

$$\mathcal{E}_F(U) = \begin{bmatrix} U_0 - u_0 \\ U_1 - F(U_0) \\ \vdots \\ U_N - F(U_{N-1}) \end{bmatrix} = 0, \tag{3.1}$$

where d is the dimension of the underlying problem, N is the number of time chunks, and F represents the action of the fine solver over a single time chunk. This is the same as (2.54) from the previous chapter, restated here for clarity. The vector $U = [U_0^\top, U_1^\top, \ldots, U_N^\top]^\top$ collects all the interface values between time chunks, with each $U_n \in \mathbb{R}^d$ representing the solution at the start of the n-th time chunk.

3.1.1 Existence and uniqueness of the fine solution

We assume that the fine solver defines a unique solution for the problem at hand, meaning that U^* exists and is unique. This assumption rests on several standard results from numerical analysis [128]. First, we need the underlying IVP to be well-posed, that is, it must have a unique solution on the time interval of interest. As we discussed in Section 2.2, this typically holds when f satisfies a Lipschitz condition.

Secondly, the fine solver F must be consistent and stable, so that it can approximate the true evolution map accurately. For one-step methods, as we discussed in Section 2.6, consistency means that the local truncation error vanishes as the step size $h \to 0$, while stability ensures that errors do not grow unboundedly. The Lax equivalence theorem tells us that consistency plus stability implies convergence to the true solution as $h \to 0$.

Thirdly, the numerical solution itself must be unique. For explicit methods, this is straightforward, because each step involves only function evaluations and arithmetic

operations. For implicit methods, we also need that the implicit equations at each step have a unique solution, which typically holds when h is small enough and f is Lipschitz continuous – compare with Theorem 2.6.1.

3.1.2 Challenges in chaotic systems

These assumptions become trickier for chaotic systems. Even though the IVP may still be well-posed and the numerical method convergent, the exponential sensitivity to initial conditions puts a hard limit on long-time accuracy. Even with exact arithmetic, discretization errors grow exponentially with rate λ, where $\lambda > 0$ is the largest Lyapunov exponent. As we have seen in Section 2.3, an initial error ϵ becomes $\epsilon e^{\lambda t}$ after a time t. This means that machine precision errors can grow to $O(1)$ after a time $t \sim \ln(10^{16})/\lambda \approx 37/\lambda$ [53, 86]. So, for chaotic systems, there exists a fundamental limit beyond which we cannot accurately compute individual trajectories, no matter which numerical method we use. This predictability time scales as $t_{\text{pred}} \sim \lambda^{-1} \log(1/\epsilon)$, where ϵ is our error tolerance. Note that the Lyapunov time $T_\lambda = 1/\lambda$ tells us how long it takes for errors to grow by a factor of e. Hence, to lose one decimal digit takes time $\ln(10)/\lambda \approx 2.303/\lambda$, or about 2.3 Lyapunov times.

Although we lose pointwise accuracy, numerical solutions often converge to the correct statistical properties (invariant measures, Lyapunov exponents, correlation functions, and so on). This is why long-time simulations of chaotic systems remain meaningful for studying plasma turbulence, the climate, and other physical phenomena. We will come back to this important observation in Chapter 4, where we exploit statistical convergence to develop MoWi.

So, when we say that the fine solver gives a "unique solution" U^*, we need to be careful about what we mean. For moderate time intervals, U^* approximates the true trajectory well. For long time intervals in chaotic systems, U^* represents one possible numerical trajectory that shares the correct statistical properties with the true dynamics.

3.1.3 Fixed-point formulation

Regardless of specific algorithmic details, we can view any time-parallel method as an iterative process

$$U^k = U^{k-1} + v(U^{k-1}), \tag{3.2}$$

where $v : \mathbb{R}^{d(N+1)} \to \mathbb{R}^{d(N+1)}$ is a vector field encoding the particular time-parallel algorithm. For instance, Newton's method is defined by $v(U^k) = -\mathcal{D}\mathcal{E}_F(U^k)^{-1}\mathcal{E}_F(U^k)$, where $\mathcal{D}\mathcal{E}_F$ is the Jacobian of the residual $\mathcal{E}_F$.

Starting from an initial guess U^0, this iteration generates a sequence $(U^k)_{k\in\mathbb{N}_0}$. Our goal is to understand when and how quickly this sequence converges to U^*.

Our first question is: Is U^* a fixed point of the iteration? That is, does $v(U^*) = 0$? Is it unique? For time-decomposition methods like Parareal, the answer is yes, as the following proposition shows.

Proposition 3.1.1. Assume that a time-parallel algorithm defined by iteration (3.2) has the property that, starting from any initial guess U^0, it recovers the fine solution exactly after at most N iterations. Then U^* is the unique fixed point of v.

Proof. Since the algorithm converges in at most N iterations, we have $U^k = U^*$ for all $k \geq N$. We shall show that $v(U) = 0$ if and only if $U = U^*$. First, take any iteration from k to $k+1$ where $k \geq N$. The update rule gives us $U^{k+1} = U^k + v(U^k)$, which becomes $U^* = U^* + v(U^*)$. This can only hold if $v(U^*) = 0$, so U^* is indeed a fixed point. Conversely, assume that U is any other fixed point with $v(U) = 0$. Starting the algorithm from $U^0 = U$, we get $U^1 = U^0 + v(U^0) = U$. By induction, $U^k = U$ for all $k \geq 0$. But the finite convergence property tells us that $U^N = U^*$, so we must have $U = U^*$. Hence, U^* is the only fixed point. □

3.1.4 Convergence analysis

Next, we define the *basin of attraction* of U^* as

$$\mathcal{A}(U^*) = \{U^0 \in \mathbb{R}^{d(N+1)} : U^k \to U^* \text{ as } k \to \infty\}. \tag{3.3}$$

This set contains all starting points from which the iteration (3.2) converges to U^*. Understanding this basin is crucial: the algorithm only succeeds if our initial guess U^0 lies inside $\mathcal{A}(U^*)$. While we typically use a coarse solver to generate U^0, there's no guarantee it lands in the basin of attraction, particularly for chaotic problems.

For linear problems, $\mathcal{A}(U^*)$ is often the whole space $\mathbb{R}^{d(N+1)}$. Hence, we have global convergence. For nonlinear problems, the basin may be a proper subset, and characterizing it explicitly is difficult, in general. In fact, the geometry of the basin can be surprisingly complex for chaotic problems. To show this, we plot $\|U_n - F(U_{n-1})\|$ for a Runge–Kutta fine solver F, varying the number of Runge–Kutta steps composing F. The fine solution satisfies $U_n^* = F(U_{n-1}^*)$. In other words, we show a 2D slice of the

error-contraction map for a single iteration. Figures 3.1 and 3.2 reveal that for linear problems, multiple Runge–Kutta steps barely change the contour structure, while for chaotic problems like Lorenz, the geometry becomes increasingly intricate. As the Lorenz system's dynamics are highly nonlinear, increasing the number of Runge–Kutta substeps in F makes the solver map more nonlinear (effectively a composition of high-degree polynomials). This folding and stretching produces the intricate, almost fractal geometry seen in the plots. This is why we seek sufficient conditions that guarantee at least a neighbourhood of U^* lies within $\mathcal{A}(U^*)$.

Within this basin, different initial guesses may converge at different rates. We characterize the convergence rate using the standard bound

$$\|U^k - U^*\| \leq \beta \|U^{k-1} - U^*\|^q, \tag{3.4}$$

where $\|\cdot\|$ denotes a norm on $\mathbb{R}^{d(N+1)}$, $\beta > 0$ is the convergence factor, and $q \geq 1$ determines the order of convergence. We make this precise using the following definition, adapted from [101, Chapter 9].

Definition 3.1.2. Let (U^k) be a sequence defined by (3.2) converging to U^* with $U^k \neq U^*$ for all $k > k_0$. The sequence converges uniformly with rate q^* if that is the supremum of all $q \geq 1$ such that

$$Q_q(U^k) = \limsup_{k\to\infty} \frac{\|U^k - U^*\|}{\|U^{k-1} - U^*\|^q} < \infty. \tag{3.5}$$

In particular, we say that (U^k) converges linearly if $Q_1(U^k) < 1$, superlinearly if $Q_1(U^k) = 0$, and quadratically if $Q_2(U^k) < \infty$.

Here, β in (3.4) is shorthand for Q_q. Linear convergence means that we gain roughly $\log_{10}(1/\beta)$ correct digits per iteration, whereas q-rate convergence means that the number of correct digits grows multiplicatively by a factor of q each iteration.

3.1.5 Sufficient conditions for convergence

When does the iteration (3.2) converge to U^*? For linear problems, where F is linear and hence $\mathcal{E}_F$ is affine, the iteration takes the form $U^{k+1} = (I + A)\,U^k + b$, for some matrix A and vector b, where $v(U) = AU + b$. Convergence occurs globally if and only if the spectral radius of $(I + A)$ is strictly less than 1.

For nonlinear problems, we can only guarantee sufficient (but not necessary) conditions for local convergence. The following theorem, adapted from [19], provides such sufficient conditions. For its formulation, recall that, for an open set $\Omega \subset \mathbb{R}^{d(N+1)}$

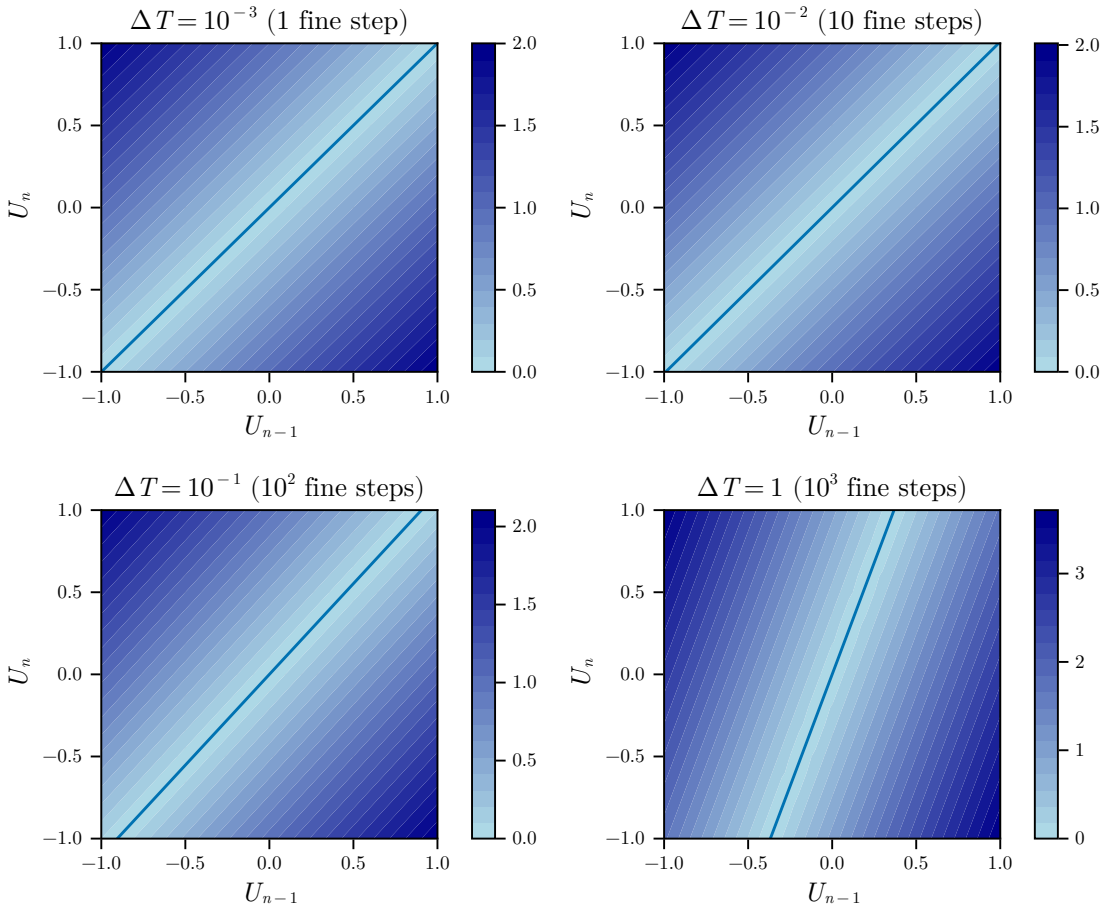

Figure 3.1: Structure of the basin of attraction for the linear Dahlquist test equation. The cyan line represents the fine solution. For this linear problem, the basin remains simple and convex even as the number of Runge–Kutta steps in the fine solver increases, illustrating a predictable convergence behaviour.

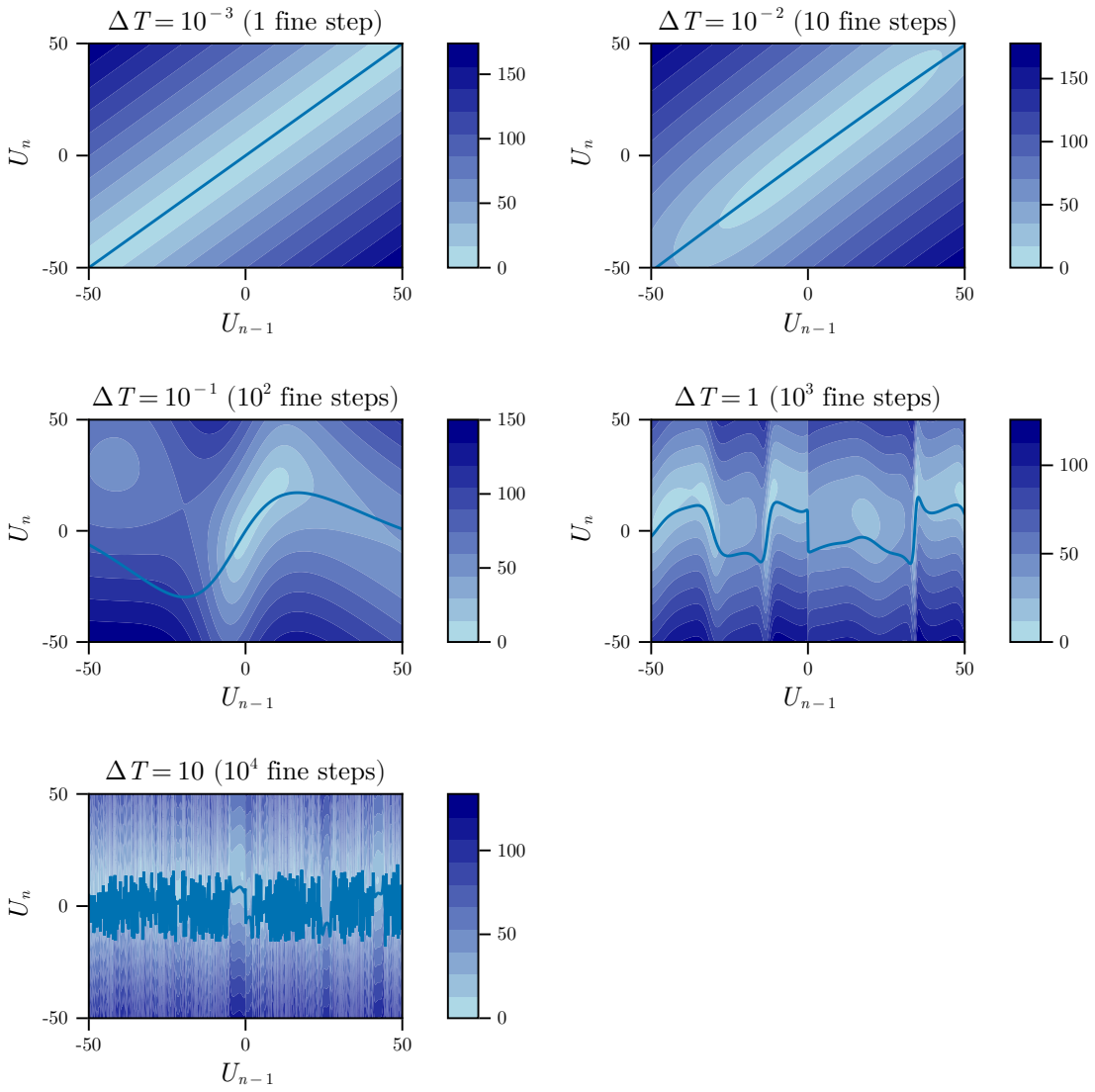

Figure 3.2: Structure of the basin of attraction for the chaotic Lorenz system. The plot setup is the same as in Figure 3.1. At very small ΔT, the integration time is short, so the dynamics are almost linear. The map from initial error to final error is nearly affine, giving a simple, smooth error map, much like the Dahlquist test case. As ΔT grows, so does the integration time and the number of fine steps, such that, in stark contrast to the linear case, the basin for this chaotic system develops a complex, fractal-like structure as the number of Runge–Kutta steps in the fine solver increases. This highlights the practical difficulty of guaranteeing convergence for chaotic dynamics.

with closure $\overline{\Omega}$, a vector field $v : \overline{\Omega} \to \mathbb{R}^{d(N+1)}$ is (Fréchet) differentiable at $U \in \Omega$ if there exists a matrix D such that

$$\lim_{\|\epsilon\| \to 0} \frac{\|v(U+\epsilon) - v(U) - D\epsilon\|}{\|\epsilon\|} = 0, \tag{3.6}$$

in which case D is unique and equal to the Jacobian $\mathcal{D}v(U)$.

Theorem 3.1.3 ([19, Theorem 2.1 and Corollary 2.2])**.** Let $\beta \in (0,1)$ and $\Omega \subset \mathbb{R}^{d(N+1)}$ be an open ball centred at U^*. Let $v : \overline{\Omega} \to \mathbb{R}^{d(N+1)}$ from (3.2) be a continuously differentiable vector field such that

1. $v(U) = 0$ if and only if $U = U^*$,
2. $\|I + \mathcal{D}v(U)\| \leq \beta$ for all $U \in \Omega$.

Then

- $U^0 \in \Omega$ implies $U^k \in \Omega$ for all $k \geq 0$,
- $U^k \to U^*$ linearly with convergence factor β.

A similar convergence result was established earlier by Dembo, Eisenstat, and Steihaug [30] in the context of inexact Newton methods solving $\mathcal{E}(U) = 0$. In their setting, one finds steps s_k such that $\mathcal{E}'(U^k)\, s_k = -\mathcal{E}(U^k) + r_k$ with $\|r_k\|/\|\mathcal{E}(U^k)\| \leq \eta_k$. Their Theorem 2.3 states that if $\eta_k \leq \eta_{\max} < \beta < 1$, where η_k controls the relative residual in their inexact Newton framework, then there exists $\epsilon > 0$ such that if $\|U^0 - U^*\| \leq \epsilon$, the sequence converges linearly with $\|U^{k+1} - U^*\|_* \leq \beta \|U^k - U^*\|_*$, where $\|V\|_* = \|\mathcal{E}'(U^*)V\|$.

3.2 Outer and inner balls

So far, we have treated the convergence rate as an asymptotic property. As defined in Definition 3.1.2, it captures the behaviour of the iterates (U^k) after some index k_0. But for time-parallel algorithms, what really matters is whether this index appears earlier or later, for two reasons. First, we consider algorithms that are guaranteed to recover the fine solution exactly after N iterations, such that $k \to N$ rather than $k \to \infty$. Secondly, parallel efficiency demands that we converge in far fewer than N iterations, ideally in $K \ll N$ steps. This raises a more practical question, that is how far the initial guess U^0 can be from the true solution U^* and still guarantee convergence within $K < N$ iterations.

3.2.1 The geometry of convergence

Suppose that our time-parallel algorithm of choice satisfies conditions 1 and 2 of Theorem 3.1.3 with U^* the solution of the fine solver and $\beta \in (0,1)$ known. Let r mark the tolerance to within which we accept the solution computed by a time-parallel algorithm as having converged. Earlier in Section 3.1, we defined the basin of attraction $\mathcal{A}(U^*)$ as the set of all initial guesses that eventually converge to U^*. We now must narrow this idea, as we want to accept any computed solution that gets within r of U^* in at most K iterations. This brings us to the K-iteration basin of attraction (with respect to r), defined by

$$\mathcal{A}_r(U^*; K) = \left\{U^0 \in \mathbb{R}^{d(N+1)} : \|U^K - U^*\| \leq r\right\}. \tag{3.7}$$

Unlike $\mathcal{A}(U^*)$, which tells us about asymptotic convergence, this set contains only those starting points from which we can reach tolerance r in at most K iterations. By construction, $\mathcal{A}_r(U^*; K) \subseteq \mathcal{A}(U^*)$.

Instead of trying to characterize $\mathcal{A}_r(U^*; K)$ exactly – which may have intricate geometry, especially for chaotic systems – we focus on balls, which give us simple, computable bounds we can work with in practice. Let R be the radius of the largest ball around U^* that fits fully inside $\mathcal{A}_r(U^*; K)$:

$$R := \sup\{\rho > 0 : B_\rho(U^*) \subseteq \mathcal{A}_r(U^*; K)\}. \tag{3.8}$$

Using the notation from Section 2.1, we define

- the *inner ball* to be $B_r(U^*) = \{U \in \mathbb{R}^{d(N+1)} : \|U - U^*\| < r\}$;
- the *outer ball* to be $B_R(U^*) = \{U \in \mathbb{R}^{d(N+1)} : \|U - U^*\| < R\}$.

By construction, $B_r(U^*) \subseteq B_R(U^*) \subseteq \mathcal{A}_r(U^*; K)$. The outer ball is the largest ball we can guarantee lies entirely within the K-iteration basin. Any initial guess inside $B_R(U^*)$ will reach the inner ball $B_r(U^*)$ within K iterations. Figure 3.3 illustrates how these sets relate to each other.

3.2.2 Quantifying the outer radius

How big can we make R relative to r? The convergence bound (3.4) gives us the answer. For convenience, let $S_K(q) = \sum_{i=0}^{K-1} q^i$. After K iterations,

$$\|U^K - U^*\| \leq \beta^{S_K(q)} \|U^0 - U^*\|^{q^K}. \tag{3.9}$$

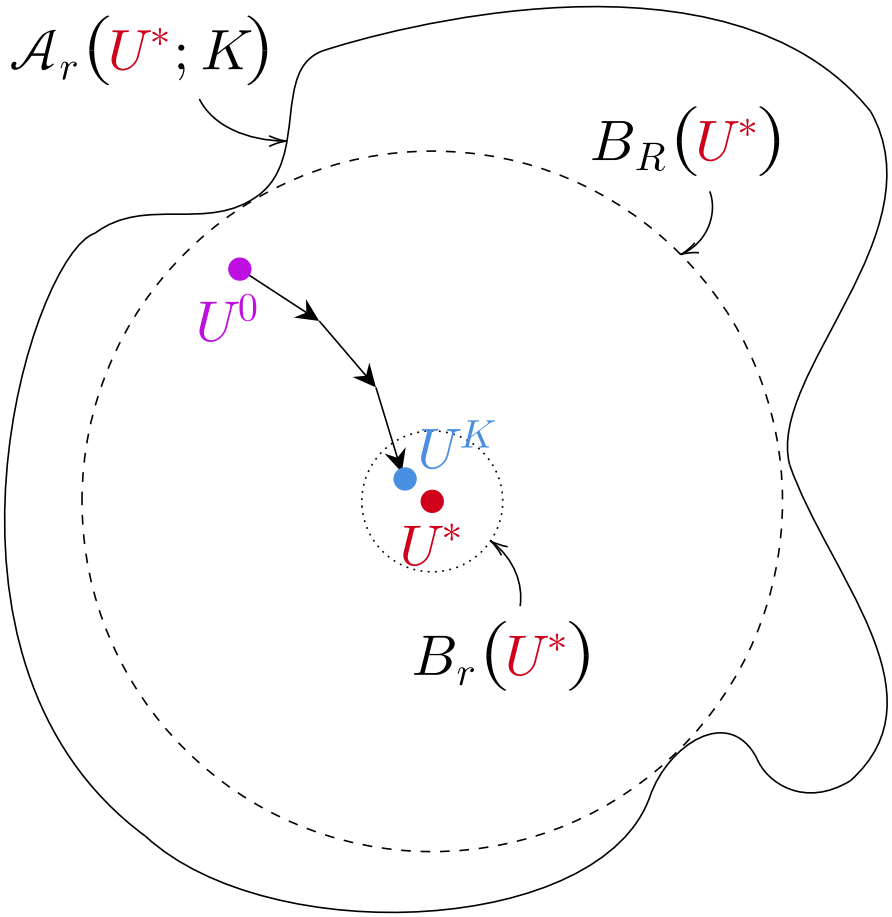

Figure 3.3: Outer and inner balls. The inner ball $B_r(U^*)$ holds all points within tolerance r of the true solution U^*. The outer ball $B_R(U^*)$ is the largest ball fully contained in the K-iteration basin $\mathcal{A}_r(U^*; K)$. Starting from any point U^0 inside $B_R(U^*)$, the time-parallel algorithm will reach $B_r(U^*)$ within K iterations. Even though the basin $\mathcal{A}_r(U^*; K)$ may have a complex shape, every point inside $B_R(U^*)$ will converge.

If we start anywhere in the outer ball, so that $\|U^0 - U^*\| \leq R$, then

$$\|U^K - U^*\| \leq \beta^{S_K(q)} R^{q^K}. \tag{3.10}$$

To make sure that $\|U^K - U^*\| \leq r$, we need

$$\beta^{S_K(q)} R^{q^K} \leq r, \tag{3.11}$$

which gives us

$$R \leq \left(\frac{r}{\beta^{S_K(q)}} \right)^{1/q^K}. \tag{3.12}$$

For linear convergence ($q = 1$), we have $S_K(1) = K$, so

$$R \leq \frac{r}{\beta^K}. \tag{3.13}$$

This tells us that we can be $R/r = 1/\beta^K$ times further away from the solution and still converge within K iterations. For example, if $\beta = 0.5$, $K = 5$, and $r = 10^{-6}$, then $R \approx 3.2 \times 10^{-5}$, which means that we can start with an error 32 times larger than our tolerance and still converge in just 5 iterations.

3.2.3 Practical implications

These bounds give us a clear strategy. The user sets the tolerance r and chooses how many iterations K they are willing to spend. In practice, we need an estimate for the convergence factor β, either through analysis – as in Section 3.3 – or through numerical experiments. Once we have this estimate, we can compute the outer radius R from (3.13).

This, in turn, tells us how accurate our coarse solver needs to be. Since we generate the initial guess U^0 using the coarse solver, we can adjust its step size to ensure $\|U^0 - U^*\| \leq R$, by keeping the number of coarse steps fixed and shrinking the step size until U^0 lands inside the outer ball. Of course, making the coarse solver more accurate costs more, so we want to pick the largest step size that still lands us inside $B_R(U^*)$.

Anyway, this framework is especially useful for chaotic systems. Given the complex geometry of the basin of attraction, it is hard to know if a given initial guess will converge. But if we can bring U^0 inside the outer ball $B_R(U^*)$, then convergence is guaranteed, no matter how tangled the basin might be.

This outer-inner ball framework will play a key role in Chapter 4, where we use it to make sure that MoWi stays 'on track' as it moves forwards in time. Each window must start within its own outer ball to guarantee convergence, and we will see how to maintain this property as we shift from one window to the next.

3.3 Application to Parareal

In this section, we show how to apply Theorem 3.1.3 in practice by using Parareal as an example. First, we need to check that Parareal meets the necessary requirements. We have already discussed in Section 3.1 that U^* is a unique fixed point, so we just need to verify the second assumption: that $\|I + \mathcal{D}v(U)\| \leq \beta$ for some $\beta \in (0, 1)$. For any $x \geq 0$ and integer $N \geq 1$, we define the partial sum function $s(x)$ as:

$$s(x) = \sum_{i=0}^{N-1} x^i = \begin{cases} N & \text{if } x = 1, \\ \frac{x^N - 1}{x - 1} & \text{otherwise.} \end{cases} \tag{3.14}$$

Theorem 3.3.1. Let $\|\cdot\|$ be a p-norm, and $\mathcal{X} \subset \mathbb{R}^{d(N+1)}$. If both F and G are continuously differentiable everywhere, then

$$\|I + \mathcal{D}v(U)\| \leq \beta, \tag{3.15}$$

where

$$\beta = \left(\sup_{\substack{U \in \mathcal{X} \\ 1 \leq n \leq N}} s\big(\|\mathcal{D}G(U_n)\|\big) \right) \cdot \left(\sup_{\substack{U \in \mathcal{X} \\ 0 \leq n \leq N-1}} \|\mathcal{D}F(U_n) - \mathcal{D}G(U_n)\| \right), \tag{3.16}$$

where $\mathcal{D}F$ and $\mathcal{D}G$ are the Jacobians of F and G, respectively.

Proof. From (3.2), we have that $[v(U^{k-1})]_n = U_n^k - U_n^{k-1}$. To find the Jacobian of v, we treat U^{k-1} as the independent variable. Recall that the Parareal iteration is

$$\begin{cases} U_0^k = u_0, \\ U_n^k = G(U_{n-1}^k) + F(U_{n-1}^{k-1}) - G(U_{n-1}^{k-1}), \quad n = 1, \ldots, N. \end{cases} \tag{3.17}$$

We can write the Jacobian $\mathcal{D}v(U^{k-1})$ acting on a direction vector V as

$$\mathcal{D}v(U^{k-1})V = \frac{\mathrm{d}}{\mathrm{d}\alpha} v(U^{k-1} + \alpha V)\bigg|_{\alpha=0}. \tag{3.18}$$

Since $U_0^k = U_0$ is constant for all k, the term U_n^k depends on α through the previous iterates. Taking derivatives gives

$$[\mathcal{D}v(U^{k-1})V]_n = \frac{\mathrm{d}U_n^k}{\mathrm{d}\alpha}\bigg|_{\alpha=0} - V_n \tag{3.19}$$

$$= \mathcal{D}G(U_{n-1}^k) \frac{\mathrm{d}U_{n-1}^k}{\mathrm{d}\alpha}\bigg|_{\alpha=0} + \big(\mathcal{D}F(U_{n-1}) - \mathcal{D}G(U_{n-1})\big) V_{n-1} - V_n. \tag{3.20}$$

For convenience, we set $A_a^b = \prod_{j=a}^{b} \mathcal{D}G(U_j^k)$ and $D_n = \mathcal{D}F(U_n) - \mathcal{D}G(U_n)$, so that

$$[\mathcal{D}v(U^{k-1})V]_n = A_{n-1}^{n-1} \left.\frac{\mathrm{d}U_{n-1}^k}{\mathrm{d}\alpha}\right|_{\alpha=0} + D_{n-1}V_{n-1} - V_n \tag{3.21}$$

$$= A_{n-2}^{n-1} \left.\frac{\mathrm{d}U_{n-2}^k}{\mathrm{d}\alpha}\right|_{\alpha=0} + A_{n-1}^{n-1}D_{n-2}V_{n-2} + D_{n-1}V_{n-1} - V_n. \tag{3.22}$$

Applying this recursively, we get

$$[\mathcal{D}v(U^{k-1})V]_n = \sum_{i=0}^{n-1} A_{i+1}^{n-1} D_i V_i - V_n. \tag{3.23}$$

In matrix form, this becomes

$$\mathcal{D}v(U^{k-1})V = \begin{bmatrix} -I_d & & & & \\ D_0 & \ddots & & & \\ A_1^1 D_0 & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \\ A_1^{N-1}D_0 & \cdots & A_{N-1}^{N-1}D_{N-2} & D_{N-1} & -I_d \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{N-1} \\ V_N \end{bmatrix}, \tag{3.24}$$

such that

$$I + \mathcal{D}v(U^{k-1}) = \begin{bmatrix} 0 & & & & \\ D_0 & \ddots & & & \\ A_1^1 D_0 & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \\ A_1^{N-1}D_0 & \cdots & A_{N-1}^{N-1}D_{N-2} & D_{N-1} & 0 \end{bmatrix}. \tag{3.25}$$

We can factor this as $I + \mathcal{D}v(U^{k-1}) = C \cdot D$, where C contains the products of coarse Jacobians and D is block-diagonal with entries D_n:

$$I + \mathcal{D}v(U^{k-1}) = \underbrace{\begin{bmatrix} 0 & & & & \\ I_d & \ddots & & & \\ A_1^1 & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \\ A_1^{N-1} & \cdots & A_{N-1}^{N-1} & I_d & 0 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} D_0 & & & & \\ & D_1 & & & \\ & & \ddots & & \\ & & & D_{N-1} & \\ & & & & 0 \end{bmatrix}}_{D}. \tag{3.26}$$

This gives us

$$\|D\| \leq \max_{0\leq n\leq N-1} \|D_n\| = \max_{0\leq n\leq N-1} \|\mathcal{D}F(U_n) - \mathcal{D}G(U_n)\|, \tag{3.27}$$

and

$$C = \underbrace{\begin{bmatrix} 0 & & & \\ I_d & \ddots & & \\ & \ddots & \ddots & \\ & & I_d & 0 \end{bmatrix}}_{Y} \sum_{i=0}^{N} B^i, \quad B = \begin{bmatrix} 0 & & & \\ A_1^1 & \ddots & & \\ & \ddots & \ddots & \\ & & A_N^N & 0 \end{bmatrix}. \tag{3.28}$$

B is a nilpotent matrix, with $B^{N+1} = 0$. The pre-multiplying matrix Y acts as a shift operator that annihilates the final term of the sum, since $Y \cdot B^N = 0$. This is because B^N has only one non-zero block at the bottom of its first column, which is mapped to zero by the final zero-row of Y. The expression for C therefore simplifies before we take the norm:

$$C = Y \sum_{i=0}^{N-1} B^i. \tag{3.29}$$

Using the function $s(x)$ defined in (3.14), we can bound the norm of the geometric series. Since $\|Y\| = 1$, we have

$$\|C\| \leq \sum_{i=0}^{N-1} \|B\|^i = s(\|B\|). \tag{3.30}$$

Since $s(x)$ is monotonically increasing for $x \geq 0$, and $\|B\| = \max_{1\leq n\leq N} \|A_n^n\|$, we can write

$$\|C\| \leq s\left(\max_{1\leq n\leq N} \|A_n^n\|\right) = \max_{1\leq n\leq N} s\big(\|A_n^n\|\big). \tag{3.31}$$

Substituting $\|A_n^n\| = \|\mathcal{D}G(U_n)\|$ and taking the supremum over all $U \in \mathcal{X}$ completes the proof. □

As a sanity check, we compare Theorem 3.3.1 with Theorem 2.7.1 from Section 2.7.3. For Dahlquist's equation with F the exact integrator and G a one-step method in its region of absolute stability with time-step size equal to the time-chunk size, letting $N \to \infty$ gives us

$$\beta = \frac{|e^z - \mathcal{R}_G(z)|}{1 - |\mathcal{R}_G(z)|}, \quad z = \mu\Delta T, \tag{3.32}$$

which matches Gander and Vandewalle's result.

3.3.1 Getting an explicit bound

Now we need to make this bound practical, that is, we need to find what drives β below 1. We focus on Runge–Kutta methods, since they are standard in ODE integration, their stability functions are well-understood, and we can compute all the necessary Jacobians explicitly.

Here is our setup. We assume that F and G are the compositions of L_F and L_G Runge–Kutta steps of orders p_F and p_G, respectively, with stability functions $\mathcal{R}_F$ and $\mathcal{R}_G$, step sizes h_F and h_G, tableaux (A_F, b_F, c_F) and (A_G, b_G, c_G), and stage numbers s_F and s_G. We make the fine and coarse grids overlap such that $\Delta T = h_F L_F = h_G L_G$, $h_G = \xi h_F$, and $L_F = \xi L_G$, where $\xi \geq 1$ is the coarse-to-fine step ratio. Also, we set $h = h_F$ and $L = L_G$ to simplify our notation.

Looking at the Runge–Kutta equations, we find that

$$\mathcal{D}F(U_n) = \prod_{i=1}^{\xi L} \mathcal{R}_F\big(hJ(T_n + (i-1)h)\big), \tag{3.33}$$

$$\mathcal{D}G(U_n) = \prod_{j=1}^{L} \mathcal{R}_G\big(\xi hJ(T_n + (j-1)\xi h)\big), \tag{3.34}$$

where $J(t) = \mathcal{D}f(u(t))$ is the Jacobian of the right-hand side function f of the IVP. For brevity, we define the arguments of $\mathcal{R}_F$ and $\mathcal{R}_G$, respectively, as

$$X_i = hJ(T_n + (i-1)h), \tag{3.35}$$

$$Y_j = \xi hJ(T_n + (j-1)\xi h). \tag{3.36}$$

We will make use of the Taylor expansion

$$X_i = Y_j/\xi + h^2\big(\xi(j-1) - (i-1)\big)J'(\tau_{ij}), \tag{3.37}$$

for some $\tau_{ij} \in [T_n + (i-1)h, T_n + (j-1)\xi h]$.

For the first term in (3.16), the bound from Theorem 2.6.1 in Section 2.6 gives us $\|\mathcal{D}G(U_n)\| \leq \Lambda_G$, where Λ_G is the Lipschitz constant of G. Since $s(x)$ is monotonically increasing for $x \geq 0$, we have

$$\sup_{\substack{U \in \mathcal{X} \\ 1 \leq n \leq N}} s\big(\|\mathcal{D}G(U_n)\|\big) \leq s(\Lambda_G). \tag{3.38}$$

For the second term, we could try splitting the difference into products. Let $S_j = \mathcal{R}_F(X_{\xi j}) \cdots \mathcal{R}_F(X_{\xi(j-1)+1})$. Then a simple add-and-subtract argument, together

with the submultiplicativity of $\|\cdot\|$, gives

$$\|\mathcal{D}F(U_n) - \mathcal{D}G(U_n)\| = \left\| \prod_{i=1}^{\xi L} \mathcal{R}_F(X_i) - \prod_{j=1}^{L} \mathcal{R}_G(Y_j) \right\| = \left\| \prod_{j=1}^{L} S_j - \prod_{j=1}^{L} \mathcal{R}_G(Y_j) \right\| \tag{3.39}$$

$$\leq \sum_{k=1}^{L} \left\| \prod_{j=k+1}^{L} S_j \right\| \|S_k - \mathcal{R}_G(Y_k)\| \left\| \prod_{j=1}^{k-1} \mathcal{R}_G(Y_j) \right\|. \tag{3.40}$$

However, this bound is too loose in practice. The problem is that it applies the triangle inequality at too early a stage, which makes it blind to crucial error cancellations. Instead, we study to what leading order $\|\mathcal{D}F(U_n) - \mathcal{D}G(U_n)\|$ decreases as $h \to 0$.

Theorem 3.3.2. Let $\|\cdot\|$ be a p-norm. Assume that there exist non-negative scalars μ and ν such that $\|J(t)\| \leq \mu$ and $\|J'(t)\| \leq \nu$ for all $t \in [0, T]$. If

$$h < \frac{1}{\mu \max\{\|A_F\|, \xi\|A_G\|\}}, \tag{3.41}$$

then

$$\sup_{\substack{U \in \mathcal{X} \\ 0 \leq n \leq N-1}} \|\mathcal{D}F(U_n) - \mathcal{D}G(U_n)\| \leq L\xi(h\mu)^2(C_1 + C_2 + C_3) + \mathcal{O}(h^3), \tag{3.42}$$

where

$$C_1 = \max\{\|A_F\|, \xi\|A_G\|\} \left(s_F \|b_F\| \|c_F\| + s_G \|b_G\| \|c_G\|\right), \tag{3.43}$$

$$C_2 = \frac{\xi - 1}{2} \left(1 + \frac{\nu}{\mu^2}\right), \tag{3.44}$$

$$C_3 = |1 + b_F^\top c_F|(\xi - 1) + |b_F^\top c_F - b_G^\top c_G|\xi(L + 1). \tag{3.45}$$

Proof. Let $I_F = I_{s_F d}$ and $I_G = I_{s_G d}$ be identity matrices, and $\mathbf{1}_F = \mathbf{1}_{s_F}$ and $\mathbf{1}_G = \mathbf{1}_{s_G}$ be column vectors of all ones. From the Runge–Kutta stability formula (2.41),

$$R_F(X_i) = I_d + (b_F^\top \otimes I_d) \left(I_F - A_F \otimes X_i\right)^{-1} (\mathbf{1}_F \otimes X_i) =: I_d + P_i, \tag{3.46}$$

$$R_G(Y_j) = I_d + (b_G^\top \otimes I_d) \left(I_G - A_G \otimes Y_j\right)^{-1} (\mathbf{1}_G \otimes Y_j) =: I_d + Q_j. \tag{3.47}$$

Let $D_n = \mathcal{D}F(U_n) - \mathcal{D}G(U_n)$. Using the multi-index notation $\mathbf{i} = (i_1, \ldots, i_{\xi L})$ and $\mathbf{j} = (j_1, \ldots, j_L)$, we expand

$$D_n = \sum_{|\mathbf{i}|=1}^{\xi L} P_{\xi L}^{i_{\xi L}} \cdots P_1^{i_1} - \sum_{|\mathbf{j}|=1}^{L} Q_L^{j_L} \cdots Q_1^{i_1}. \tag{3.48}$$

This yields

$$D_n = \underbrace{\sum_{i_1=1}^{\xi L} P_{i_1} - \sum_{j_1=1}^{L} Q_{j_1}}_{\Delta_1} + \underbrace{\sum_{i_1=1}^{\xi L} P_{i_1} \sum_{i_2=1}^{i_1} P_{i_2} - \sum_{j_1=1}^{L} Q_{j_1} \sum_{j_2=1}^{j_1} Q_{j_2}}_{\Delta_2} + \mathcal{O}(h^3), \tag{3.49}$$

where Δ_1 collects the first-order terms and Δ_2 the second-order terms.

We begin by looking at Δ_1. Assume that either $A_F \neq 0$ or $A_G \neq 0$. For step size

$$h < \frac{1}{\mu \max\{\|A_F\|, \xi\|A_G\|\}}, \tag{3.50}$$

we can use the Neumann series to expand

$$(I_F - A_F \otimes X_i)^{-1} = \sum_{k=0}^{\infty} A_F^k \otimes X_i^k, \tag{3.51}$$

$$(I_G - A_G \otimes Y_j)^{-1} = \sum_{k=0}^{\infty} A_G^k \otimes Y_j^k. \tag{3.52}$$

Since useful Runge–Kutta methods must be consistent and at least first-order, then $A_F \mathbf{1}_F = c_F$, $A_G \mathbf{1}_G = c_G$, and $b_F^\top \mathbf{1}_F = b_G^\top \mathbf{1}_G = 1$, we obtain

$$P_i = \sum_{k=0}^{\infty} \left(b_F^\top A_F^k \mathbf{1}_F\right) X_i^{k+1} = X_i + \sum_{k=1}^{\infty} \left(b_F^\top A_F^k c_F\right) X_i^{k+1}, \tag{3.53}$$

$$Q_j = \sum_{k=0}^{\infty} \left(b_G^\top A_G^k \mathbf{1}_G\right) Y_j^{k+1} = Y_j + \sum_{k=1}^{\infty} \left(b_G^\top A_G^k c_G\right) Y_j^{k+1}. \tag{3.54}$$

Dropping the subscripts in i_1 and j_1 to lighten our notation, the first-order terms decompose as

$$\sum_{i=1}^{\xi L} P_i - \sum_{j=1}^{L} Q_j = \underbrace{\sum_{i=1}^{\xi L} X_i - \sum_{j=1}^{L} Y_j}_{k=0 \text{ term}} + \underbrace{\sum_{k=1}^{\infty} \left(b_F^\top A_F^{k-1} \mathbf{1}_F \sum_{i=1}^{\xi L} X_i^{k+1} - b_G^\top A_G^{k-1} \mathbf{1}_G \sum_{j=1}^{L} Y_j^{k+1} \right)}_{(3.55.2)}. \tag{3.55}$$

Using the Taylor expansion (3.37), the $k = 0$ term (that is, the part linear in the Jacobians) satisfies

$$\left\| \sum_{i=1}^{\xi L} X_i - \sum_{j=1}^{L} Y_j \right\| \tag{3.56}$$

$$\leq h^2 \sum_{j=1}^{L} \sum_{i=\xi(j-1)+1}^{\xi j} [(i-1) - \xi(j-1)] \|J'(t_{ij})\| \tag{3.57}$$

$$\leq Lh^2\nu\frac{\xi(\xi-1)}{2}. \tag{3.58}$$

For the $1 \leq k < p$ terms of (3.55.2), we use the order conditions $b^\top A^{k-1}\mathbf{1} = 1/k!$, which are derived directly from matching the Taylor series of the stability function, $\mathcal{R}(z)$, to the Taylor series of the true exponential, e^z.

Let $\tilde{p} = \min(p_F, p_G)$. Then

$$\sum_{k=1}^{\tilde{p}-1}\frac{1}{(k+1)!}\left(\sum_{i=1}^{\xi L}X_i^{k+1} - \sum_{j=1}^{L}Y_j^{k+1}\right) \tag{3.59}$$

$$= \sum_{k=1}^{\tilde{p}-1}\frac{1}{(k+1)!}\sum_{j=1}^{L}\left[\left(\sum_{i=\xi(j-1)+1}^{\xi j}X_i^{k+1}\right) - Y_j^{k+1}\right] \tag{3.60}$$

$$= \sum_{k=1}^{\tilde{p}-1}\frac{1}{(k+1)!}\left(\frac{1}{\xi^k}-1\right)\sum_{j=1}^{L}Y_j^{k+1} + \mathcal{O}(h^3). \tag{3.61}$$

Taking norms,

$$\left\|\sum_{k=1}^{\tilde{p}-1}\frac{1}{(k+1)!}\left(\frac{1}{\xi^k}-1\right)\sum_{j=1}^{L}Y_j^{k+1}\right\| \tag{3.62}$$

$$\leq L\sum_{k=1}^{\tilde{p}-1}\frac{(\xi h\mu)^{k+1}}{(k+1)!}\left(1-\frac{1}{\xi^k}\right) \tag{3.63}$$

$$= L\left(\frac{e^{\xi h\mu}\Gamma(\tilde{p}+1,\xi h\mu) - \xi e^{h\mu}\Gamma(\tilde{p}+1,h\mu)}{\Gamma(\tilde{p}+1)} + \xi - 1\right), \tag{3.64}$$

where $\Gamma(x)$ is the gamma function and $\Gamma(n,x)$ is the incomplete gamma function.

We analyse the expression by expanding the numerator around $h = 0$. Let $a = \tilde{p}+1$ and $x = h\mu$. We use the known series expansions for the exponential function, that is

$$e^z = 1 + z + \frac{z^2}{2} + \mathcal{O}(z^3), \tag{3.65}$$

and the lower incomplete gamma function $\gamma(a,z) = \int_0^z t^{a-1}e^{-t}\mathrm{d}t$, that is

$$\Gamma(a,z) = \Gamma(a) - \gamma(a,z) = \Gamma(a) - \left(\frac{z^a}{a} - \frac{z^{a+1}}{a+1} + \mathcal{O}(z^{a+2})\right). \tag{3.66}$$

Combining these, we can expand the term $e^{zx}\Gamma(a,zx)$ as

$$e^{zx}\Gamma(a,zx) = \left(1 + zx + \frac{(zx)^2}{2} + \mathcal{O}(x^3)\right)\left(\Gamma(a) - \frac{(zx)^a}{a} + \mathcal{O}(x^{a+1})\right) \tag{3.67}$$

$$= \Gamma(a) + zx\Gamma(a) + \frac{(zx)^2}{2}\Gamma(a) - \frac{(zx)^a}{a} + \mathcal{O}(x^3) + \mathcal{O}(x^{a+1}). \tag{3.68}$$

Since $\tilde{p} \geq 1$, we have $a \geq 2$. Therefore, the term $(zx)^a$ is at least of order $\mathcal{O}(x^2)$.

We now substitute $z = \xi, 1$ into the expansion above and compute the difference in the numerator of our main expression, that is $e^{\xi x}\Gamma(a, \xi x) - \xi e^x \Gamma(a, x)$. Collecting terms by powers of $x = h\mu$, we have

$$\Gamma(a) - \xi\Gamma(a) = (1 - \xi)\Gamma(a), \tag{3.69}$$

$$\xi x\Gamma(a) - \xi(x\Gamma(a)) = 0, \tag{3.70}$$

$$\frac{(\xi x)^2}{2}\Gamma(a) - \xi\left(\frac{x^2}{2}\Gamma(a)\right) = \xi(1 - \xi)\frac{x^2\Gamma(a)}{2}. \tag{3.71}$$

All higher-order terms are at least $\mathcal{O}(x^3)$. Summing these gives

$$e^{\xi x}\Gamma(a, \xi x) - \xi e^x \Gamma(a, x) = (1 - \xi)\Gamma(a) + \xi(1 - \xi)\frac{x^2\Gamma(a)}{2} + \mathcal{O}(x^3). \tag{3.72}$$

Substituting this back into our original expression,

$$(3.64) = L\left(\frac{(1 - \xi)\Gamma(a) + \frac{(h\mu)^2\Gamma(a)}{2}\xi(\xi - 1) + \mathcal{O}(h^3)}{\Gamma(a)} + \xi - 1\right) \tag{3.73}$$

$$= L\left((1 - \xi) + \frac{(h\mu)^2}{2}\xi(\xi - 1) + (\xi - 1) + \mathcal{O}(h^3)\right) \tag{3.74}$$

$$= L(h\mu)^2\frac{\xi(\xi - 1)}{2} + \mathcal{O}(h^3). \tag{3.75}$$

That cancellation of the constant terms $(1 - \xi)$ and $(\xi - 1)$ is not an accident: it is a direct and fundamental consequence of the fact that both the fine and coarse Runge–Kutta solvers are at least first-order accurate, that is, consistent.

The remaining terms for $k \geq \tilde{p}$ can be split into $k = \tilde{p}$ and $k > \tilde{p}$ terms. The latter are $\mathcal{O}(h^3)$, so we need only to look at the former. The triangle inequality gives

$$\left\| b_F^\top A_F^{\tilde{p}} \mathbf{1}_F \sum_{i=1}^{\xi L} X_i^{\tilde{p}+1} - b_G^\top A_G^{\tilde{p}} \mathbf{1}_G \sum_{j=1}^{L} Y_j^{\tilde{p}+1} \right\|$$

$$\leq |b_F^\top A_F^{\tilde{p}} c_F| \sum_{i=1}^{\xi L} \|X_i\|^{\tilde{p}+1} + |b_G^\top A_G^{\tilde{p}} c_G| \sum_{j=1}^{L} \|Y_j\|^{\tilde{p}+1}. \tag{3.76}$$

Using Hölder's inequality,

$$|b_F^\top A_F^{\tilde{p}} c_F| \leq s_F \|b_F\| \|c_F\| \|A_F\|^{\tilde{p}}, \quad |b_G^\top A_G^{\tilde{p}} c_G| \leq s_G \|b_G\| \|c_G\| \|A_G\|^{\tilde{p}}. \tag{3.77}$$

Since $\|X_i\| \leq h\mu$ and $\|Y_j\| \leq \xi h\mu$,

$$|b_F^\top A_F^{\tilde{p}} c_F| \sum_{i=1}^{\xi L} \|X_i\|^{\tilde{p}+1} + |b_G^\top A_G^{\tilde{p}} c_G| \sum_{j=1}^{L} \|Y_j\|^{\tilde{p}+1}$$
$$\leq L\xi(h\mu)^2 \left[s_F \|b_F\| \|c_F\| \|A_F\|^{\tilde{p}} (h\mu)^{\tilde{p}-1} + s_G \|b_G\| \|c_G\| (\xi \|A_G\|)^{\tilde{p}} (h\mu)^{\tilde{p}-1} \right]. \quad (3.78)$$

Since $h < 1/(\mu \max\{\|A_F\|, \xi\|A_G\|\})$ and $\tilde{p} \geq 1$,

$$s_F \|b_F\| \|c_F\| \|A_F\|^{\tilde{p}} (h\mu)^{\tilde{p}-1} + s_G \|b_G\| \|c_G\| (\xi \|A_G\|)^{\tilde{p}} (h\mu)^{\tilde{p}-1}$$
$$< \max\{\|A_F\|, \xi\|A_G\|\} \left(s_F \|b_F\| \|c_F\| + s_G \|b_G\| \|c_G\| \right), \quad (3.79)$$

so that

$$\left\| b_F^\top A_F^{\tilde{p}} \mathbf{1}_F \sum_{i=1}^{\xi L} X_i^{\tilde{p}+1} - b_G^\top A_G^{\tilde{p}} \mathbf{1}_G \sum_{j=1}^{L} Y_j^{\tilde{p}+1} \right\|$$
$$< L\xi(h\mu)^2 \max\{\|A_F\|, \xi\|A_G\|\} \left(s_F \|b_F\| \|c_F\| + s_G \|b_G\| \|c_G\| \right). \quad (3.80)$$

We now look at Δ_2 in (3.49), that is the second-order terms. We use the Neumann expansions to obtain

$$P_{i_1} P_{i_2} = \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} \left(b_F^\top A_F^k \mathbf{1}_F \right) \left(b_F^\top A_F^l \mathbf{1}_F \right) X_{i_1}^{k+1} X_{i_2}^{l+1} \quad (3.81)$$
$$= X_{i_1} X_{i_2} + b_F^\top c_F \left(X_{i_1}^2 + X_{i_2}^2 \right) + \mathcal{O}(h^3), \quad (3.82)$$

$$Q_{j_1} Q_{j_2} = \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} \left(b_G^\top A_G^k \mathbf{1}_G \right) \left(b_G^\top A_G^l \mathbf{1}_G \right) Y_{j_1}^{k+1} Y_{j_2}^{l+1} \quad (3.83)$$
$$= Y_{j_1} Y_{j_2} + b_G^\top c_G \left(Y_{j_1}^2 + Y_{j_2}^2 \right) + \mathcal{O}(h^3). \quad (3.84)$$

Hence,

$$\Delta_2 = \sum_{i_1=1}^{\xi L} X_{i_1} \sum_{i_2=1}^{i_1} X_{i_2} - \sum_{j_1=1}^{L} Y_{j_1} \sum_{j_2=1}^{j_1} Y_{j_2} \quad (3.85)$$
$$+ b_F^\top c_F \left(\sum_{i_1=1}^{\xi L} i_1 X_{i_1}^2 + \sum_{i_2=1}^{\xi L} i_2 X_{i_2}^2 \right) - b_G^\top c_G \left(\sum_{j_1=1}^{L} j_1 Y_{j_1}^2 + \sum_{j_2=1}^{L} j_2 Y_{j_2}^2 \right) \quad (3.86)$$
$$+ \mathcal{O}(h^3). \quad (3.87)$$

On one hand,

$$(3.85) = \sum_{i=1}^{L} \left\{ \frac{Y_i}{\xi} \left[\left(\sum_{n=0}^{\xi-1} \frac{\xi - n}{\xi} \right) Y_i + \xi \left(\sum_{j=1}^{i-1} Y_j \right) \right] - Y_i \left(\sum_{j=1}^{i} Y_j \right) \right\} \quad (3.88)$$

$$= \sum_{i=1}^{L} \left[\left(\sum_{n=0}^{\xi-1} \frac{\xi - n}{\xi^2} \right) - 1 \right] Y_i^2 + \mathcal{O}(h^4) \tag{3.89}$$

$$= \frac{1-\xi}{2\xi} \sum_{i=1}^{L} Y_i^2 + \mathcal{O}(h^4). \tag{3.90}$$

On the other hand,

$$\frac{(3.86)}{2} = b_F^\top c_F \sum_{i=1}^{\xi L} i X_i^2 - b_G^\top c_G \sum_{j=1}^{L} j Y_j^2 \tag{3.91}$$

$$= b_F^\top c_F \sum_{j=1}^{L} \sum_{i=\xi(j-1)+1}^{\xi j} i X_i^2 - b_G^\top c_G \sum_{j=1}^{L} j Y_j^2 \tag{3.92}$$

$$= \sum_{j=1}^{L} Y_j^2 \left[b_F^\top c_F \left(\sum_{i=\xi(j-1)+1}^{\xi j} \frac{i}{\xi^2} \right) - b_G^\top c_G j \right] + \mathcal{O}(h^4) \tag{3.93}$$

$$= \sum_{j=1}^{L} Y_j^2 \left[b_F^\top c_F \left(j + \frac{1-\xi}{2\xi} \right) - b_G^\top c_G j \right] + \mathcal{O}(h^4) \tag{3.94}$$

$$= (b_F^\top c_F - b_G^\top c_G) \sum_{j=1}^{L} j Y_j^2 + b_F^\top c_F \frac{1-\xi}{2\xi} \sum_{i=1}^{L} Y_i^2 + \mathcal{O}(h^4). \tag{3.95}$$

Putting (3.90) and (3.95) back together, we obtain

$$\Delta_2 = (1 + b_F^\top c_F) \frac{1-\xi}{\xi} \sum_{j=1}^{L} Y_j^2 + 2 (b_F^\top c_F - b_G^\top c_G) \sum_{j=1}^{L} j Y_j^2 + \mathcal{O}(h^4), \tag{3.96}$$

such that, taking norms,

$$\|\Delta_2\| \leq |1 + b_F^\top c_F| \frac{\xi - 1}{\xi} \sum_{j=1}^{L} \|Y_j\|^2 + 2|b_F^\top c_F - b_G^\top c_G| \sum_{j=1}^{L} j \|Y_j\|^2 + \mathcal{O}(h^3) \tag{3.97}$$

$$\leq |1 + b_F^\top c_F| \frac{\xi - 1}{\xi} L(\xi h \mu)^2 + |b_F^\top c_F - b_G^\top c_G| L(L+1)(\xi h \mu)^2 + \mathcal{O}(h^3) \tag{3.98}$$

$$= L\xi(h\mu)^2 [|1 + b_F^\top c_F|(\xi - 1) + |b_F^\top c_F - b_G^\top c_G| \xi (L+1)] + \mathcal{O}(h^3). \tag{3.99}$$

Finally, we join (3.58), (3.75), (3.80) and (3.99) to get the required result. □

Remark 3.3.3. We are yet to cover the case when $A_F = A_G = 0$. This corresponds to both F and G being based on the explicit Euler method. In this case, $P_i = X_i$ and $Q_j = Y_j$. Following the same steps, we still arrive at (3.42), but now with $\|A_F\| = \|A_G\| = 0$ and no constraints on h.

Remark 3.3.4. The convergence rate of the Parareal algorithm depends directly on how well the coarse propagator G mimics the fine propagator F. The main difficulty in this mimicry lies in reproducing how F captures the curvature of the solution's trajectory. This effect is quantified by the constant C_3 in the error bound, which has two parts.

The first term, $|1 + b_F^\top c_F|(\xi - 1)$, reflects the structure of the fine method F itself. The second, $|b_F^\top c_F - b_G^\top c_G|, \xi(L + 1)$, comes from a mismatch between how F and G represent curvature at the most basic level. Crucially, this mismatch term vanishes if the methods are chosen so that $b_F^\top c_F = b_G^\top c_G$. This holds regardless of whether the methods capture curvature well or badly.

For example, if both F and G are at least second-order accurate, then by construction $b^\top c = 1/2$, and the condition is satisfied. Similarly, if F and G are two different first-order methods built to share the same $b^\top c$ value (say $1/4$), the condition again holds. In all such cases, C_3 simplifies to

$$C_3 = |1 + b_F^\top c_F|(\xi - 1), \tag{3.100}$$

removing the dependence on the number of coarse steps L and thus tightening the convergence bound.

Combining Theorems 3.3.1 and 3.3.2, we get the explicit bound

$$\beta \leq \frac{\Lambda_G^N - 1}{\Lambda_G - 1} L\xi(h\mu)^2(C_1 + C_2 + C_3) + \mathcal{O}(h^3). \tag{3.101}$$

Because Λ_G will get quite big as h increases, (3.38) remains the biggest contributor, and we expect our bound on β to be conservative. This is due to the fact that it essentially comes from linearizing the problem. Still, it applies to general nonlinear problems, and the $\mathcal{O}(h^2)$ scaling remains the same.

Also, all parameters in (3.42) are known upfront, before the simulation starts, with the notable exception of μ and ν.

To estimate them, we need the Jacobian $J(t) = \mathcal{D}f(u(t))$ along the solution trajectory. For the time derivative, the chain rule gives

$$J'(t) = \frac{\mathrm{d}}{\mathrm{d}t} J(t) = \sum_j f_j(t) \frac{\partial}{\partial u_j} \mathcal{D}f(u(t)), \tag{3.102}$$

which requires second derivatives of f.

In practice, we can estimate μ and ν during a coarse solve by tracking $\|J(t)\|$ and $\|J'(t)\|$ at the coarse or fine time points. Since the coarse solver evaluates f and (for

implicit methods) $\mathcal{D}f$ anyway, the additional cost is negligible. For explicit methods that do not compute $\mathcal{D}f$, we can either use finite differences to approximate J at coarse points or use problem-specific bounds.

Nevertheless, the main takeaway is clear: β scales as $\mathcal{O}(h^2)$ for small h. This quadratic dependence is crucial – halving h cuts β by about four. If Parareal struggles to converge, we know exactly what to do: make h smaller. So if Parareal doesn't converge, we make h smaller until it does. This means that for fixed ξ and L, there's always an h small enough to make Parareal converge linearly. This means that for a fixed number of fine and coarse steps – i.e. for ξ and L fixed – there exists a step size h small enough such that Parareal converges linearly to U^* and for its iterates U^k to remain in a ball around it.

But h cannot be too small. For plasma systems, making the time step too small might change the problem. For example, in plasma turbulence, too small a step size might resolve phenomena that should be modelled, not simulated.

Hence, there is a trade-off: h must be small enough for convergence ($\beta < 1$) but large enough to solve the right problem. Once h is set, then we can adjust the parameters of the coarse solver to further improve convergence.

It is worthwhile to notice that *this framework does not depend on our specific bound for* β. Any tighter bound can be used instead. What matters is the overall procedure: find the parameters on which the bound on β depends and tune them to $\beta < 1$. This will be enough to prove linear convergence and to give us finite-time guarantees that we can use in practice. Anyhow, for the lack of a better bound on β, we will use this specific bound in Chapter 4 to guarantee that each window in MoWi converges within a fixed computational budget.

3.3.2 Experiments

We conclude this section by showing how our estimate of the convergence rate β changes with the fine-solver step size h. We run our experiments for the logistic, Lorenz, and Lorenz-96 equations, tracking convergence by measuring the Euclidean (2-norm) difference between the current iterate and the fine solution, i.e. $\|U^k - U^*\|$, as $k \to N$. Each simulation uses our custom-made packages `NSDERungeKutta.jl` and `NSDETimeParallel.jl` – see Section 2.8 for more details.

We begin with the logistic equation, for which we can derive every relevant quantity by hand. This has general solution

$$u(t) = \phi(t, u_0) = \frac{u_0 e^t}{u_0(e^t - 1) + 1}. \tag{3.103}$$

Let $u_0 = 1/2$, such that $u(t) = e^t/(e^t+1)$. Since $1/2 \leq u < 1$, $J = 1-2u$, and $J' = 2u\,(u-1)$, then $0 \leq \|J\| < 1$ and $0 < \|J'\| \leq 1/2$, i.e. $\mu = 1$ and $\nu = 1/2$. Let L be the number of coarse time steps on a time chunk. We set Implicit Euler for both fine and coarse solvers, and solve the logistic equation using Parareal with $\xi = 10$ and $L = 100$, such that $\Delta T = L\xi h = 10^3 h$. Theorem 3.3.2 implies that

$$\delta \leq 10^3 h^2 \left(\frac{1}{1-h} + \frac{10}{1-10h} + \frac{27}{4} \right) + \mathcal{O}(h^3) \tag{3.104}$$

for $h < 10^{-1}$. We take $N = 10$, such that the overall time domain is $[0, 10^4 h]$. Following [64], we have

$$\Lambda_G = \frac{1}{1-\mu\Delta T} = \frac{1}{1-10^3 h}, \tag{3.105}$$

for $\mu\Delta T < 1$, i.e. $h < 10^{-3}$. Hence, the error amplification term is

$$s(\Lambda_G) = \frac{\Lambda_G^N - 1}{\Lambda_G - 1} = \frac{(1-10^3 h)^{-10} - 1}{10^3 h (1-10^3 h)^{-1}}. \tag{3.106}$$

Putting everything back together, we get that $\beta < 1$ when $h < 4.282 \times 10^{-4}$. To evaluate such a bound for h, we plot the actual convergence of Parareal in Figure 3.4. As we can see, the bound becomes tighter and tighter as $h \to 0$, but it becomes too loose as h increases above the predicted limit. Hence, our estimate is too conservative. As a side note, notice that for $\|U^k - U^*\|$ very small (around 10^{-15}) we have reached the level of the rounding errors of the finite arithmetics used. Therefore, we cap the bound at $\sqrt{dN}\epsilon$, where ϵ denotes the machine epsilon.

Next, we repeat the same experiments on the Lorenz equations (2.29). We take $x(0) = 2$, $y(0) = 3$, $z(0) = -14$ as the initial condition. Unlike the logistic case, it is not straightforward to get some a-priori bounds for μ and ν. Nevertheless, the Jacobian of f and its time derivative can be expressed explicitly as

$$J(t) = \mathcal{D}f(u(t)) = \begin{bmatrix} -\sigma & \sigma & 0 \\ \rho - z(t) & -1 & -x(t) \\ y(t) & x(t) & -\beta \end{bmatrix} \tag{3.107}$$

and

$$J'(t) = \left(x' \frac{\partial}{\partial x} + y' \frac{\partial}{\partial y} + z' \frac{\partial}{\partial z} \right) \mathcal{D}f(u(t)) \tag{3.108}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ \beta z(t) - x(t)y(t) & 0 & \sigma(x(t) - y(t)) \\ x(t)(\rho - z(t)) - y(t) & \sigma(y(t) - x(t)) & 0 \end{bmatrix}, \tag{3.109}$$

such that we can compute an estimate μ and ν as we carry out the iterations of Parareal. We show our results in Figure 3.5. Once again, the bound becomes better as $h \to 0$ and too pessimistic as h increases.

We conclude by showing some tests on the Lorenz-96 equations (2.30). As in the Lorenz case, we can express $J(t)$ and $J'(t)$ explicitly, so that estimates for μ and ν can be obtained while running our simulations. We use a representative initial condition known to lie on the attractor. We show our results in Figure 3.6, which are once again consistent with what we obtained in the previous two cases.

Our numerical experiments show that our bound for β is indeed quite conservative when it comes to moderately sized h. On the upside, it works quite well for small values of h, which is extremely valuable in turbulent simulations. In those scenarios, the fine solver needs to be extremely accurate anyway to be able to capture physical phenomena that happen on very fine time scales: as we have mentioned in Section 1.3, the turbulence responsible for the plasma transport evolves on a timescale 10^6–10^9 times smaller than that of its confinement.

Hence, our experiments suggest that Theorem 3.3.2 might be really useful in this context.

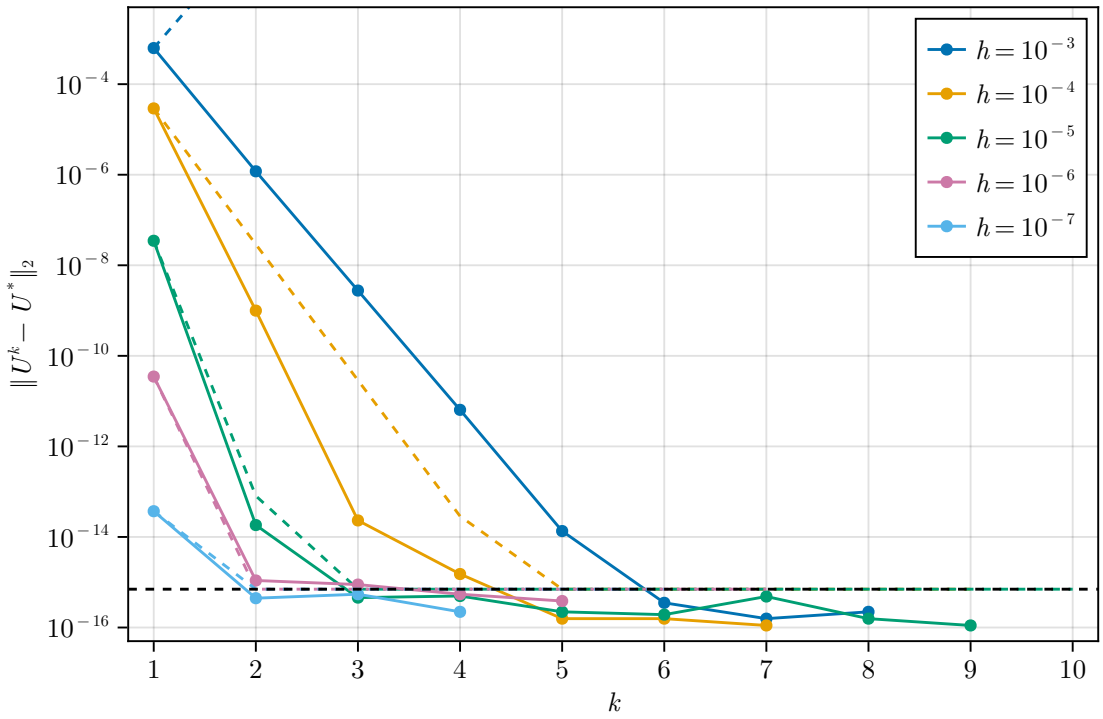

Figure 3.4: Convergence of Parareal with Implicit Euler for the logistic equation, as we vary the fine step size h. We compare the actual convergence of Parareal (solid lines) with our upper bound for β from Theorems 3.3.1 and 3.3.2 (dashed lines).

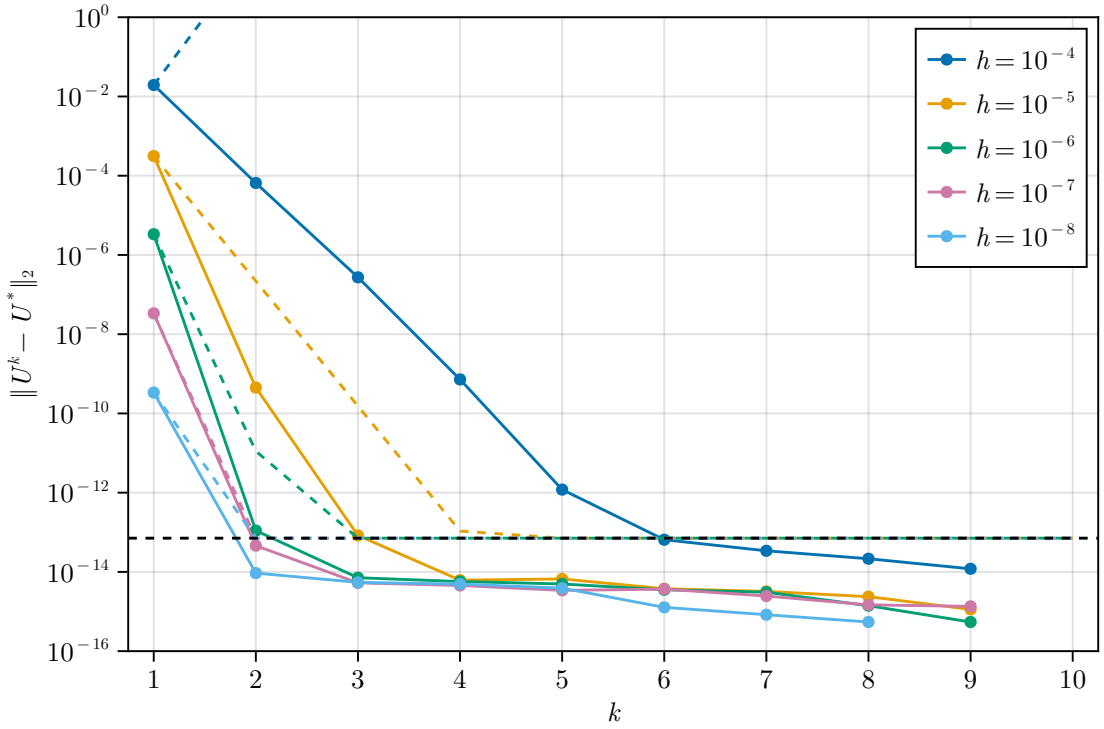

Figure 3.5: Convergence of Parareal with Implicit Euler for the Lorenz equations, as we vary the fine step size h. We compare the actual convergence of Parareal (solid lines) with our upper bound for β from Theorems 3.3.1 and 3.3.2 (dashed lines).

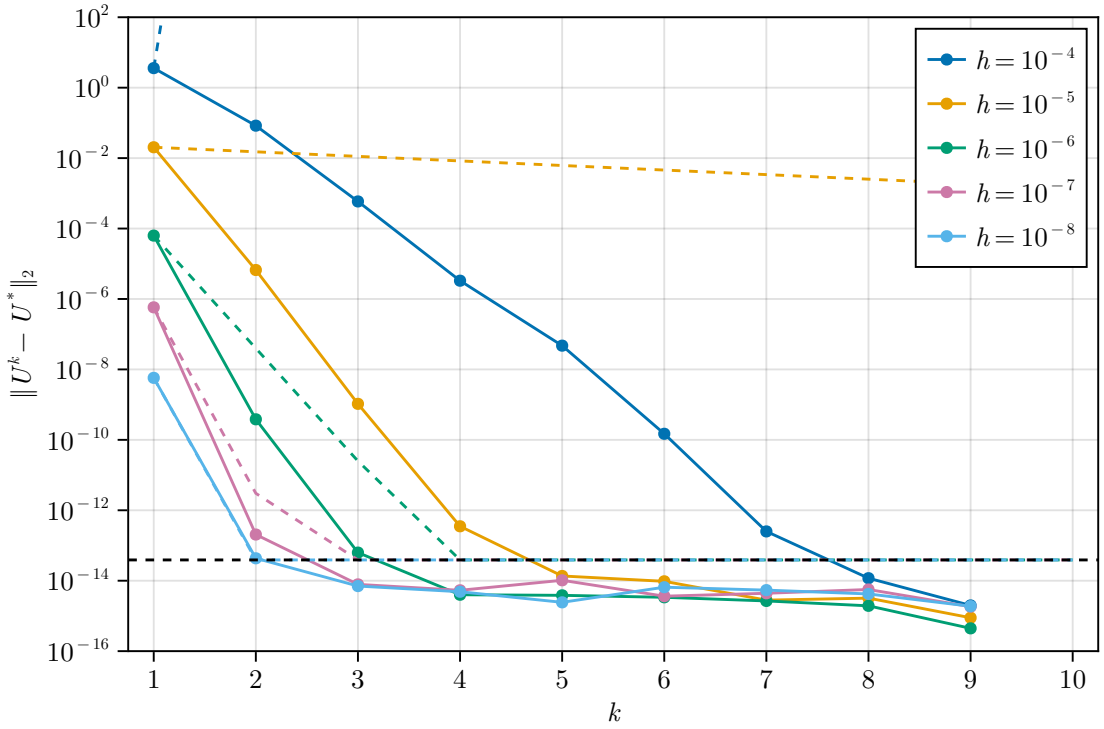

Figure 3.6: Convergence of Parareal with Implicit Euler for the Lorenz-96 equations, as we vary the fine step size h. We compare the actual convergence of Parareal (solid lines) with our upper bound for β from Theorems 3.3.1 and 3.3.2 (dashed lines).

3.4 The need for a proximity function

Until now, we have talked about convergence in terms of $\|U^k - U^*\|$. Since we do not know U^* a priori, we cannot compute $\|U^k - U^*\|$ directly. Therefore, we need an alternative way to check how close we are to the target solution at a given iteration k. For Parareal, the most simple and widely used method to check for convergence is to compute the relative normed difference between two consecutive iterates [6, 119], i.e.

$$\frac{\|U_n^k - U_n^{k-1}\|}{\|U_n^{k-1}\|}. \tag{3.110}$$

Such a method has the advantage of being easy to implement, and it can be done during a simulation. However, it can lead to substantial over-resolution of the problem, higher iteration counts, and thus lower parallel speedup, especially if one is interested only in averaged dynamics [24], like in tokamak simulations.

Instead, we propose to use the proximity function

$$\psi(U) = \frac{1}{N}\sum_{n=1}^{N} \|W_n(U_n - F(U_{n-1}))\|, \tag{3.111}$$

where W_n $(n = 1, \dots, N)$ are symmetric positive definite matrices that act as weights for the solution discontinuities between adjacent time chunks[1].

The exact form of W_n should depend on the kind of problem addressed. In the context of chaotic ODEs, we propose to use the largest positive Lyapunov exponent (LE) λ of the system, which determines the maximal rate of divergence of trajectories, setting

$$W_n = e^{-\lambda(T_n - T_0)} I_d = e^{-\lambda n \Delta T} I_d, \tag{3.112}$$

where $I_d \in \mathbb{R}^{d\times d}$ is the identity matrix. In doing so, when λ is large, our proximity function mainly focuses on the early time chunks. This means that even if there are discrepancies in the solution at later time chunks, we can still deem our solution converged. Our choice is especially justified in the context of chaotic systems, where we are concerned with the statistical properties of our solution and where dynamics are constrained in some manner (such as having an attractor that limits blow-ups). In fact, the exponential divergence of trajectories makes it hard to minimize solution discontinuities at later time chunks. Since the exponential divergence of trajectories is determined by λ, which describes the rate of growth of vectors in the tangent space

[1] Similar functions have been adopted in [21, 91, 108], although designed as objective functions to derive new optimization methods related to Parareal, suitable for optimal control and weather forecasting.

of the phase space, it is reasonable to assume that the weights W_n should be scaled accordingly. In the experiments section, we will present results that support our claims. Note that I_d can be replaced with the relevant spatial operator in PDE applications.

There are some drawbacks in using λ. For starters, LEs are difficult to calculate analytically. One option is to instead use an upper bound, say μ, of the (log-)norm of the Jacobian $J(t)$ of the right-hand side function of the IVP. In fact, $\lambda \leq \mu$ [82], and μ is often easier and cheaper to calculate than λ. More importantly, however, LEs are asymptotic quantities, whereas our focus is on finite time domains: it might be more meaningful to use localized alternatives to global LEs. One possibility is to use the maximal finite-time LE (FTLE) $\lambda_{\Delta T}$, as described in Section 2.3. There are several, convenient algorithms for the evaluation of (FT)LEs, such as [10, 11, 31], but their cost adds to the overall sequential load. Luckily, we can avoid computing $\lambda_{\Delta T}$ altogether. In fact, since $U_n^* = F(U_{n-1}^*)$, then

$$\lambda_{\Delta T} = \frac{1}{\Delta T} \lim_{\delta U_{n-1}^* \to 0} \log\left(\frac{\|\delta U_n^*\|}{\|\delta U_{n-1}^*\|}\right) \tag{3.113}$$

$$= \frac{1}{\Delta T} \lim_{\delta U_{n-1}^* \to 0} \log\left(\frac{\|F(U_{n-1}^* + \delta U_{n-1}^*) - F(U_{n-1}^*)\|}{\|\delta U_{n-1}^*\|}\right). \tag{3.114}$$

But $\|U_n^* - V_n^*\| = \|F(U_{n-1}^*) - F(V_{n-1}^*)\| \leq \Lambda_F \|U_{n-1}^* - V_{n-1}^*\|$ for all $0 \leq t \leq \Delta T$, so

$$\lambda_{\Delta T} \leq \frac{\log(\Lambda_F)}{\Delta T}, \tag{3.115}$$

that is

$$e^{\lambda_{\Delta T} \Delta T} \leq \Lambda_F. \tag{3.116}$$

This means that we can employ the Lipschitz constant Λ_F of the fine solver in our definition of the weight used for the proximity function, such that

$$W_n = \Lambda_F^{-n} I_d. \tag{3.117}$$

Choosing to do so is particularly convenient because Parareal computes $N - k$ values U_n^k and as many $F(U_n^k)$ at each iteration k. We can therefore estimate Λ_F from the evaluations of $\|F(U_i^k) - F(U_j^k)\| / \|U_i^k - U_j^k\|$. Estimates of the Lipschitz constant based on function evaluations have been suggested and analysed by several authors; see [139] and references therein. In practice, however, we find that taking the maximum of such terms, so that we obtain an underestimate of Λ_F, has the advantage of being essentially cost-free, and works well enough for our purposes.

Anyway, due to the clear availability of options, we keep the discussion as general as possible and set

$$W_n = w^{-n} I_d, \tag{3.118}$$

for some $w > 0$. From now on, we refer to this scalar w as the *base weight* of the proximity measure ψ.

Another important point to add here is that ψ is closely related to the standard normed difference from the fine solution. To see how, let

$$W = \begin{bmatrix} W_1 & & \\ & \ddots & \\ & & W_N \end{bmatrix} \in \mathbb{R}^{dN \times dN}. \tag{3.119}$$

Since W is a block matrix with symmetric positive definite blocks, then $\|\cdot\|_W = \|W(\,\cdot\,)\|$ is a norm. $\|\cdot\|_W$ itself is equivalent to any other norm $\|\cdot\|$, being $\mathbb{R}^{dN}$ finite-dimensional. At the same time, as we will show in the following theorem, whenever $\psi(U^k)$ goes to zero, $\|U^k - U^*\|_W$ follows, and vice versa (at least in the context of p-norms).

Theorem 3.4.1. Let $\|\cdot\|$ be a p-norm. Assume that F satisfies a Lipschitz condition in $\|\cdot\|$ with constant Λ_F. Define $\theta_F = \Lambda_F / w$. Then

$$c\psi(U) \le \|U - U^*\|_W \le C_N \psi(U), \tag{3.120}$$

where

$$c = \frac{1}{1+\theta_F}, \quad C_N = N \cdot \begin{cases} \frac{\theta_F^N - 1}{\theta_F - 1}, & \theta_F > 1, \\ N, & \theta_F = 1, \\ 1, & \theta_F < 1. \end{cases} \tag{3.121}$$

In particular, define

$$\psi^*(U) = \frac{1}{N} \sum_{n=1}^{N} w^{-n} \|U_n - U_n^*\|. \tag{3.122}$$

Then

$$c\psi(U) \le \psi^*(U) \le \|U - U^*\|_W \le N\psi^*(U) \le C_N \psi(U). \tag{3.123}$$

Proof. It is enough to prove the four inequalities in (3.123) in order of appearance. (1) Since F is Lipschitz and $U_n^* = F(U_{n-1}^*)$,

$$\psi(U) = \frac{1}{N} \sum_{n=1}^{N} w^{-n} \|U_n - F(U_{n-1})\| \tag{3.124}$$

$$\le \frac{1}{N} \sum_{n=1}^{N} w^{-n} \big(\|U_n - U_n^*\| + \Lambda_F \|U_{n-1} - U_{n-1}^*\| \big) \tag{3.125}$$

$$\leq \frac{1}{N}\sum_{n=1}^{N}\big(w^{-n}+w^{-(n+1)}\Lambda_F\big)\|U_n-U_n^*\| \tag{3.126}$$

$$= (1+\theta_F)\psi^*(U). \tag{3.127}$$

(2) By definition,

$$\|U-U^*\|_W = \left(\sum_{n=1}^{N} w^{-n}\|U_n-U_n^*\|^p\right)^{1/p}. \tag{3.128}$$

Using Hölder's inequality,

$$\psi^*(U) = \frac{1}{N}\sum_{n=1}^{N} w^{-n}\|U_n-U_n^*\| \leq N^{-\frac{1}{p}}\|U-U^*\|_W \leq \|U-U^*\|_W. \tag{3.129}$$

(3) Using the triangle inequality,

$$\|U-U^*\|_W \leq \sum_{n=1}^{N} w^{-n}\|U_n-U_n^*\| = N\psi^*(U). \tag{3.130}$$

(4) Using again the Lipschitz continuity of F,

$$N\psi^*(U) \leq \sum_{n=1}^{N} w^{-n}\left(\|U_n-F(U_{n-1})\|+\Lambda_F\|U_{n-1}-U_{n-1}^*\|\right) \tag{3.131}$$

$$\leq \sum_{n=1}^{N} w^{-n}\sum_{j=0}^{n}\Lambda_F^j\|U_{n-j}-F(U_{n-j-1})\| \tag{3.132}$$

$$= \sum_{n=1}^{N}\Lambda_F^{-n}\|U_n-F(U_{n-1})\|\sum_{j=0}^{N-n}\theta_F^{N-j}. \tag{3.133}$$

Using the geometric sum,

$$\sum_{j=0}^{N-n}\theta_F^{N-j} = \theta_F^n\cdot\begin{cases}\frac{\theta_F^{N-n+1}-1}{\theta_F-1}, & \theta_F\neq 1,\\ N-n+1, & \theta_F = 1.\end{cases} \tag{3.134}$$

Maximizing over $n\in\{1,\ldots,N\}$ gives the required result. □

Although C_N is independent of U, it does depend on N, i.e. the number of time chunks into which we subdivide the time domain. This suggests that, as N increases, it becomes more and more difficult to drive down $\|U^k-U^*\|_W$ through $\psi(U^k)$. In practice, one usually sets N to be equal to the number of processes available to perform time parallelization. Due to the limitations imposed by the communication network, such a number is usually around 10^2 to 10^3 in relevant applications.

3.4.1 Experiments

In this section, we run some experiments on the same example problems used in Section 3.3.2, i.e. the logistic, Lorenz and Lorenz-96 equations, comparing ψ against the standard check (3.110) from the literature.

We run the following simulations. First, we show the outcomes of employing the standard check, in which no weighting is involved. Next, we show the outcomes relative to ψ with $w = 1$, with $w = e^{\max(\lambda,0)}$, and with $w = \Lambda_F$. In all simulations, we set the tolerance to $\varepsilon = 10^{-12}$.

Note that for the logistic equation

$$\lim_{t\to\infty}\lim_{\epsilon\to 0}\frac{1}{t}\ln\left|\frac{\phi(t,u_0+\epsilon)-\phi(t,u_0)}{\epsilon}\right| = -1, \tag{3.135}$$

so that we do not run the simulations for the weighted proximity function in this case, as the system is not chaotic and there is no exponential divergence to weight out.

We use the following notation: N denotes the number of time chunks (or processors) used, K denotes the total number of iterations to convergence, and S denotes the overall speedup achieved. Also, Δ denotes the distance between the fine solution and the parallel solution. We assess this distance using two methods. For the logistic case, we simply compute the 2-norm of the difference between the fine and parallel solutions. For the Lorenz and Lorenz-96 cases, we require a more robust statistical comparison. Simple comparisons of low-order moments (mean and variance) are insufficient, as different distributions can share identical moments. Furthermore, metrics like the Kullback–Leibler divergence are brittle for finite-sample data; if one simulation produces a value where the other has zero probability (disjoint support), the divergence becomes infinite.

To overcome these limitations, we use the 1-Wasserstein distance between the probability density functions of the fine and parallel solutions. The 1-Wasserstein distance between two probability distributions, P_1 and P_2, denoted by $W_1(P_1, P_2)$, quantifies the minimum cost of transforming P_1 into P_2 with respect to a given cost function. This distance metric provides a direct and intuitive comparison between distributions, with smaller values indicating greater likeness. In our implementation, we compute this distance by discretizing the distributions into normalized histograms with a fixed bin width over a bounded domain. We treat these histograms as discrete measures. The 1-Wasserstein distance is then computed as the optimal transport cost between these discrete measures.

We collect our results in Tables 3.7 and 3.8. In the case of the logistic equation, we can see that both the standard check and the unweighted proximity function yield similar performance. This is to be expected, given our earlier analysis. The same applies for the Lorenz and Lorenz-96 systems. However, the situation changes when looking at the weighted proximity function simulations: we notice a significant difference in the total number of iterations when compared to the unweighted cases, with little change in the distance between fine and parallel solutions. These findings seem to suggest that the weighting established with ψ has indeed accelerated convergence, increased parallel speedup, and resulted in similar PDFs. It seems that, therefore, ψ has the potential to benefit the time-parallel community in context where statistics and averaged dynamics are more important than the pointwise convergence, such as in [119] and [24].

3.4.2 Further experiments

The benefits of the weighted proximity function become starkly clear when we analyse the behaviour of the Parareal algorithm across different problem scales. The following figures (Figures 3.10 to 3.12) compare the performance of the standard convergence check against our proximity function with two weighting schemes: one based on the Lyapunov exponent, that is $w = e^{\lambda \Delta T}$, and one on the fine solver's Lipschitz constant, that is $w = \Lambda_F$. We analyse three key metrics: the number of iterations to convergence K, the effective serial work $K\Delta T$, and the (theoretical) parallel speedup S.

We start with the iteration count K which, for speedup to be possible, must be significantly smaller than the number of chunks N. Figure 3.10 shows the iteration count under three scenarios:

- **Top panels (fixed N, increasing T):** With the standard check (blue line), K grows quickly with the total integration time T before saturating near $K = N$. This is a catastrophic failure of the method, as it implies that for long simulations, Parareal requires almost as many iterations as there are processors, yielding no parallel benefit. In contrast, both weighted proximity functions (red and green lines) keep the iteration count low and nearly constant, regardless of T.

- **Middle and bottom panels (increasing N):** The standard check show K increasing linearly with N. This means that adding more processors makes the algorithm take more iterations, which is a fundamental scaling problem. Again, the weighted methods maintain a small, constant number of iterations even as N grows to 100.

		strong scaling (T fixed)				weak scaling ($T \propto N$)			
	N	K	S_{theo}	S_{true}	$\|\cdot\|_2$	K	S_{theo}	S_{true}	$\|\cdot\|_2$
standard check	2	1	1.96	1.07	1.39×10^{-11}	2	1.31	13.93	2.48×10^{-15}
	4	1	3.85	0.98	1.59×10^{-11}	3	1.71	0.49	4.66×10^{-15}
	8	1	7.41	0.74	3.57×10^{-11}	3	2.82	0.38	2.00×10^{-14}
	16	3	4.90	0.31	1.52×10^{-11}	3	4.90	0.35	2.62×10^{-14}
	32	3	8.34	0.30	1.61×10^{-11}	3	8.34	0.34	3.17×10^{-14}
	64	3	13.21	0.31	1.65×10^{-11}	3	13.21	0.28	9.21×10^{-12}
	128	3	18.86	0.29	1.67×10^{-11}	3	18.86	0.30	1.67×10^{-11}
unweighted ψ	2	1	1.96	0.99	1.39×10^{-11}	2	1.31	0.69	2.48×10^{-15}
	4	1	3.85	1.32	1.59×10^{-11}	2	2.20	0.57	1.16×10^{-13}
	8	1	7.41	0.95	3.57×10^{-11}	2	3.95	0.52	4.96×10^{-13}
	16	2	7.12	0.51	1.60×10^{-11}	2	7.12	0.26	5.18×10^{-13}
	32	2	12.31	0.50	1.65×10^{-11}	2	12.31	0.50	5.21×10^{-13}
	64	2	19.67	0.49	1.67×10^{-11}	2	19.67	0.79	9.50×10^{-12}
	128	2	28.18	0.42	1.68×10^{-11}	2	28.18	0.48	1.68×10^{-11}

Table 3.7: Simulations for the logistic equation, running Parareal with RK4 for both fine and coarse solvers, with $h = 10^{-3}$, $\xi = 100$, and $\epsilon = 10^{-12}$.

		strong scaling (T fixed)				weak scaling ($T \propto N$)			
	N	K	S_{theo}	S_{true}	$W_1(\cdot)$	K	S_{theo}	S_{true}	$W_1(\cdot)$
standard check	2	2	1.31	0.83	2.03×10^{-5}	2	1.31	0.67	5.22×10^{-4}
	4	4	1.54	0.63	1.11	4	1.54	0.38	0.00
	8	8	1.65	0.20	1.04	4	2.28	0.30	0.00
	16	16	1.62	0.11	1.75×10^{-1}	5	3.15	0.19	0.00
	32	29	1.49	0.07	5.89×10^{-1}	20	1.72	0.06	5.14×10^{-4}
	64	56	1.22	0.04	1.21×10^{-1}	47	1.30	0.04	2.77×10^{-1}
	128	112	0.88	0.02	3.00×10^{-1}	112	0.88	0.02	3.00×10^{-1}
unweighted ψ	2	2	1.31	0.54	2.03×10^{-5}	2	1.31	0.69	5.22×10^{-4}
	4	4	1.54	0.40	1.11	3	1.71	0.45	0.00
	8	8	1.65	0.31	1.04	3	2.82	0.38	0.00
	16	16	1.62	0.11	1.75×10^{-1}	4	3.80	0.26	0.00
	32	28	1.50	0.06	5.89×10^{-1}	9	3.08	0.11	5.09×10^{-4}
	64	52	1.25	0.04	1.21×10^{-1}	39	1.42	0.03	2.77×10^{-1}
	128	102	0.91	0.01	3.00×10^{-1}	102	0.91	0.02	3.00×10^{-1}
ψ with $w = e^{\max(\lambda,0)}$	2	1	1.96	0.73	3.61×10^{-1}	2	1.31	0.67	5.22×10^{-4}
	4	1	3.85	0.99	2.42×10^{-1}	3	1.71	0.44	0.00
	8	2	3.95	0.40	1.60	3	2.82	0.38	0.00
	16	2	7.12	0.76	1.15	3	4.90	0.11	9.45×10^{-6}
	32	3	8.34	0.29	1.12	3	8.34	0.55	6.85×10^{-1}
	64	3	13.21	0.34	5.05×10^{-1}	3	13.21	0.28	1.04×10^{-1}
	128	3	18.86	0.50	6.69×10^{-1}	3	18.86	0.42	6.69×10^{-1}
ψ with $w = \Lambda_F$	2	2	1.31	0.63	2.03×10^{-5}	2	1.31	0.68	5.22×10^{-4}
	4	1	3.85	0.50	2.42×10^{-1}	2	2.20	0.58	0.00
	8	2	3.95	0.41	1.60	3	2.82	0.37	0.00
	16	2	7.12	0.77	1.15	3	4.90	0.35	9.45×10^{-6}
	32	2	12.31	0.33	1.31	2	12.31	0.22	5.43×10^{-1}
	64	2	19.67	0.50	6.73×10^{-1}	2	19.67	1.32	2.72×10^{-1}
	128	2	28.18	0.62	1.57×10^{-1}	2	28.18	0.66	1.57×10^{-1}

Table 3.8: Simulations for the Lorenz equation, running Parareal with RK4 for both fine and coarse solvers, with $h = 10^{-4}$, $\xi = 100$, and $\epsilon = 10^{-9}$.

		strong scaling (T fixed)				weak scaling ($T \propto N$)			
	N	K	S_{theo}	S_{true}	$W_1(\cdot)$	K	S_{theo}	S_{true}	$W_1(\cdot)$
standard check	2	2	1.31	0.68	3.02×10^{-1}	1	1.96	1.05	1.19×10^{-4}
	4	4	1.54	0.43	3.27×10^{-1}	3	1.71	0.43	0.00
	8	8	1.65	0.19	2.27×10^{-1}	4	2.28	0.30	0.00
	16	16	1.62	0.10	1.45×10^{-1}	6	2.72	0.20	6.46×10^{-4}
	32	32	1.47	0.07	2.59×10^{-1}	22	1.64	0.06	2.04×10^{-1}
	64	59	1.21	0.03	4.18×10^{-1}	53	1.24	0.03	1.49×10^{-1}
	128	113	0.88	0.02	2.13×10^{-1}	113	0.88	0.01	2.13×10^{-1}
unweighted ψ	2	2	1.31	0.35	3.02×10^{-1}	1	1.96	1.00	1.19×10^{-4}
	4	4	1.54	0.23	3.27×10^{-1}	3	1.71	0.45	0.00
	8	8	1.65	0.23	2.27×10^{-1}	3	2.82	0.38	0.00
	16	16	1.62	0.12	1.45×10^{-1}	4	3.80	0.27	6.05×10^{-4}
	32	32	1.47	0.05	2.59×10^{-1}	17	1.90	0.07	2.01×10^{-1}
	64	57	1.22	0.03	4.18×10^{-1}	49	1.27	0.03	1.49×10^{-1}
	128	109	0.89	0.02	2.13×10^{-1}	109	0.89	0.02	2.13×10^{-1}
ψ with $w = \Lambda_F$	2	2	1.31	0.49	3.02×10^{-1}	1	1.96	1.03	1.19×10^{-4}
	4	2	2.20	0.36	3.58×10^{-1}	2	2.20	0.59	0.00
	8	4	2.28	0.41	2.99×10^{-1}	2	3.95	0.53	1.26×10^{-5}
	16	5	3.15	0.19	1.88×10^{-1}	2	7.12	1.33	2.73×10^{-1}
	32	5	5.17	0.14	3.38×10^{-1}	2	12.31	0.93	6.46×10^{-1}
	64	4	9.99	0.17	1.58×10^{-1}	2	19.67	0.76	2.37×10^{-1}
	128	2	28.18	0.83	3.25×10^{-1}	2	28.18	0.64	3.25×10^{-1}

Table 3.9: Simulations for the Lorenz-96 equation, running Parareal with RK4 for both fine and coarse solvers, with $h = 10^{-4}$, $\xi = 100$, and $\epsilon = 10^{-9}$.

The key takeaway is that the weighted proximity function decouples the iteration count from the problem size. By focusing on accuracy in the early, predictable time chunks and allowing for divergence later (where chaos dominates anyway), it prevents the algorithm from chasing pointless pointwise accuracy, thus keeping K small.

A more sophisticated metric is the effective serial work, $K\Delta T$. This quantity represents the total time interval processed serially by the coarse solver during the correction steps. According to Amdahl's law, this serial portion is the ultimate bottleneck for parallel scalability. An ideal time-parallel method should have a serial work cost that is bounded and, crucially, independent of the total simulation time T. Figure 3.11 demonstrates the most significant advantage of our approach.

- **Top panels (fixed N, increasing T):** For the standard check, $K\Delta T$ grows linearly with T. This confirms it is not a scalable approach; the serial bottleneck grows in direct proportion to the problem size. For the weighted proximity functions, $K\Delta T$ remains low and remarkably flat. This shows that the serial work is bounded and independent of the total integration time T.

- **Bottom panels ($T = 5N$, so ΔT is constant):** The standard check's serial work again grows linearly with the problem size, while the weighted methods keep it low and constant.

Finally, we examine the parallel speedup S. This measures the wall-clock time reduction compared to a purely sequential fine solve. Figure 3.12 shows that the performance of the standard check (blue line) is consistently poor. Speedup quickly drops towards 1 (meaning no benefit) as either T or N increases. On the other hand, the weighted methods, particularly $w = e^{\lambda\Delta T}$ (red line), show significant and sustained speedup.

In summary, the analysis of these three metrics provides evidence for the merits of the weighted proximity function. By focusing on short-term accuracy while accepting long-term statistical similarity, it solves the fundamental scaling problems that plague standard implementations of Parareal, turning it into a genuinely scalable method for long-time chaotic simulations.

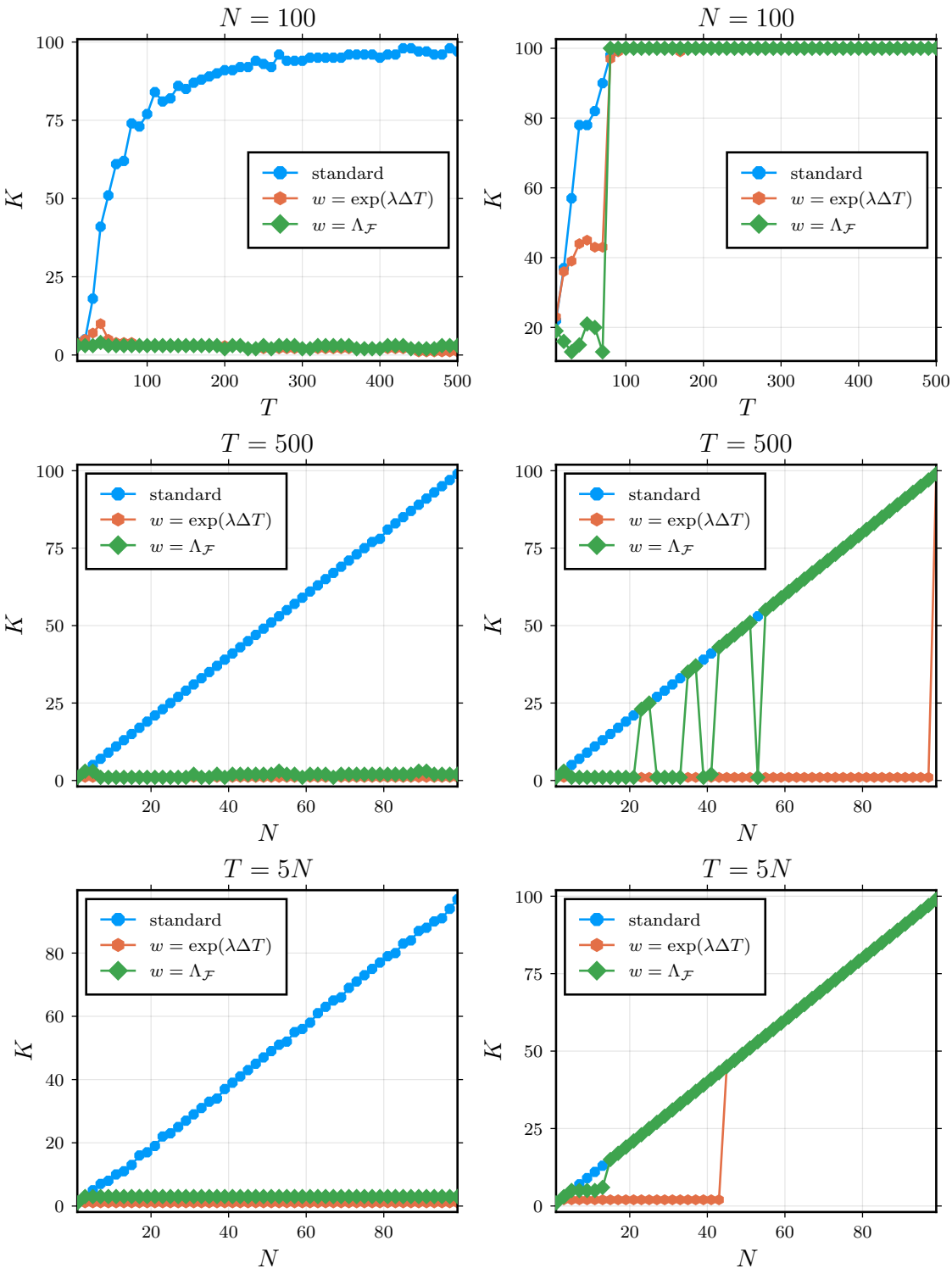

Figure 3.10: Number of Parareal iterations K to convergence for the Lorenz system, comparing the standard check (blue) with the weighted proximity functions. Here, (left) $\xi = 10$, (right) $\xi = 100$. (top) T increasing, N fixed; (middle) T fixed, N increasing; (bottom) T increasing linearly with N. In all scenarios, the weighted proximity functions maintain a small and nearly constant number of iterations, effectively decoupling the algorithmic cost from the problem scale.

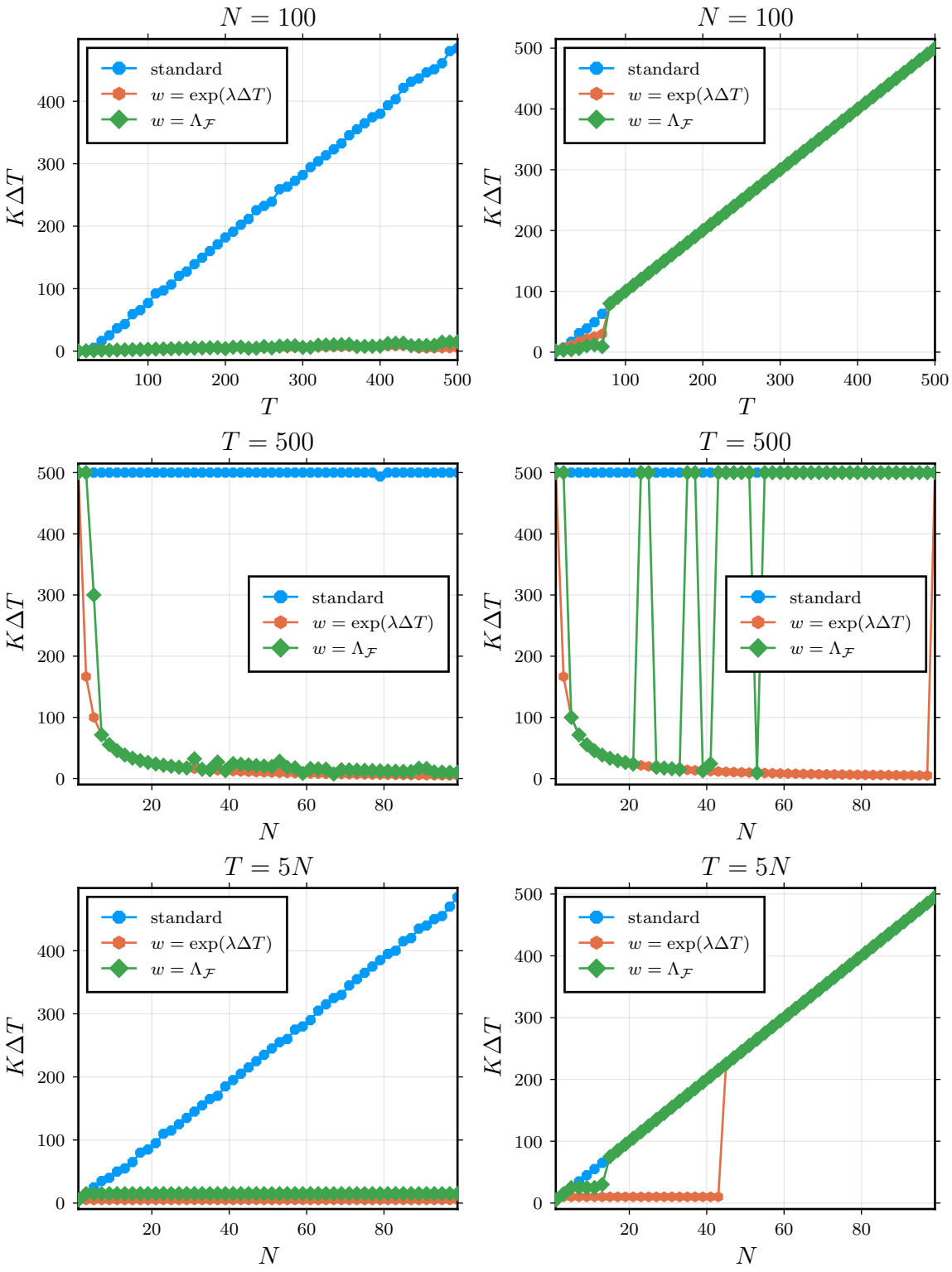

Figure 3.11: Effective serial work $K\Delta T$, which represents the sequential bottleneck of the Parareal algorithm. Here, (left) $\xi = 10$, (right) $\xi = 100$. (top) T increasing, N fixed; (middle) T fixed, N increasing; (bottom) T increasing linearly with N. For the weighted methods, the serial portion of the algorithm does not grow with the overall problem size, a critical property for parallel scalability.

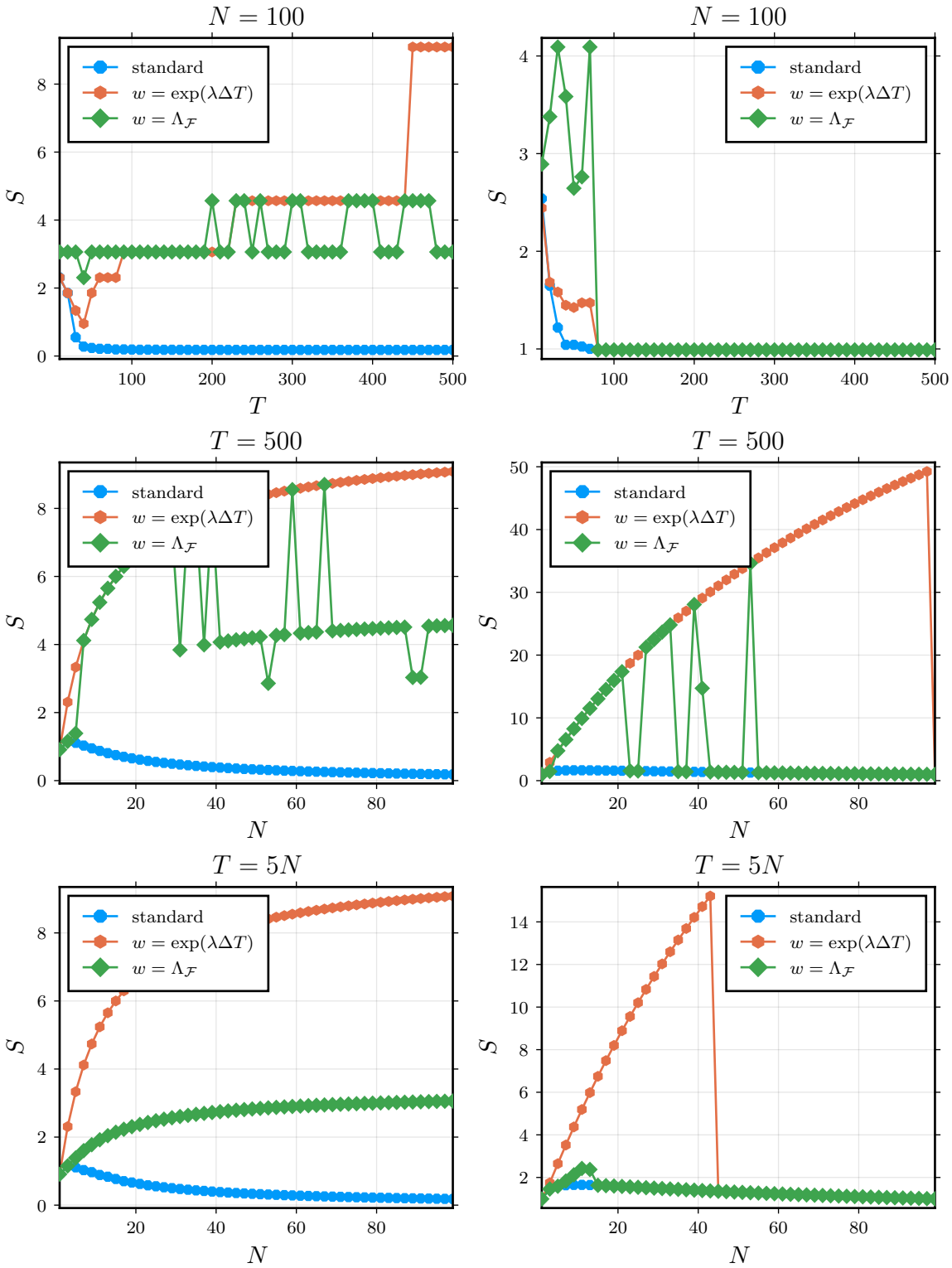

Figure 3.12: Parallel speedup S achieved with different convergence criteria for the Lorenz system. Here, (left) $\xi = 10$, (right) $\xi = 100$. (top) T increasing, N fixed; (middle) T fixed, N increasing; (bottom) T increasing linearly with N. The weighted methods achieve significant and sustained speedup.

Chapter 4

MoWi

In the context of chaotic systems, especially ergodic, one often runs long-time simulations to get statistically reliable estimates of the physical quantities of interest. Rather than focussing on individual trajectories from arbitrarily chosen initial conditions, often the main goal is to determine properties of the steady state or attractor. To analyse stability and transport properties, we rely on statistical measures such as mean, variance, and standard deviation. For example, in gyrokinetic simulations, the mean of electron-density fluctuations determines the background density profile, while the mean plasma-flow velocity governs neoclassical and turbulent transport, both essential for understanding steady-state equilibrium. On the other hand, variance and standard deviation measure the intensity of fluctuations, which is important for identifying instability risks. In fusion plasmas, in fact, excessively large fluctuations in temperature and density can trigger edge-localized modes (ELMs) or even lead to plasma disruptions.

As we have seen, however, the number of iterations needed for time-parallel methods like Parareal to converge increases with the length of the time domain. This effect is even stronger in turbulent problems, where the sensitive dependence on initial conditions makes it harder to minimize solution discontinuities at the time-chunk endpoints – see Section 3.4 for some experimental backing. To tackle this issue, we introduce a procedure for the long-time integration of chaotic systems that uses time parallelization as a subroutine. For reasons that will shortly become clear, we call this the moving-window algorithm (MoWi).

In particular, we will see that the proximity measure ϕ – introduced in Section 3.4 – naturally lends itself to the *tracking analysis* of MoWi, a kind of analysis that examines how well statistics are estimated while making sure that the time-parallel subroutine converges within each time window of MoWi and that the resulting sequence of window solutions remains physically meaningful according to the chosen criterion.

We will extend MoWi with adaptive strategies in Chapter 5 to handle varying chaoticity.

4.1 Description

In this section, we explain how MoWi works. Our goal is to solve an IVP of form

$$\begin{cases} u'(t) = f(u(t), t), & t \in [0, T], \\ u(0) = u_0, \end{cases} \tag{2.12}$$

for $T > 0$ so large that Parareal would take an unfeasibly high number of iterations to converge (relative to the speedup we aim for). Instead of solving the IVP over the full time domain all at once, we split it into smaller time intervals, called (*time*) *windows*. This way, Parareal converges in a smaller number of iterations over each window. Moreover, we ensure that each window overlaps with its nearest neighbours, so that we can reuse solutions from previous windows to improve the initial guess for Parareal, which traditionally would otherwise be only a coarse solution, further speeding up its convergence. Formally, we define a collection of windows $\{\mathcal{T}_m : m = 0, \ldots, M > 0\}$ covering the whole time domain, with the following properties:

1. The windows together fully cover the time domain:

$$\bigcup_{m=0}^{M} \mathcal{T}_m \equiv [0, T]. \tag{4.1}$$

2. Each window intersects the next one:

$$\mathcal{T}_m \cup \mathcal{T}_{m+1} \neq \emptyset, \quad \forall m. \tag{4.2}$$

3. Take $\mathcal{T}_m = [t_m^{\text{start}}, t_m^{\text{end}}]$. The windows shift forwards as m grows:

$$t_m^{\text{start}} < t_{m+1}^{\text{start}} < t_m^{\text{end}} < t_{m+1}^{\text{end}}. \tag{4.3}$$

This setup allows us to progressively solve the IVP in overlapping segments, rather than all at once. Next, we describe the steps of MoWi:

1. We solve the IVP over the first window $\mathcal{T}_0$ using Parareal.
    - We make sure that Parareal converges within a fixed number of iterations, say K.

- The output is a vector (of chunks' initial conditions) $U^{K,0}$, which approximates the fine solution vector $U^{*,0}$ over window $\mathcal{T}_0$.

2. We move on to window $\mathcal{T}_1$, shifted forward from $\mathcal{T}_0$ by $\Delta N \leq N$ time chunks.

   - The last $N - \Delta N$ (d-dimensional) entries of $U^{K,0}$ become the first $N - \Delta N$ entries of $U^{0,1}$:

   $$U_n^{0,1} = U_{n+\Delta N}^{K,0}, \quad n = 0, \ldots, N - \Delta N. \tag{4.4}$$

   - We compute the remaining ΔN entries of $U^{0,1}$ using a coarse solver G:

   $$U_n^{0,1} = G(U_{n-1}^{0,1}), \quad n = N - \Delta N + 1, \ldots, N. \tag{4.5}$$

   That is, we complete the remaining time chunks' interface values using the coarse solver. Since we are using Parareal as an example time-parallel subroutine here, we simply use its coarse solver for this task. In general, however, it can be a different solver: it just needs to fill in the remaining time chunk's interface values.

3. We use $U^{0,1}$ as the initial guess for Parareal over window $\mathcal{T}_1$.

   - This speeds up convergence in the new window.

4. We repeat this process for all subsequent windows until we have covered the full time domain.

We give an outline of this procedure in Algorithm 2 and a schematic representation in Figure 4.1.

Also, some remarks. As shown in Figure 4.1, we do not need windows to shift in a way that perfectly aligns their time chunks. In practice, our implementation of MoWi allows for flexible shifts, meaning that we can advance by any amount, as long as it is less than the full length of a time window. In other words, rather than writing this condition as $\Delta N < N$, we should write $\Delta\tau < \tau$, where τ is the length of a window and $\Delta\tau$ is the shift length. However, for the sake of clarity and consistency in our analysis, we assume throughout this discussion that shifts are always aligned with time chunks.

Furthermore, K here corresponds to the (ideal) parallel speedup we want to achieve, as discussed in Section 2.7.2. And although Parareal may satisfy its convergence criterion earlier, stopping at $U^{k,0}$ for some $k \leq K$, we use $U^{K,0}$ for notational simplicity.

Algorithm 2 MoWi with Parareal as a subroutine

for $m = 0, \ldots, M$ **do**
    InitializeWindow(m)
    Parareal(m)
end for

function InitializeWindow(m)
    if $m = 0$ **then**
        ▷ *Do a full coarse solve on the first time window* ◁
        $U_0^{0,0} = u_0$
        for $n = 1, \ldots, N$ **do**
            $U_n^{0,0} = G(U_{n-1}^{0,0})$
        end for
    else
        ▷ ***Shifting*** *- Take overlapping points from the previous window and coarse solve the remaining points* ◁
        for $n = 0, \ldots, N - \Delta N$ **do**
            $U_n^{0,m} = U_{\Delta N+n}^{K,m-1}$
        end for
        for $n = N - \Delta N + 1, \ldots, N$ **do**
            $U_n^{0,m} = G(U_{n-1}^{0,m})$
        end for
    end if
end function

function Parareal(m)
    $k = 0$
    repeat
        for $n = 1, \ldots, N$ **do in parallel**
            $\hat{U}_n^{k,m} = F(U_{n-1}^{k-1,m})$
        end for
        $U_0^{k,m} = U_0^{0,m}$
        for $n = 1, \ldots, N$ **do**
            $U_n^{k,m} = G(U_{n-1}^{k,m}) + \hat{U}_n^{k,m} - G(U_{n-1}^{k-1,m})$
        end for
        $k = k + 1$
    until convergence (according to ψ)
end function

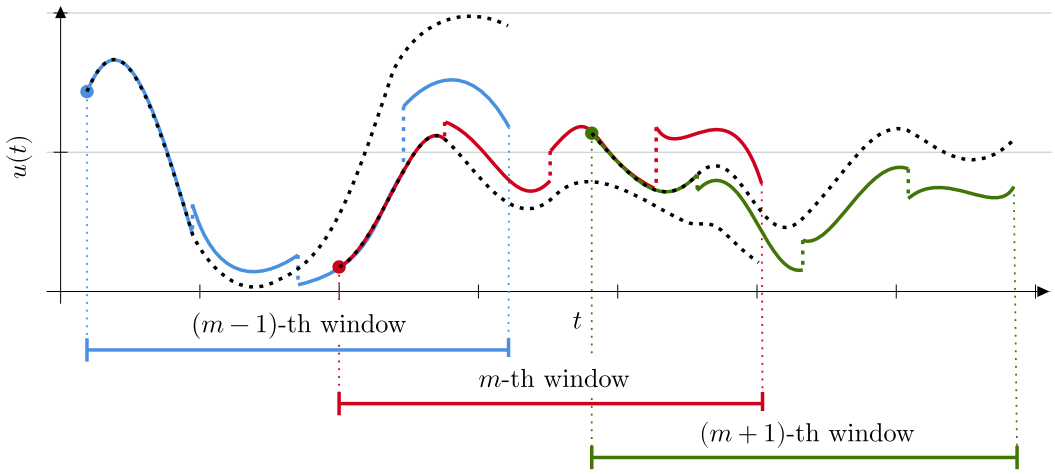

Figure 4.1: A schematic of the MoWi algorithm. The time domain is covered by a series of overlapping windows (blue, red, green). We use the solution from a converged window to provide an initial guess for the subsequent window, and repeat this process until the full domain is integrated.

Finally, distinguishing between $U^{*,0}, U^{*,1}, \ldots$ might seem unnecessary at first. However, the fine solution vector $U^{*,m}$ that Parareal is trying to converge to is not merely a slice of a single trajectory defined over $[0, T]$. Instead, because the initial condition for each window is taken from the previous window's approximate solution, each shift is equivalent to a jump into a new orbit, as shown in Figure 4.1. In other words, MoWi outputs a collection of multiple short-time solutions rather than pieces of the same long-time trajectory. If kept under control, this is actually beneficial, because it allows us to explore different regions of the phase space in one single run, selecting trajectories that are near the attractor (after an initial transient). As such, MoWi can be seen as a method for generating initial conditions that are particularly relevant for the statistical analysis of the dynamics of the system.

4.2 Statistics

Next, we explain how we compute statistical estimates. As shown in the previous section, MoWi generates a collection of finite-time solutions that cover the time domain using overlapping windows. These solutions can be interpreted as an ensemble of time series.

Given M time series $X_1(t), \ldots, X_M(t)$, each defined over the relevant time interval

(the window), we compute the average time series simply as

$$\bar{X}(t) = \frac{1}{M} \sum_{i=1}^{M} X_i(t). \tag{4.6}$$

Similarly, we compute the variance as

$$\sigma^2(t) = \frac{1}{M} \sum_{i=1}^{M} \left(X_i(t) - \bar{X}(t) \right)^2, \tag{4.7}$$

where $\sigma(t) = \sqrt{\sigma^2(t)}$ is the standard deviation. This approach applies equally to any functional dependent on the system state.

Crucially, we must distinguish between the physical fluctuations of the system and the statistical uncertainty of our estimate. The standard deviation $\sigma(t)$ quantifies the intrinsic variability of the attractor; for a chaotic system, this converges to a finite, non-zero constant representing the width of the physical dynamics. However, we are often interested in the precision of the estimated mean $\bar{X}(t)$. This is given by the Standard Error of the Mean (SEM), defined as:

$$\mathrm{SEM}(t) = \frac{\sigma(t)}{\sqrt{M}}. \tag{4.8}$$

Unlike the standard deviation, the SEM decreases as the ensemble size M increases, reflecting the fact that our estimate of the mean becomes more precise with more data.

It is natural to ask whether averaging values from overlapping trajectory segments is a sound way to estimate the system's time-independent statistics, and how this relates to the Ergodic hypothesis. The answer is yes, provided the system is a stationary process. The ergodic hypothesis states that the long-time average of a single trajectory equals the ensemble average over the invariant measure. MoWi effectively estimates this ensemble average directly. It generates an ensemble of M trajectory segments $\{X_i(t)\}$, where each segment serves as a valid sample from the system's attractor.

For a statistically stationary system, the true ensemble mean is constant in time. Hence, as the ensemble size grows, $\bar{X}(t)$ converges to a constant value, which represents our estimate of the system's true invariant mean. This method is a standard statistical estimator for stationary systems, providing a robust and unbiased estimate of the underlying attractor statistics.

Finally, since the ensemble mean $\bar{X}(t)$ is itself a stationary time series, we can further improve our estimate by computing its running time-average over the window duration. This double averaging exploits both the parallel data (ensemble size M)

and the temporal data (window length τ) to maximize precision. In our experimental results, we compare this cumulative time-average of the MoWi ensemble against the standard cumulative time-average of the serial solution.

4.3 Tracking analysis

We now set out to analyse under what circumstances it is guaranteed that MoWi stays on track, meaning that an initial computational cost to set up the algorithm in window $\mathcal{T}_0$, the Parareal iterations on each of the subsequent windows $\mathcal{T}_1, \mathcal{T}_2, \dots$ take at most K iterations to converge.

First, every time we shift from one window to the next, we would like to be able to guarantee that the initial guess of the new window is close enough to its fine solution, so that Parareal converges quickly, and the parallel speed-up is quantifiable a priori. To achieve this, we apply the outer- and inner-ball strategy developed in Section 3.2. In this framework,

- r is the tolerance chosen, e.g., so that functionals of the solution, like ensemble averages, stay within some prescribed numerical error from the experimental value.
- R is the radius of the largest ball containing only points for which Parareal converges to within the tolerance r in at most a fixed number K of iterations.

We denote the inner and outer balls centred at the fine solution vector $U^{*,m}$ for the m-th window by $B_r(U^{*,m})$ and $B_R(U^{*,m})$, respectively. Another key ingredient in this analysis is given by the shifting process, which we represent through the nonlinear mapping

$$\Xi : \mathbb{R}^{d(N+1)} \to \mathbb{R}^{d(N+1)}$$

such that

$$\Xi(U^{K,m}) = \Xi\left(\begin{bmatrix} U_0^{K,m} \\ \vdots \\ U_{N-\Delta N}^{K,m} \\ U_{N-\Delta N+1}^{K,m} \\ \vdots \\ U_N^{K,m} \end{bmatrix}\right) = \begin{bmatrix} U_{\Delta N}^{K,m} \\ \vdots \\ U_N^{K,m} \\ G(U_{N-\Delta N}^{0,m+1}) \\ \vdots \\ G(U_{N-1}^{0,m+1}) \end{bmatrix} = \begin{bmatrix} U_0^{0,m+1} \\ \vdots \\ U_{N-\Delta N}^{0,m+1} \\ U_{N-\Delta N+1}^{0,m+1} \\ \vdots \\ U_N^{0,m+1} \end{bmatrix} = U^{0,m+1}. \tag{4.9}$$

In other words, Ξ saves the last $N-\Delta N+1$ components of $U^{K,m}$ as the first $N-\Delta N+1$ components of $U^{0,m+1}$, and computes the remaining entries of $U^{0,m+1}$ with the coarse solver G applied recursively starting from $U_{N-\Delta N}^{0,m+1} \equiv U_N^{K,m}$.

Using this notation, our requirement for error control translates to ensuring that

$$\Xi\big(B_r(U^{*,m})\big) \subseteq B_R(U^{*,m+1}), \quad \forall m \geq 1, \tag{4.10}$$

i.e. that every point in the inner ball $B_r(U^{*,m})$ of window $\mathcal{T}_m$ is mapped under Ξ to a point within the outer ball $B_R(U^{*,m+1})$ of window $\mathcal{T}_{m+1}$. A schematic of this interpretation is shown in Figure 4.2.

The key question is: How big can we make ΔN for a given K and ΔT? That is, how many more time chunks can we compute using the coarse solver when shifting from one window to the next, without pushing the initial guess outside the outer ball? The proximity measure ψ, developed in Section 3.4, plays a crucial role in answering this question. We remind the reader that

$$\psi(U) = \frac{1}{N}\sum_{n=1}^{N} w^{-n}\|U_n - F(U_{n-1})\|. \tag{4.11}$$

Uniform proximity of the initialization over a window to the window's fine solution can only be guaranteed for very short time chunks, meaning very small ΔT. By contrast, using a weighted proximity measure provides more flexibility. It ensures that only the relevant parts of the initialization remain close to the fine solution, while allowing for larger shifts.

Before proceeding, we must formalize the notion of the one-chunk error introduced by the coarse solver. We assume that the system has a compact global attractor $\mathcal{A}$, in the sense of Definition 2.4.2, which we established as a common feature of dissipative chaotic systems. Since the fine and coarse propagators, F and G, are continuous functions, by the extreme-value theorem the difference $\|G(\cdot) - F(\cdot)\|$ will be bounded on this compact set. This allows us to define the maximum possible one-chunk error.

Definition 4.3.1. Let $\mathcal{A}$ be the compact global attractor of the dynamical system. We define γ as the maximum one-chunk error between the coarse and fine solvers over all points on the attractor and for all time chunks in a window:

$$\gamma := \max_{U\in\mathcal{A}} \max_{n\in\{1,\dots,N\}} \|G(U_n) - F(U_n)\|. \tag{4.12}$$

Also, we define ΔN as the maximum number of time chunks that can be shifted forwards while guaranteeing that (4.10) holds. Note that ΔN depends on K through R. In what follows, we derive an estimate for ΔN. To simplify our analysis, we keep the total number N of time chunks constant across all window.

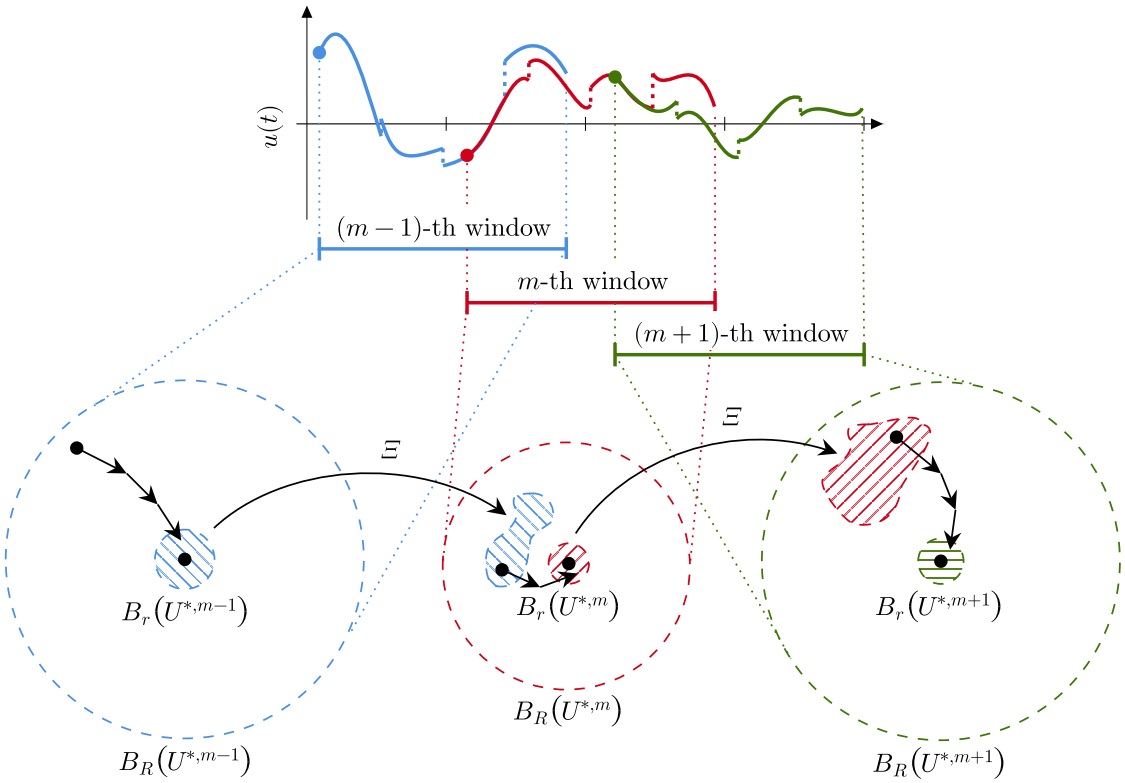

Figure 4.2: A schematic of MoWi's tracking analysis framework. MoWi's stability is guaranteed if the shifting map, Ξ, projects the converged solution (within the inner ball B_r) of one window into the basin of attraction (the outer ball B_R) of the next. This ensures the Parareal subroutine in each window converges within its iteration budget.

Theorem 4.3.2. Let F satisfy a Lipschitz condition on $\mathbb{R}^d$ (in the sense of Definition 2.2.2) in some norm $\|\cdot\|$ with constant Λ_F. To ensure the tracking condition (4.10) holds, we choose a base weight $w > \max\{1, \Lambda_F\}$ for the proximity function ψ. Let γ be the maximum one-chunk error of the coarse solver, as defined in Definition 4.3.1. Then

$$\Delta N = \left\lfloor \log_w \left(R + \gamma \frac{w^{-N}}{w-1} \right) - \log_w \left(r + \gamma \frac{w^{-N}}{w-1} \right) \right\rfloor. \tag{4.13}$$

Proof. Our goal is to show that $\Xi\left(B_r(U^{*,m-1})\right) \subseteq B_R(U^{*,m})$. This is equivalent to claiming that an initial state within the inner ball of the previous window, meaning

$$\|U^{K,m-1} - U^{*,m-1}\|_W \leq r, \tag{4.14}$$

leads to an initial guess for the new window that is within the outer ball, meaning

$$\|U^{0,m} - U^{*,m}\|_W \leq R. \tag{4.15}$$

From Theorem 3.4.1, we know that the proximity function ψ provides upper and lower bounds on the true error norm. Therefore, we can work with ψ directly. We will show that if the run-time checkable criterion

$$\psi(U^{K,m-1}) \leq \frac{r}{C_N} \tag{4.16}$$

is met, then it is guaranteed that

$$\psi(U^{0,m}) \leq \frac{R}{C_N}, \tag{4.17}$$

which in turn ensures the solution remains within the required outer ball.

So, due to Theorem 3.4.1,

$$\|U^{K,m-1} - U^{*,m-1}\|_W \leq \sum_{n=1}^{N} w^{-n} \|U_n^{K,m-1} - U_n^{*,m-1}\| = N\psi^*(U^{K,m-1}), \tag{4.18}$$

$$\|U^{0,m} - U^{*,m}\|_W \leq \sum_{n=1}^{N} w^{-n} \|U_n^{0,m} - U_n^{*,m}\| = N\psi^*(U^{0,m}). \tag{4.19}$$

By (3.123), (4.16) implies

$$N\psi^*(U^{K,m-1}) \leq C_N \psi(U^{K,m-1}) \leq r. \tag{4.20}$$

(4.18) shows that (4.16) is a sufficient criterion for (4.14). We use this to show that

$$N\psi^*(U^{0,m}) \leq R, \tag{4.21}$$

which by (4.19) implies (4.15). To do so, it suffices to show (4.17). In particular, we show

$$N\psi(U^{0,m}) \leq \frac{RN}{C_N}. \tag{4.22}$$

We start with the left-hand side and split it into the part inherited from the previous window ($n = 1, \ldots, N - \Delta N$) and the part computed by the coarse solver ($n = N - \Delta N + 1, \ldots, N$):

$$N\psi(U^{0,m}) = \sum_{n=1}^{N-\Delta N} w^{-n}\|U_n^{0,m} - F(U_{n-1}^{0,m})\| + \sum_{n=N-\Delta N+1}^{N} w^{-n}\|U_n^{0,m} - F(U_{n-1}^{0,m})\|. \tag{4.23}$$

For the first part,

$$\sum_{n=1}^{N-\Delta N} w^{-n}\|U_n^{0,m} - F(U_{n-1}^{0,m})\| \leq w^{\Delta N} N\psi(U^{K,m-1}) \tag{4.24}$$

$$\leq w^{\Delta N}\frac{Nr}{C_N}, \tag{4.25}$$

where the last step is due to (4.16). For the second part,

$$\sum_{n=N-\Delta N+1}^{N} w^{-n}\|U_n^{0,m} - F(U_{n-1}^{0,m})\| = \sum_{n=N-\Delta N+1}^{N} w^{-n}\|G(U_{n-1}^{0,m}) - F(U_{n-1}^{0,m})\| \tag{4.26}$$

$$\leq \gamma \sum_{n=N-\Delta N+1}^{N} w^{-n} \tag{4.27}$$

$$\leq \gamma w^{-N}\frac{w^{\Delta N} - 1}{w - 1}. \tag{4.28}$$

Putting (4.25) and (4.28) together with (4.23),

$$N\psi(U^{0,m}) \leq w^{\Delta N}\frac{Nr}{C_N} + \gamma w^{-N}\frac{w^{\Delta N} - 1}{w - 1}. \tag{4.29}$$

The last step is to make sure that (4.22) is satisfied. To do so, we check

$$w^{\Delta N}\frac{Nr}{C_N} + \gamma w^{-N}\frac{w^{\Delta N} - 1}{w - 1} \leq \frac{RN}{C_N}. \tag{4.30}$$

Because we choose $w > \max\{1, \Lambda_F\}$, then $C_N = N$ – compare with Theorem 3.4.1. Therefore, our requirement becomes

$$w^{\Delta N} \leq \frac{R + \gamma\frac{w^{-N}}{w-1}}{r + \gamma\frac{w^{-N}}{w-1}}, \tag{4.31}$$

which holds true when ΔN is the solution of (4.13). □

Equation (4.13) has an interesting physical interpretation. This can be readily seen by plugging in $w = \exp(\lambda \Delta T)$, where λ is the MLE of the system. In fact, then (4.13) can be recast as

$$\Delta\tau = \Delta N \Delta T = \frac{1}{\lambda} \left\lfloor \log\left(R + \gamma \frac{e^{-N\lambda\Delta T}}{e^{\lambda\Delta T} - 1}\right) - \log\left(r + \gamma \frac{e^{-N\lambda\Delta T}}{e^{\lambda\Delta T} - 1}\right) \right\rfloor . \tag{4.32}$$

Even better,

$$\Delta\tau \leq \frac{1}{\lambda} \log\left(\frac{R}{r}\right). \tag{4.33}$$

As explained in Section 2.3, the MLE is dimensionally equivalent to a frequency, and its inverse identifies the turbulent timescale. From this viewpoint, (4.32) can be interpreted as

$$\text{shift allowed} = \text{turbulent timescale} \times \text{multiplicative factor}.$$

In other words, (4.32) tells us how many turbulent timescales $1/\lambda$ we can loosely predict in the future without losing too much of the information propagated by the time-parallel algorithm, which in turn is encoded into the multiplicative factor. This helps us complete the picture by establishing how the iterates converge from the outer to the inner ball within each window. We can simply plug in our naïve bound for R from Section 3.2, which we remind the reader to be

$$R \leq \left(\frac{r}{\beta^{S_K(q)}}\right)^{\frac{1}{q^K}}, \tag{3.12}$$

to get

$$\Delta\tau \leq \frac{S_K(q)}{\lambda q^K} \log\left(\frac{1}{\beta r^{q-1}}\right), \quad S_K(q) = \sum_{i=0}^{K-1} q^i. \tag{4.34}$$

This simple trick allows us to qualitatively examine the bound for various convergence rates. As shown in Figure 4.3, for superlinearly convergent time-parallel algorithms, i.e. $q > 1$, the multiplicative factor is limited mainly by r, which is typically much smaller than β. In fact, Figure 4.3 suggests that the smaller the radius r reached within K iterations, the better. Also, the higher the order of convergence q is, the looser the limit to the multiplicative factor becomes. The situation changes dramatically when $q = 1$: for linearly convergent time-parallel algorithms (like Parareal),

$$\Delta\tau \leq \frac{K}{\lambda} \log\left(\frac{1}{\beta}\right), \tag{4.35}$$

which is independent of r. This expression provides a powerful insight: it links the algorithmic parameters of Parareal (K, β) directly to the shift of MoWi. To make this

bound fully explicit, we can substitute our estimate for $\beta = \mathcal{O}(h^2)$ from Chapter 3 to quantify the shift as a function of the fine step size h. Overall, this qualitative analysis suggests that the faster the time-parallel subroutine is in transferring information over the time domain, the further in time we can push our shifting.

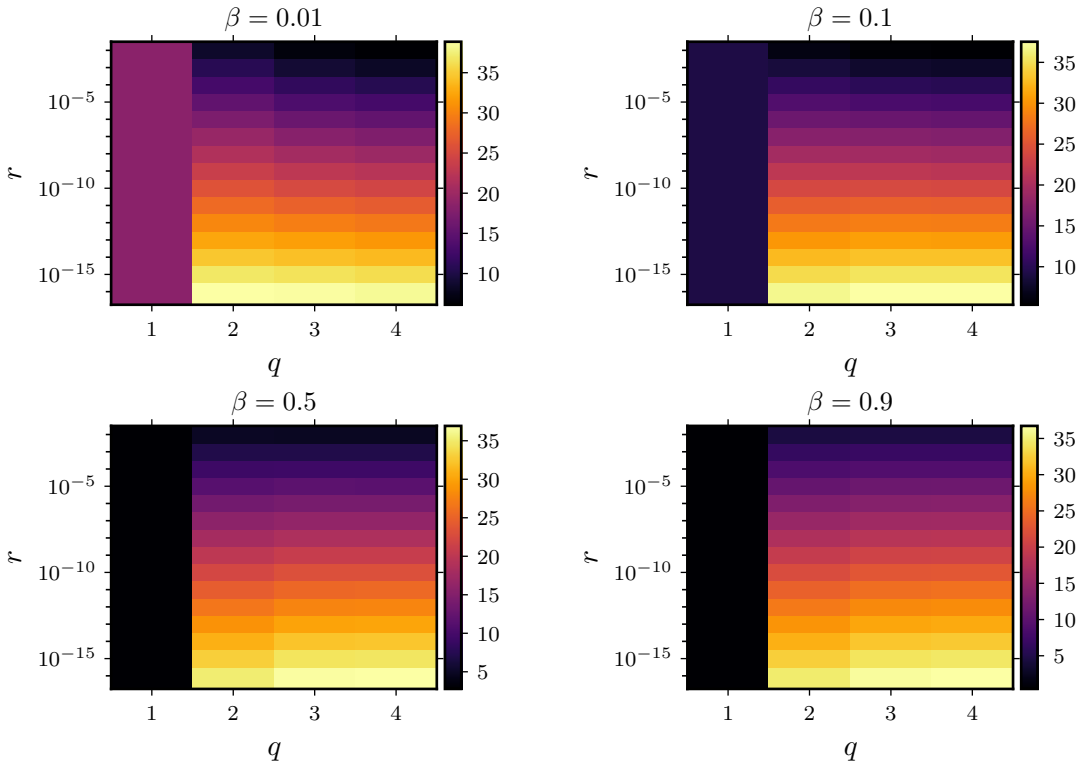

Figure 4.3: Plot of the theoretical bound (4.34) on the MoWi shift as a function of β, q, r. Notice that for $q = 1$, than the bound loses its dependence on r. For $q > 1$, however, we get higher values, and thus possibly a larger shift allowed, as we lower r, i.e. as we force our tolerance on the parallel solution within each window to be stricter, and as we lower β, i.e. as the convergence rate of the time-parallel algorithm employed increases.

4.3.1 Experiments

When moving from one window to the next, we check for $\psi(U^0) \leq R$ and $\psi(U^K) \leq r$. In other words, we measure convergence with $\psi(U^k)$ rather than $\|U^k - U^*\|_W$.

The outcomes of our simulations are shown in Figures 4.5 to 4.8. They are all based on the Lorenz equations. In Figure 4.5, we use an RK4 solver for both fine and coarse solvers, with time-step ratio $\xi = 10$. We run MoWi for different values of the outer radius R. As we lower R, and thus as the requirement for the shift to a new window to be successful becomes stricter, there is a transition between independence from the chaoticity of the problem, i.e. $\Delta N = N$, to a situation where our bound becomes descriptive. In Figure 4.6, we run the same simulations with a worse coarse solver, the midpoint method. As expected, the situation worsens, that is, dependence on how much chaos is present at all levels.

However, we do not have access to R, so this is not realistic. More realistically, we want to set the minimum speedup S we want to achieve within each window. Doing so, we can find the corresponding K via one of the estimates for the parallel efficiency in Section 2.7.2. Using the naïve estimate (2.61) for the parallel efficiency of Parareal, we get

$$K = \left\lfloor \frac{N}{S(1+\zeta N)} \right\rfloor, \tag{4.36}$$

whereas, using the improved estimate (2.65), we get

$$K = \left\lfloor N + \frac{1}{2} - \sqrt{\left(N + \frac{1}{2}\right)^2 - \frac{2N^2}{S(1+\zeta N)}} \right\rfloor, \quad S \geq \frac{2N^2}{(1+\zeta N)\left(N+\frac{1}{2}\right)^2}. \tag{4.37}$$

We can then use

$$R \leq \frac{r}{\beta^K}. \tag{3.13}$$

Section 4.3.1 gives an idea of the magnitudes involved.

It is important to distinguish between the initialisation of the algorithm and its subsequent operation. The tracking analysis and the strict K-iteration budget apply to the windows once the simulation is "on track" (that is, once the solution is evolving on or near the system's attractor). The very first window may require more than K iterations to handle this initial transient phase. This is treated as a one-time initialisation cost which, for the long time domains that MoWi is designed for, is amortised over the many efficient windows that follow.

We run the same simulations we have run for Figures 4.5 and 4.6, but fixing the speedup S instead. We show the outcomes in Figures 4.7 and 4.8, respectively. The

N	ξ	S	K
10	10	2	2
		3	1
		4	1
	100	2	6
		3	3
		4	2
100	10	2	4
		3	3
		4	2
	100	2	29
		3	18
		4	13

Table 4.4: Example iteration budgets for Parareal, calculated using the improved efficiency estimate from (2.65). The table shows how K varies for different numbers of time chunks (N), coarse-to-fine step ratios (ξ), and target parallel speedups (S).

same comments we made before apply in this case too, but the notable difference that R changes throughout the runs.

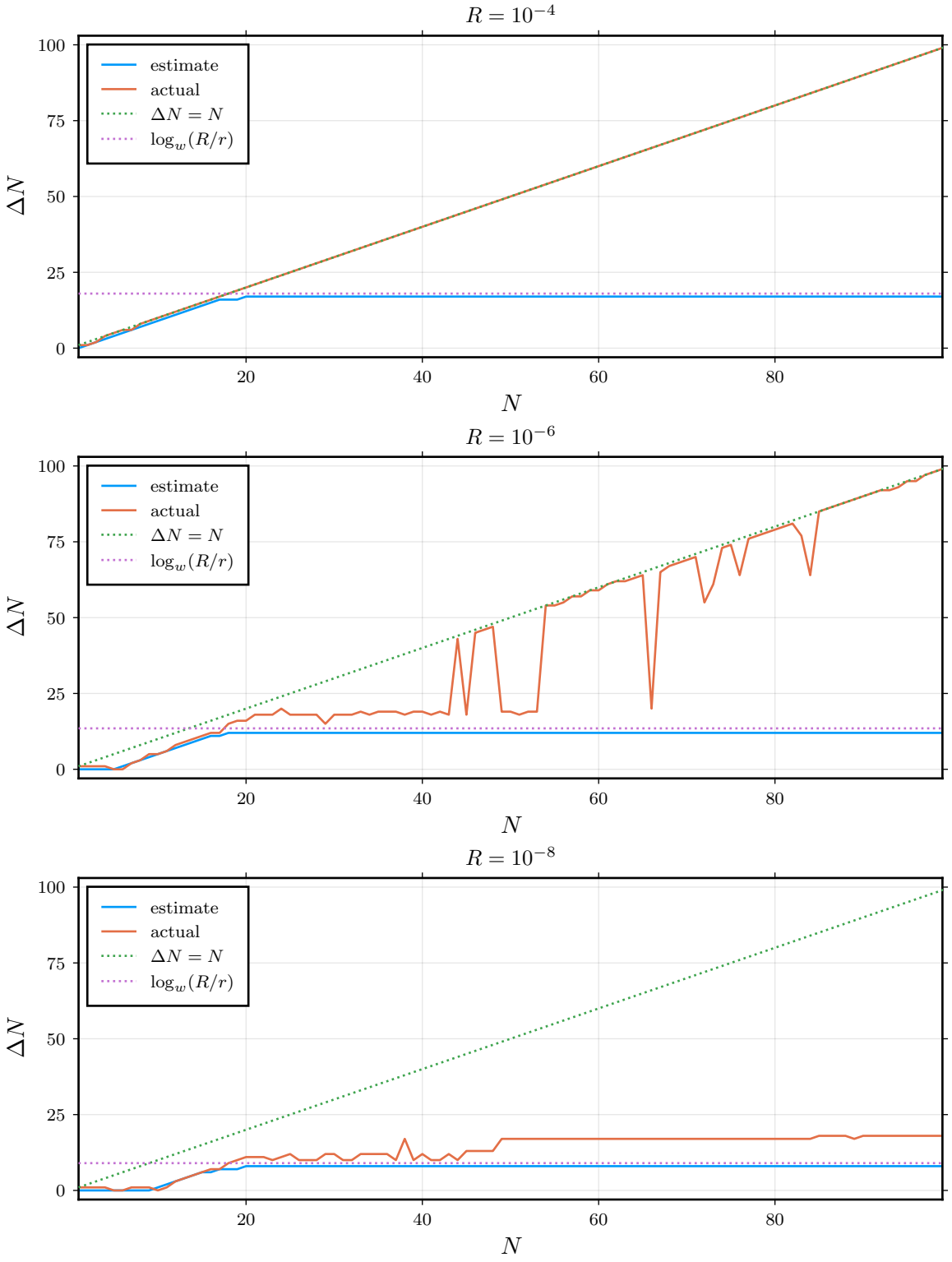

Figure 4.5: The maximum allowed forward shift, ΔN, as a function of the number of chunks N, using RK4 for both G and F, with $\xi = 10$. The different subplots correspond to stricter convergence requirements (smaller outer radius R). For loose requirements (top, $R = 10^{-4}$), the shift can be maximal ($\Delta N = N$). As the requirement becomes stricter (bottom, $R = 10^{-8}$), the maximum shift becomes constrained and follows the theoretical bound (dotted line).

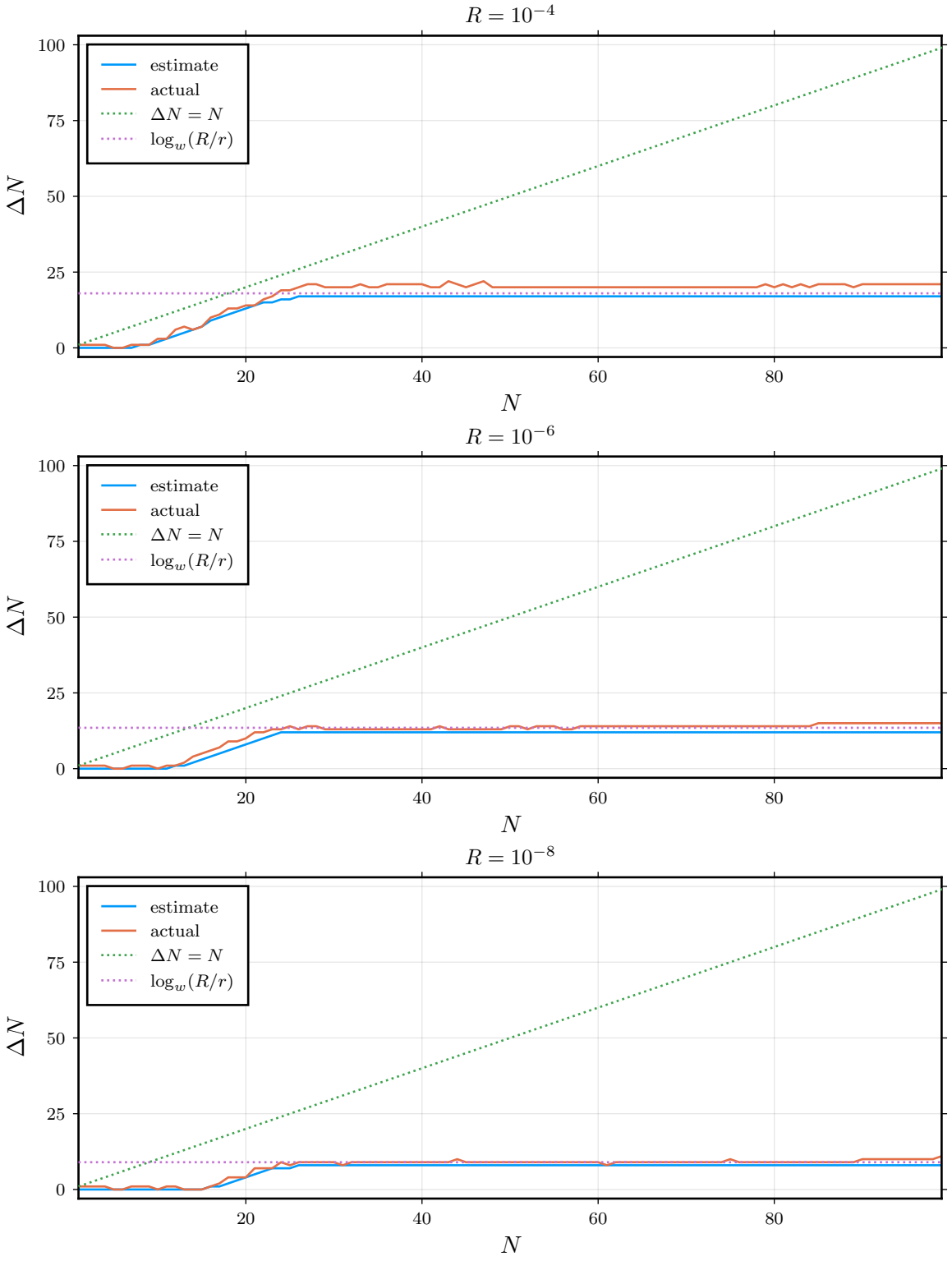

Figure 4.6: The maximum allowed forward shift, ΔN, as a function of the number of chunks N, using the Midpoint method for both G and F, with $\xi = 10$. As in Figure 4.5, stricter convergence requirements (smaller R) constrain the maximum allowed shift, forcing it to follow the theoretical bound derived from our tracking analysis.

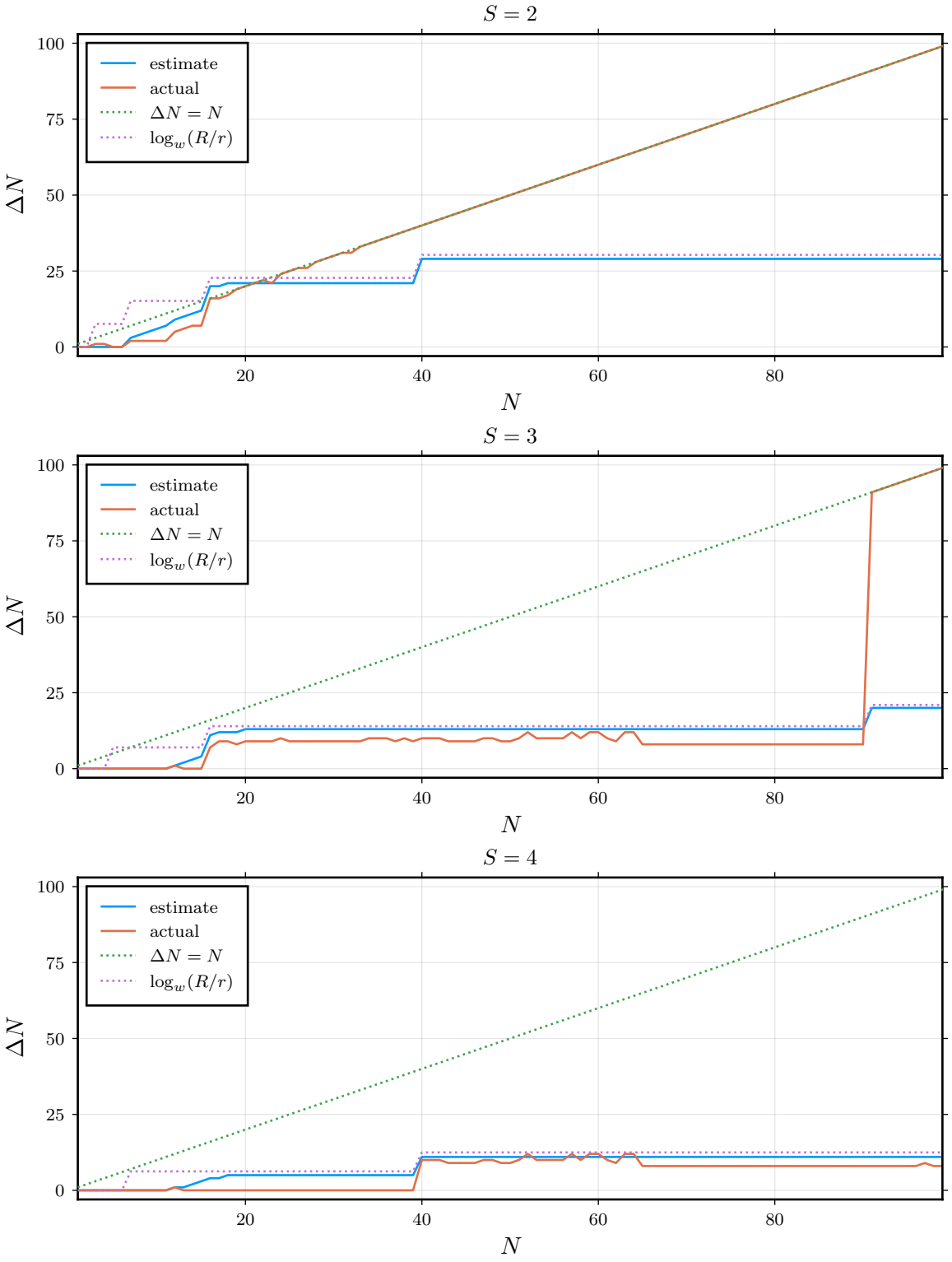

Figure 4.7: The maximum allowed forward shift, ΔN, for a fixed target speedup S, using RK4 for both G and F, with $\xi = 10$. A higher speedup target requires a smaller iteration budget K, which implies a stricter convergence condition (a smaller effective R). The subplots show that as the desired speedup increases from $S = 2$ (top) to $S = 4$ (bottom), the maximum allowed shift becomes more constrained.

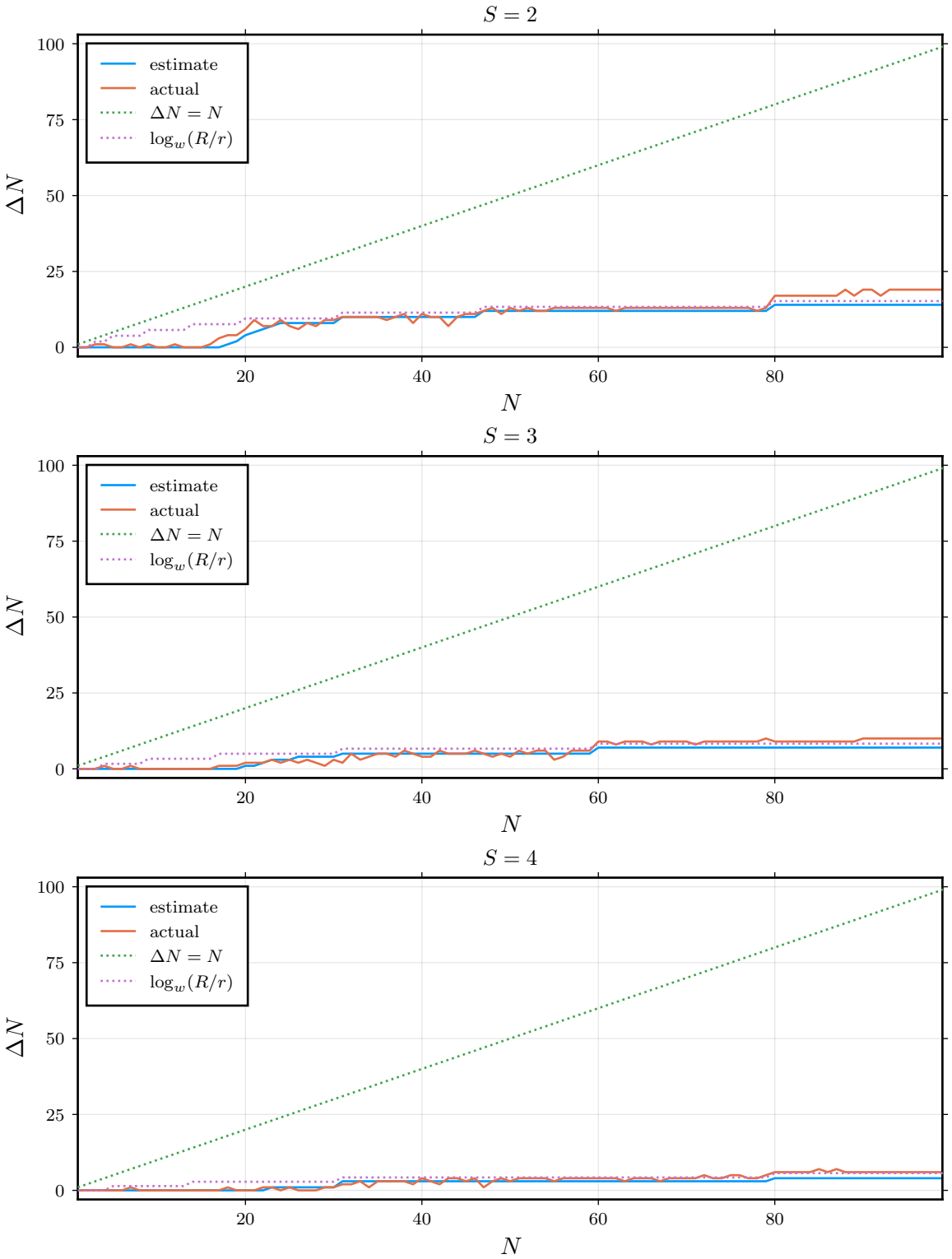

Figure 4.8: The maximum allowed forward shift, ΔN, for a fixed target speedup S, using Midpoint for both G and F, with $\xi = 10$. The results are consistent with Figure 4.7, confirming that requiring a higher parallel speedup (and thus faster convergence per window) reduces the maximum possible shift between windows.

4.4 Cost

By reusing parts of the solutions found at the preceding time windows, the time-parallel subroutine starts from an initial guess that is ideally closer to the fine solution of each time window, to speed up convergence. In this section, we check this by comparing the performance of MoWi (with Parareal as the employed subroutine) with that of Parareal, to establish under what conditions it is beneficial to switch to MoWi. We develop our analysis step by step, to emphasize what parameters each task brings into the picture.

Let τ be the time-window size and K_m be the number of iterations that Parareal takes to converge at the m-th time window. We begin our analysis by neglecting all coarse computations. Under the same assumptions of Section 2.7.2, the computational cost of MoWi writes as

$$C_{\mathrm{M}} = C_F \frac{\tau}{Nh} \sum_{m=1}^{M} K_m. \tag{4.38}$$

Comparing this formula with that of Parareal, we get

$$\frac{C_{\mathrm{M}}}{C_{\mathrm{P}}} = \frac{\tau}{T} \sum_{m=1}^{M} \frac{K_m}{K}. \tag{4.39}$$

By construction, $T = \tau + (M-1)\Delta\tau$. In practice, $T \gg \tau$, so $M \gg 1$. Qualitatively, this means that $T \approx M\Delta\tau$, such that

$$\frac{C_{\mathrm{M}}}{C_{\mathrm{P}}} \approx \frac{\kappa\tau}{K\Delta\tau}, \tag{4.40}$$

where κ denotes the average number of Parareal iterations across all time windows, that is

$$\kappa = \frac{1}{M} \sum_{m=1}^{M} K_m. \tag{4.41}$$

For MoWi to be more efficient than standard Parareal ($C_{\mathrm{M}} \ll C_{\mathrm{P}}$). Equation (4.40) shows two conditions must be met. First, the average per-window iteration count must be significantly smaller than the prohibitively large number of iterations standard Parareal would require over the full domain (that is, $\kappa \ll K$). Second, the windowing must be efficient, with a shift that is close to the window size (that is, $\Delta\tau \lesssim \tau$). In other words, we want Parareal to converge faster (on average) as we decrease the length of the time windows, which seems plausible given the results displayed in Figure 3.10, and at the same time we want to shift forward as much as possible, to be able to cover the full length of the time domain more quickly. This reveals the

central trade-off of MoWi: for the method to be efficient, the benefit of achieving a low average iteration count κ in small windows must outweigh the computational overhead of using a small forward shift $\Delta\tau$ relative to the window size τ.

Next, we take the coarse computations into account, which leads us to

$$C_{\mathrm{M}} = C_F \frac{\tau}{Nh}(1+\zeta N)\sum_{m=1}^{M} K_m + C_G(M-1)\frac{\Delta\tau}{\xi h} \tag{4.42}$$

$$= C_F \frac{\tau}{Nh}(1+\zeta N)\left(\sum_{m=1}^{M} K_m + \frac{\zeta N}{1+\zeta N}(M-1)\frac{\Delta\tau}{\tau}\right), \tag{4.43}$$

such that

$$\frac{C_{\mathrm{M}}}{C_{\mathrm{P}}} = \frac{\tau}{KT}\left(\sum_{m=1}^{M} K_m + \frac{\zeta N}{1+\zeta N}(M-1)\frac{\Delta\tau}{\tau}\right). \tag{4.44}$$

Following the same qualitative argument as above,

$$\frac{C_{\mathrm{M}}}{C_{\mathrm{P}}} \approx \frac{1}{K}\left(\frac{\kappa\tau}{\Delta\tau} + \frac{\zeta N}{1+\zeta N}\right). \tag{4.45}$$

The presence of an additional term in (4.45) compared to (4.40) is mostly due to the shifting procedure. To minimize its effect, (4.45) suggests that the selected coarse solver should be as cheap to apply as possible, especially as we increase the number of initial conditions that we have to compute in a time window.

Lastly, we consider the slightly optimized implementation of Parareal considered in Section 2.7.2, in which we skip the computations relative to the time chunks that have unquestionably converged. Repeating the same calculations as above, we obtain

$$C_{\mathrm{M}} = C_F \frac{\tau}{Nh}(1+\zeta N)\left[\sum_{m=1}^{M} K_m\left(1-\frac{K_m-1}{2N}\right) + \frac{\zeta N}{1+\zeta N}(M-1)\frac{\Delta\tau}{\tau}\right], \tag{4.46}$$

such that

$$\frac{C_{\mathrm{M}}}{C_{\mathrm{P}}} = \frac{\tau}{T}\frac{\sum_{m=1}^{M} K_m\left(1-\frac{K_m-1}{2N}\right) + \frac{\zeta N}{1+\zeta N}(M-1)\frac{\Delta\tau}{\tau}}{K\left(1-\frac{K-1}{2N}\right)}, \tag{4.47}$$

which qualitatively boils down to

$$\frac{C_{\mathrm{M}}}{C_{\mathrm{P}}} \approx \frac{\frac{\kappa\tau}{\Delta\tau}\left(1-\frac{\kappa-1}{2N}\right) + \frac{\zeta N}{1+\zeta N}}{K\left(1-\frac{K-1}{2N}\right)}. \tag{4.48}$$

Although there is a slight improvement, the same comments made about (4.45) apply here too.

4.4.1 Experiments

For the experiments in this section, we fix the number of chunks per window at $N = 100$. The cost analysis considers the amortised performance of the algorithm once it is 'on track'; therefore, the potentially higher computational cost of the first window ($m = 0$) is treated as a one-time initialisation expense and is excluded from the calculation of the cost ratio C_M/C_P. The estimate curves in the plots are based on our theoretical cost formulas. To use these formulas, a value for K (the iteration count for a full-domain Parareal run) is required. As K is unknown for truly long problems, we use an empirically measured value of K from a shorter reference simulation to allow for a direct comparison.

The discussion above is valid when considering parallel runtime. We show an example simulation in Figures 4.9 and 4.10. Its limit is the absence of any term that takes into account the time spent communicating data among the processors. This, of course, depends on the design we have chosen for the parallel implementation of the algorithm. In particular, we use a master-worker approach using the tools provided by Julia.

The results in Figure 4.9 require careful interpretation. At a first glance, as the shift $\Delta\tau \to 0$, the MoWi algorithm becomes equivalent to a standard Parareal solve, suggesting the cost ratio C_M/C_P should approach 1. However, our plots show the ratio becoming very large in this limit. This is a practical artefact of the implementation: for a very small forward shift, the algorithm incurs the overhead of initialising a new window for very little progress through the time domain. Furthermore, the standard convergence check forces the solver to take many iterations to resolve the discontinuities in each window, causing the poor performance seen across the plot and confirming that MoWi is ineffective without a chaos-aware stopping criterion like ψ.

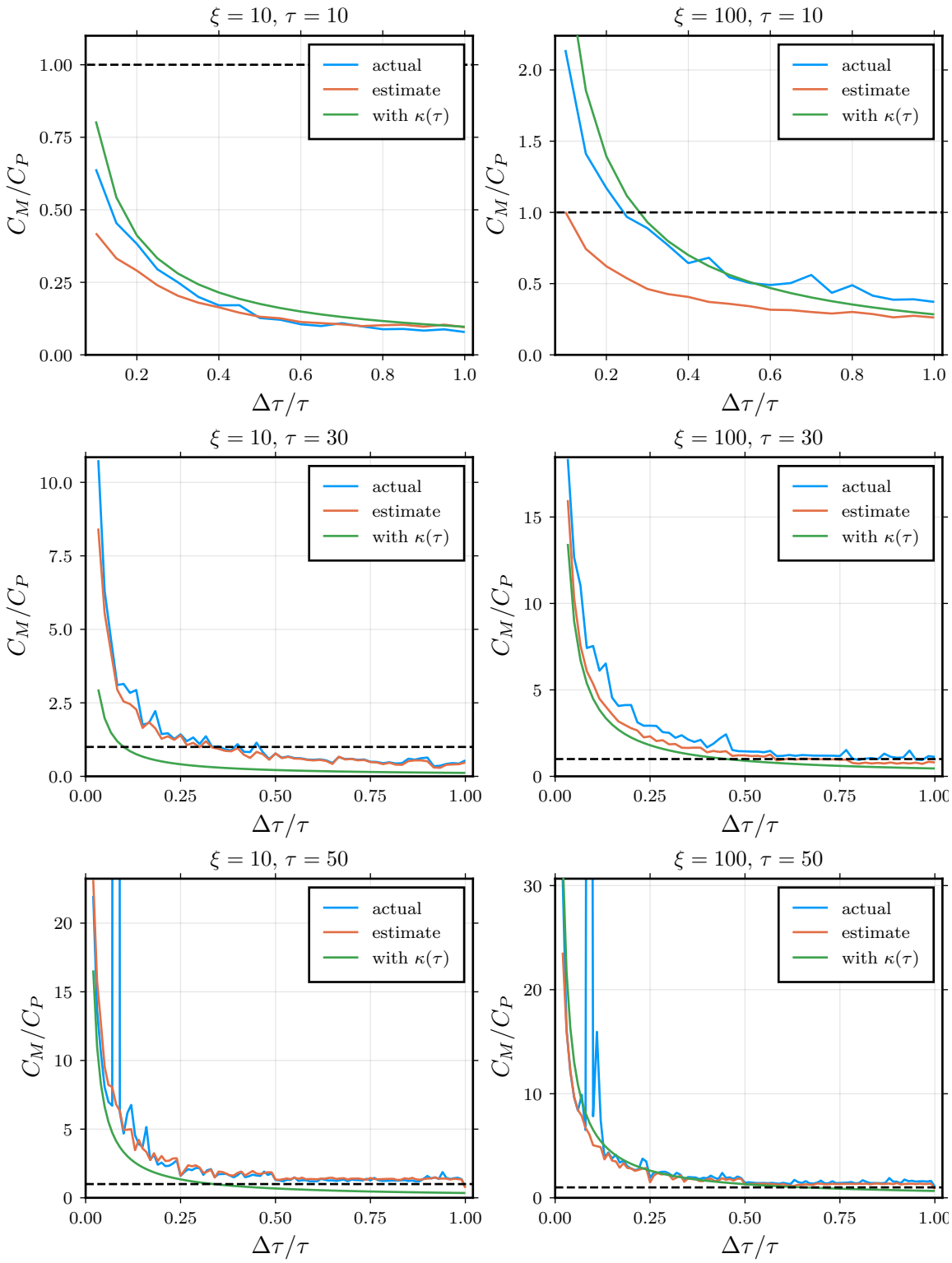

Figure 4.9: Cost ratio (C_M/C_P) of MoWi versus standard Parareal when using a standard convergence check, for Lorenz, using RK4 for both G and F. A ratio above the dashed line ($C_M/C_P > 1$) means MoWi is more expensive. The different panels show varying window sizes (τ) and coarse/fine step ratios (ξ). The results show that without the proximity function, MoWi rarely provides a cost benefit.

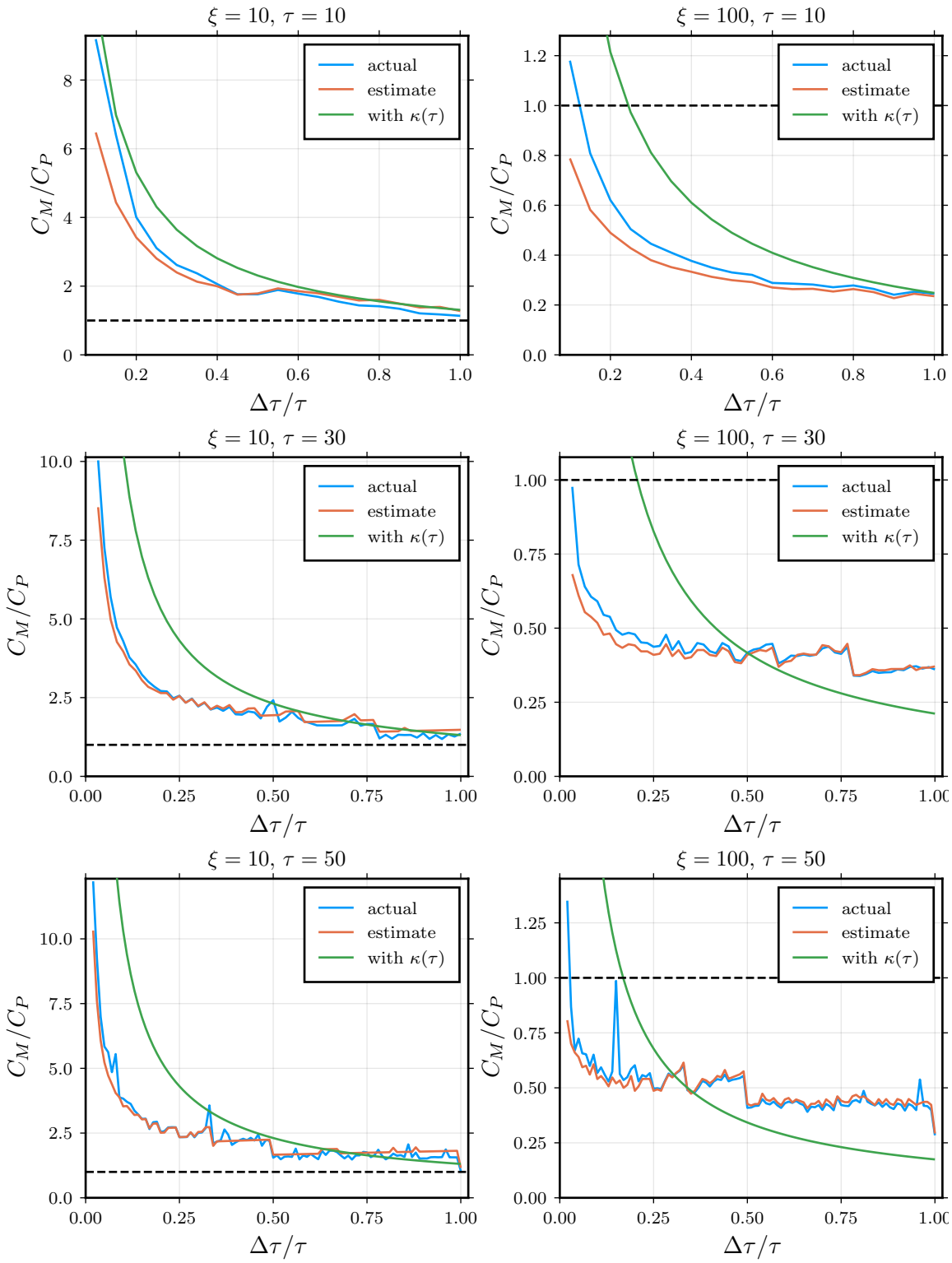

Figure 4.10: Cost ratio (C_M/C_P) of MoWi versus standard Parareal when using the weighted proximity function ψ, for Lorenz, using RK4 for both G and F. In contrast to Figure 4.9, the cost ratio consistently drops below 1, demonstrating that MoWi is more efficient than a direct Parareal implementation. The benefit is greatest for large shifts ($\Delta\tau/\tau \to 1$) and when the coarse solver is relatively cheap (right column, $\xi = 100$).

Chapter 5

Adaptive MoWi

In Chapter 4, we introduced MoWi as a way to combine multiple time windows to achieve efficient time-parallel integration of chaotic dynamical systems over long time domains. MoWi leverages time parallelization as a subroutine to speed up the resolution of these systems. However, in its standard form, MoWi can be too rigid when the system's dynamics vary significantly over time, i.e. across different regions of the phase space. In such cases, it may fail to fully exploit the potential speedup available. To address this, we present three adaptive strategies aimed at making MoWi more flexible and responsive to these changes. Each strategy adjusts relevant parameters on the fly during computations. We refer to this enhanced version of MoWi as adaptive MoWi (AMoWi).

The adaptive logic operates in two phases. First, during a window solve, if the time-parallel subroutine (Parareal in this thesis) fails to converge within its iteration budget, the algorithm restarts the computation on the same window but with more conservative parameters (for example, a shorter window or a stricter proximity weight). To avoid wasting work, the unconverged solution is used as an improved initial guess for the restart. Second, after a successful window solve, the algorithm prepares for the next window by adjusting its parameters to be more optimistic (for example, a longer window or a larger shift), probing for potential speedups in regions of lower chaoticity.

All strategies aim to speed up MoWi while staying within a fixed computational budget. When shifting between windows, we make sure that each window's initial guess lies well within its outer ball. This outer ball must be so that the time-parallel subroutine reaches the inner ball within a limited number of iterations. As we have seen in Chapter 4, the radius of the outer ball depends on the convergence speed of the time-parallel subroutine, its convergence criterion, and ultimately on the proximity function ψ. Therefore, the outer ball is determined by the convergence and cost analysis of the time-parallel algorithm. For simplicity, we fix the rate of adaptive

change and aim for small-time solutions that converge within each window according to the convergence criterion of the time-parallel subroutine.

We now present each adaptive strategy individually. Due to their differing mechanisms, a direct comparison of their mechanisms is difficult. Instead, we evaluate them based on their performance – specifically, the length of the time domain they can integrate within a fixed computational budget, such as wall-clock time.

5.1 AMoWi-1: Adapting the window length

In AMoWi-1, we adaptively increase (*stretching*) and decrease (*shrinking*) the window length whilst keeping the number of steps and the ratio between the fine and coarse time-step sizes constant. If convergence succeeds, we extend the window length for the next window, whilst keeping the number of coarse steps constant.

Following the notation introduced in Chapter 4, let τ be the window length and $\Delta\tau$ be the shift length. AMoWi-1 adjusts τ through stretching or shrinking. Since we keep the number of coarse steps fixed, the coarse step-size h_G must change accordingly. As a consequence,$\Delta\tau$ also changes. What about h_F, that is, h? Changing the fine-solver step-size can alter the underlying physics of the problem, particularly in systems like Navier-Stokes where changing time-scales affects the dynamics. To avoid this, we fix the fine step-size and do not modify it.

Also, sometimes we may want to shrink more aggressively than we stretch to remain cautious. Other times we might want to do the opposite, depending on the context. We will determine the most effective strategy through numerical experiments. In any case, to distinguish these two operations, we introduce separate parameters: δ^+ for stretching (increasing τ) and δ^- for shrinking (decreasing τ).

5.1.1 Analysis

A fully rigorous proof of convergence for the implemented AMoWi-1 is challenging because the strategy modifies only the coarse grid, which breaks the assumptions of our framework from Chapter 3. However, we can gain valuable insight by analysing a simplified, idealised model where both coarse and fine step sizes are shrunk proportionally. We also assume that each restart begins from a fresh coarse solve, rather than the previous unconverged state. While these assumptions differ from our practical implementation, they allow for a tractable analysis that provides theoretical backing for the strategy's core mechanism.

In what follows, we define $\delta > 1$ and set $\delta^+ = \delta$ for stretching and $\delta^- = 1/\delta$ for shrinking. In our implementation, we only adjust the coarse step-size, not the fine step-size. Therefore, during shrinking,

$$h_{G,r} = \delta^- h_{G,r-1} = \frac{h_{G,r-1}}{\delta}, \tag{5.1}$$

whilst $h_F = h$ stays fixed. As mentioned, however, for the purpose of this analysis, we make the simplifying assumption that

$$h_{F,r} = \delta^- h_{F,r-1} = \frac{h_{F,r-1}}{\delta}. \tag{5.2}$$

This simplification allows us to apply results from Chapter 3 and analyse the behaviour of the adaptive strategy more easily. After a successful window solve, we shift forwards in time by $\Delta\tau$ and increase the window length. Note that, although we do not directly act on it, $\Delta\tau$ changes proportionally due to adjustments in h_G and h.

Theoretical guarantees for AMoWi-1 with Parareal are summarized in Theorem 5.1.1. By shrinking the window by a fixed factor δ, we can ensure convergence within a certain number of restarts ι. This analysis shows that it is possible to guarantee that every window eventually converges within the inner ball, so that MoWi stays on track.

Theorem 5.1.1. Consider AMoWi-1 with Parareal as a subroutine. Each window is guaranteed to converge (and lead to a window shift) after at most

$$\iota = \left\lceil \frac{1}{2K} \log_\delta \left(\frac{C_N}{cr} \psi(U^0) \right) + \frac{\log_\delta (Ch^2)}{2} \right\rceil \tag{5.3}$$

restarts.

Proof. From Theorem 3.4.1 and (3.9), noting that Parareal converges linearly ($q = 1$) so that $S_K(1) = K$, we have

$$\psi(U^K) \leq \frac{C_N}{c} \beta^K \psi(U^0). \tag{5.4}$$

According to Theorem 3.3.2, $\beta \leq Ch^2$, so we get

$$\psi(U^K) \leq \frac{C_N C^K}{c} h^{2K} \psi(U^0). \tag{5.5}$$

By construction, after each restart, we shrink the window and all discretization step sizes (including the fine step size) by a factor of δ. Hence, after ι restarts, the fine step size is

$$h_F = \frac{h}{\delta^\iota}, \tag{5.6}$$

Substituting h with h/δ^ι into (5.5), we get

$$\psi(U^K) \leq \frac{C_N C^K}{c} \left(\frac{h}{\delta^\iota}\right)^{2K} \psi(U^0). \tag{5.7}$$

For a window to converge, we require $\psi(U^K) \leq r$, where r is the preset tolerance. Hence, we bound instead (5.7). Solving the resulting inequality, which is satisfied after at most ι restarts, for ι yields (5.3). □

Whilst this result relies on a simplified model, it provides theoretical backing for the core idea: shrinking the time window (and thus the effective step size) is a guaranteed way to induce convergence. The number of restarts required scales logarithmically, suggesting the strategy is efficient.

5.1.2 Experiments

We conduct a series of experiments to evaluate the performance and accuracy of AMoWi-1.

We begin by identifying the region in parameter space where AMoWi-1 performs the fastest. Specifically, we determine the optimal values of δ^+ and δ^- by comparing wall-clock times for the same total integration time. This analysis will inform our choice of parameters for more complex studies later on. To maintain consistency with Chapter 4, we use the Lorenz equations as our test problem and compare standard MoWi with AMoWi-1 by plotting the speedup achieved for different parameter values. We show our results in Figure 5.1, which reveals several key patterns. The algorithm seems to be more sensitive to δ^+ than to δ^-. The darkest region, indicating the highest speedup, is approximately located at $\delta^+ \in [1.25, 1.75]$. This suggests that a low stretching parameter results in optimal performance. Increasing δ^+ in this range generally improves speedup, but a sharp performance cliff appears around $\delta^+ \approx 2$. On the other hand, along the vertical axis, the plot is mostly uniform, suggesting that performance is less sensitive to variations in δ^-.

Next, we rigorously verify that AMoWi-1 preserves the correct statistical properties of the chaotic system using the Lorenz-96 equations. We employ two complementary metrics: a geometrical assessment via Poincaré sections and a quantitative assessment via the ergodic convergence of the energy functional. The left panel of Figure 5.2 displays a Poincaré section at the hyperplane $u_1 = 2.5$. The AMoWi-1 solution (black points) densely populates the same regions as the serial baseline (red points), confirming that the adaptive windowing does not alter the geometric structure of the attractor.

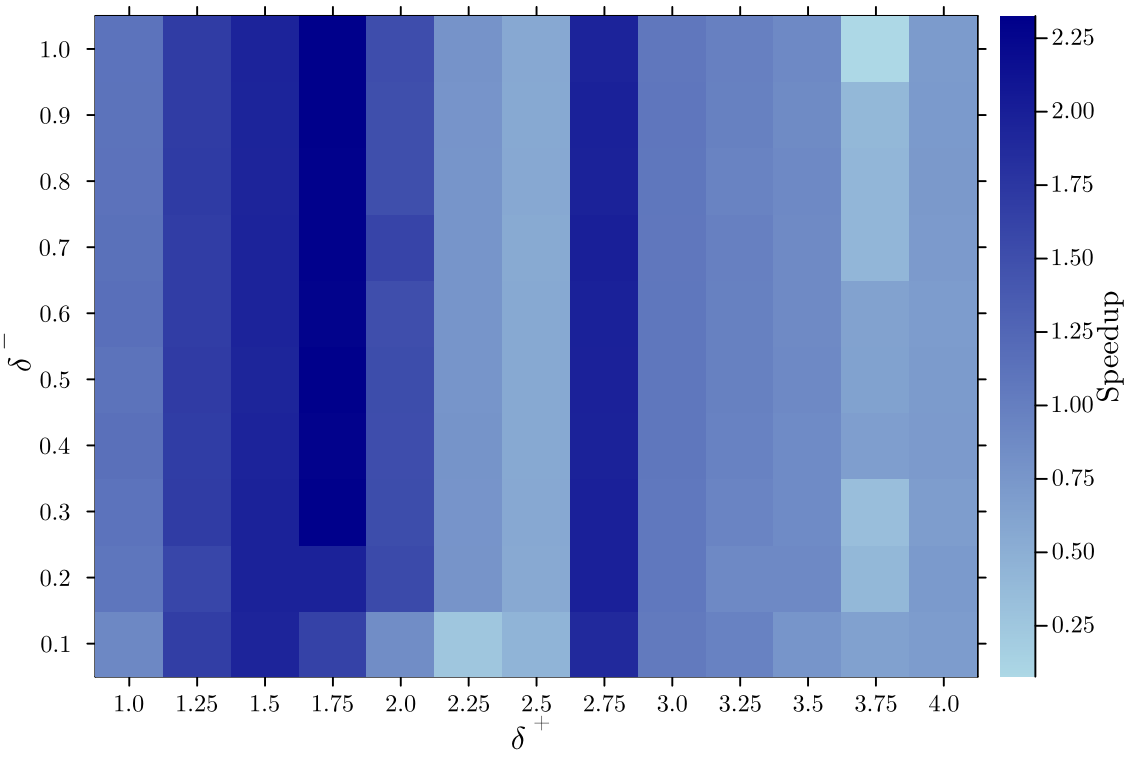

Figure 5.1: Speedup of AMoWi-1 on the Lorenz system as a function of the shrinking (δ^-) and stretching (δ^+) parameters. The heatmap shows that performance is most sensitive to the stretching factor, with an optimal region for $\delta^+ \in [1.25, 1.75]$. Overly aggressive stretching ($\delta^+ \geq 2$) leads to a sharp drop in performance.

The right panel of Figure 5.2 compares the convergence of the system's energy, $E(t)$, to its ergodic limit. To make a fair comparison between the single-trajectory methods (Serial, Parareal) and the ensemble-based methods (MoWi, AMoWi-1), we compute the cumulative time-average of the ensemble mean for the latter. The plot shows the evolution of this cumulative mean for all solvers. Crucially, the AMoWi-1 curve (purple) tracks the Serial, Parareal, and standard MoWi curves almost exactly, converging to the same asymptotic value. This overlap is significant: it proves that the adaptive resizing of windows introduces no statistical bias. The algorithm successfully exploits local speedups without disrupting the long-term ergodic statistics of the dynamics.

Finally, we test AMoWi-1 on the Kuramoto–Sivashinsky equation to demonstrate that it can be effectively applied to partial differential equations. Figure 5.3 presents two visualizations of the solution $u(x,t)$ over the spatial domain $x \in [0,1]$ and time interval $t \in [0,100]$. The top panel shows the solution computed using a standard MoWi approach, whilst the bottom panel shows the solution obtained using AMoWi-1.

The bottom plot shows distinct vertical strips. These are discontinuities and are not numerical errors, but rather the visual signature of the adaptive algorithm operating in a chaotic regime. As the solver progresses, the adaptive stretching and shrinking of τ causes the initial condition of each new window to land on a slightly different shadowing orbit of the global attractor. Because the system is chaotic, these orbits diverge over the window duration. In the plot, the windows overlap; the converged solution of a given window covers the initial segment of the previous one. Consequently, the visible strips correspond to the divergent tail ends of the window solutions, exposing the inevitable orbital mismatch that occurs when patching together local trajectories in a chaotic system.

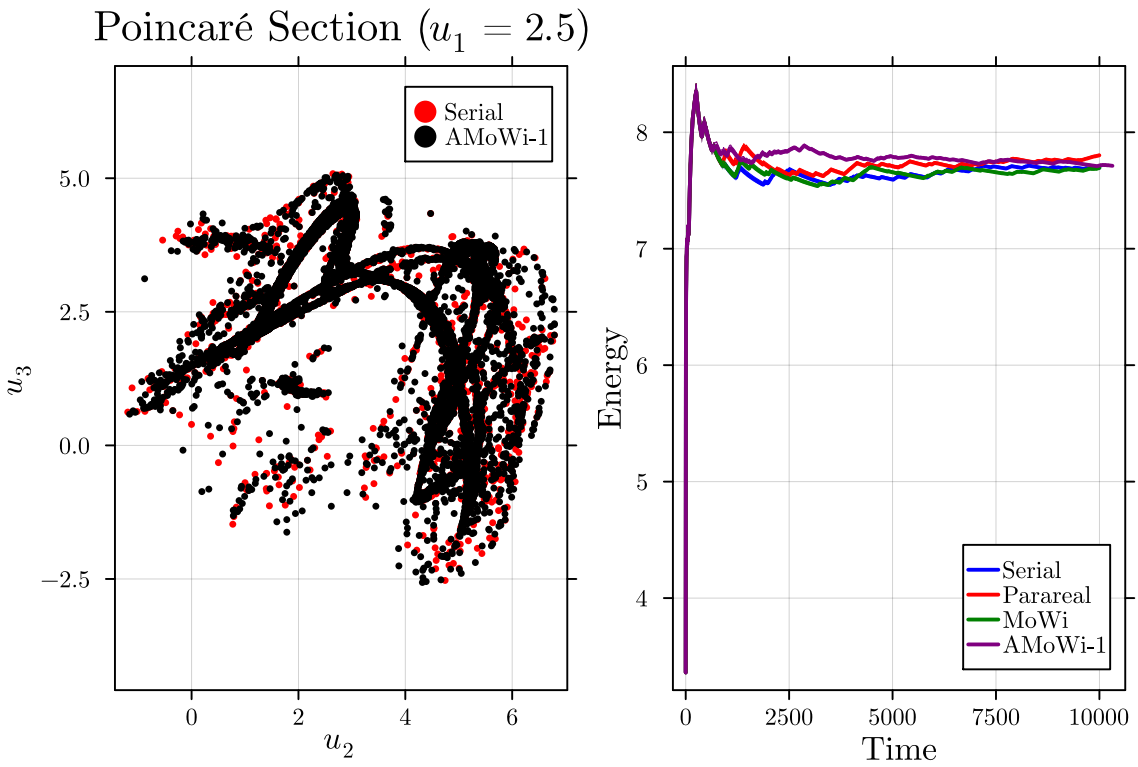

Figure 5.2: Comparison of AMoWi-1 with standard MoWi, Parareal, and the serial solver for the Lorenz-96 equations. (left) Poincaré section at $u_1 = 2.5$, demonstrating that the adaptive windowing preserves the attractor's geometry. (right) Evolution of the cumulative mean energy. The AMoWi-1 estimator (purple) aligns with the serial and Parareal baselines, suggesting convergence to the correct ergodic limit without statistical bias.

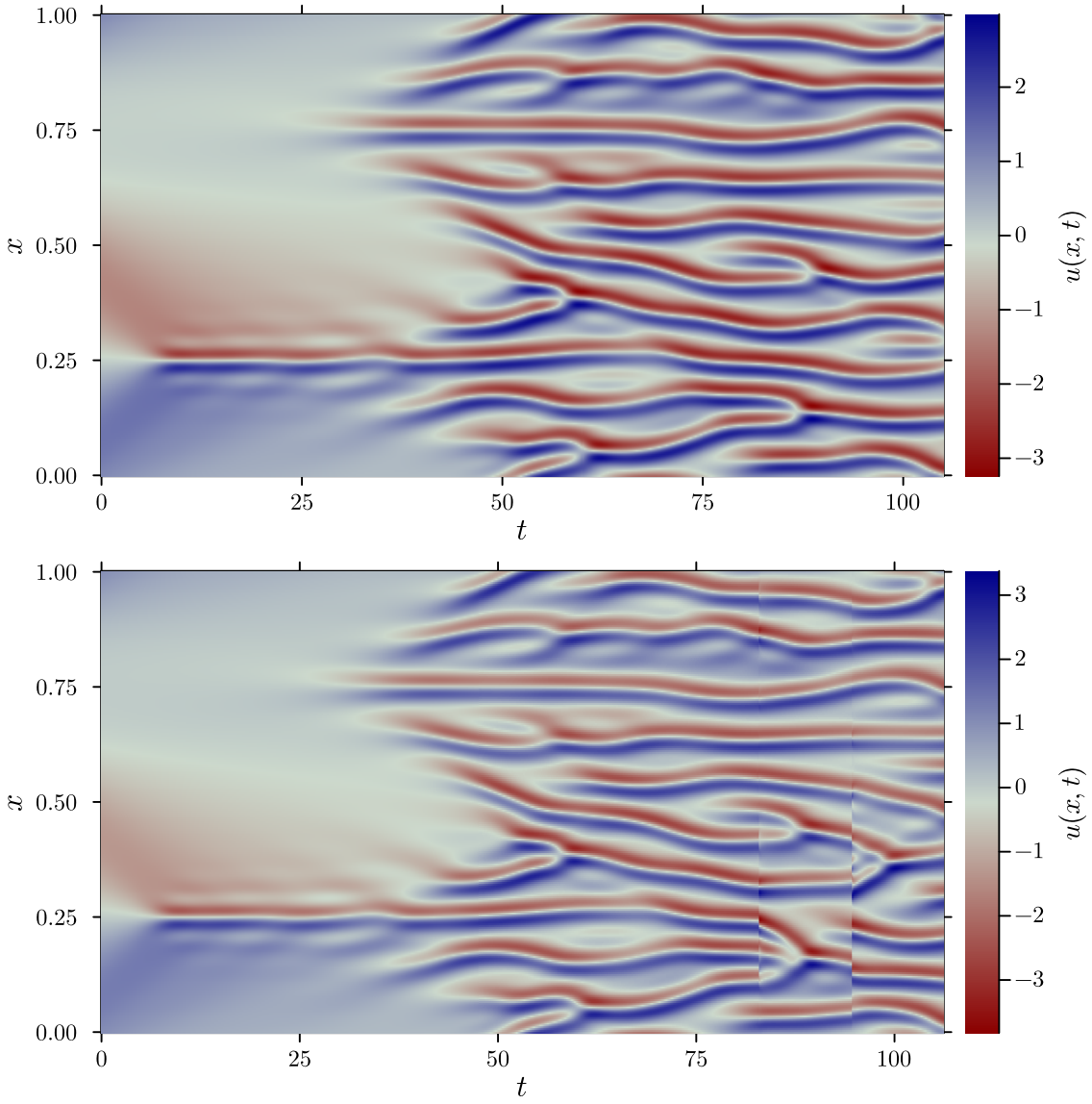

Figure 5.3: Comparison of AMoWi-1 and standard MoWi applied to the Kuramoto–Sivashinsky equation. (top) Standard MoWi. (bottom) AMoWi-1 with adaptive stretching.

5.2 AMoWi-2: Adapting the window shift

In AMoWi-2, we adaptively increase (*leaping*) and decrease (*hopping*) the amount by which we move forward in time from one window to the next. Unlike AMoWi-1, where we adjusted the window length while keeping the shift size constant, AMoWi-2 fixes the window length τ and varies the shift $\Delta\tau$ instead. This approach preserves the overall time window structure but allows flexible spacing between consecutive windows. In this case, only $\Delta\tau$ changes – all the other parameters stay the same.

After solving the first window, we try to shift forwards by running the time-parallel subroutine on the next window. If the window solve is successful, we keep the resulting solution and proceed. Otherwise, we do not shift yet and instead restart with a reduced $\Delta\tau$. In order not to waste computations, we use the recovered solution (or the relevant part of it) as the initial guess.

Note that AMoWi-2 relies on the first time window having converged by default. Since we only update $\Delta\tau$ and not τ, the first window is not affected by our adaptive procedure. If the initial window fails to converge, the entire approach collapses because subsequent windows will depend on the unconverged window. Therefore, the user must choose τ carefully. If τ is too small, there is little room to increase $\Delta\tau$ in subsequent windows. If τ is too large, the first window might fail to converge, rendering the entire procedure useless.

Also, AMoWi-2 is constrained by a fundamental physical limitation: the maximum shift is limited by how quickly the time-parallel subroutine can transfer information. As discussed in Chapter 3, if $\Delta\tau$ is increased too aggressively, the underlying parallel method cannot propagate accurate information across the windows quickly enough, causing the solution to degrade.

Finally, like for AMoWi-1, we may want to leap and hop at different rates. In this case too, we will determine the most effective strategy through numerical experiments. To distinguish these two operations, we introduce separate parameters: δ^+ for leaping (increasing $\Delta\tau$) and δ^- for hopping (decreasing $\Delta\tau$).

5.2.1 Analysis

The biggest difference between AMoWi-1 and AMoWi-2 is that, while AMoWi-1 checks convergence for the current window, AMoWi-2 checks convergence for the next window.

After a successful window solve, we multiply the current shift $\Delta\tau$ by δ^+ to leap forward more. However, we cannot let $\Delta\tau$ grow arbitrarily. We must maintain enough

overlap between consecutive windows to ensure that the time-parallel method can still communicate boundary information accurately. Hence, we impose the constraint

$$\Delta\tau_m = \min(\delta^+ \Delta\tau_{m-1},\, \tau),$$

where τ is the fixed window length. This ensures that AMoWi-2 does not leap too far ahead within each window.

On the other hand, if the window solve is unsuccessful, we decrease $\Delta\tau$ and try again. This constitutes a restart, where

$$\Delta\tau_{m,r} = \delta^- \Delta\tau_{m,r-1}. \tag{5.8}$$

We keep restarting until convergence is achieved. Only then we shift forwards.

The tracking analysis for AMoWi-2 follows directly from Theorem 4.3.2, which provides a bound on the maximum allowable shift, which we denote $\Delta\tau^*$.

Theorem 5.2.1. Consider AMoWi-2 with Parareal as a subroutine. A valid window shift satisfying the tracking condition is guaranteed to be found after at most

$$\iota = \left\lceil \log_{\delta^-} \left(\frac{\Delta\tau^*}{\Delta\tau_{m,0}} \right) \right\rceil \tag{5.9}$$

restarts.

Proof. According to Theorem 4.3.2, there exists a stability bound $\Delta\tau^*$ such that the MoWi algorithm is guaranteed to stay on track provided the window shift satisfies $\Delta\tau \leq \Delta\tau^*$. After ι restarts with a shrinking factor of $\delta^- < 1$, the proposed shift becomes

$$\Delta\tau_{m,\iota} = (\delta^-)^\iota \Delta\tau_{m,0}. \tag{5.10}$$

We are guaranteed to satisfy the tracking condition when the reduced shift falls below the theoretical bound, i.e., when $\Delta\tau_{m,\iota} \leq \Delta\tau^*$. Substituting the expression for the shifted time step, we require

$$(\delta^-)^\iota \Delta\tau_{m,0} \leq \Delta\tau^* \implies (\delta^-)^\iota \leq \frac{\Delta\tau^*}{\Delta\tau_{m,0}}. \tag{5.11}$$

Since $\delta^- < 1$, taking the logarithm to the base δ^- reverses the inequality, yielding

$$\iota \geq \log_{\delta^-} \left(\frac{\Delta\tau^*}{\Delta\tau_{m,0}} \right). \tag{5.12}$$

Because the number of restarts must be an integer, taking the ceiling of this lower bound yields (5.9). This confirms that convergence can always be forced by sufficiently reducing the shift. □

As $\Delta\tau^*$ itself depends on the convergence factor β and the required tolerance r, this analysis directly links the physical dynamics of the system to the behaviour of the adaptive algorithm.

5.2.2 Experiments

We conduct a series of experiments to evaluate the performance and accuracy of AMoWi-2, following the same methodology as used for AMoWi-1, with adjustments appropriate to this context.

We begin by identifying the region in parameter space where AMoWi-2 achieves the best performance. Specifically, we determine the optimal values of δ^+ and δ^- by comparing wall-clock times for the same total integration time. This analysis will inform our choice of parameters for more complex studies later on.

As before, we use the Lorenz equations as our test problem, comparing standard MoWi with AMoWi-2 by plotting the speedup achieved for different parameter values. We show our results in Figure 5.4. The plot is mostly dark, indicating good speedup across a wide range of parameter values. This is because the Lorenz equations exhibit low chaoticity ($\lambda < 1$) so $\Delta\tau^*$ in (5.8) is relatively large. We expect that the stronger the chaoticity, the stricter the upper limit on $\Delta\tau$ – that is, $\Delta\tau^*$ – therefore reducing the size of the high speedup region (dark region). Nevertheless, higher values of δ^+ and δ^- generally produce better speedup. There is a lighter patch around $(\delta^+, \delta^-) \approx (3.75, 0.3)$, which might suggest instability or inefficiency at that particular combination.

Next, we rigorously verify that AMoWi-2 maintains statistical accuracy using the Lorenz-96 equations. As with AMoWi-1, we employ Poincaré sections to assess geometrical fidelity and the energy functional to assess ergodic convergence. The left panel of Figure 5.5 displays a Poincaré section at $u_1 = 2.5$. The AMoWi-2 solution (black points) faithfully recovers the complex structure of the attractor, overlapping densely with the serial baseline (red points).

The right panel of Figure 5.5 presents the evolution of the system's energy, $E(t)$. To enable a direct comparison, we compute the cumulative time-average of the ensemble mean for the adaptive method. The plot confirms that the AMoWi-2 solution (purple curve) converges to the same asymptotic value as the Serial (blue), Parareal (red), and standard MoWi (green) solutions. The curves track each other closely, demonstrating that adaptively varying the window shift does not bias the statistical sampling of the attractor. The method remains stable and ergodically consistent even over long integration times.

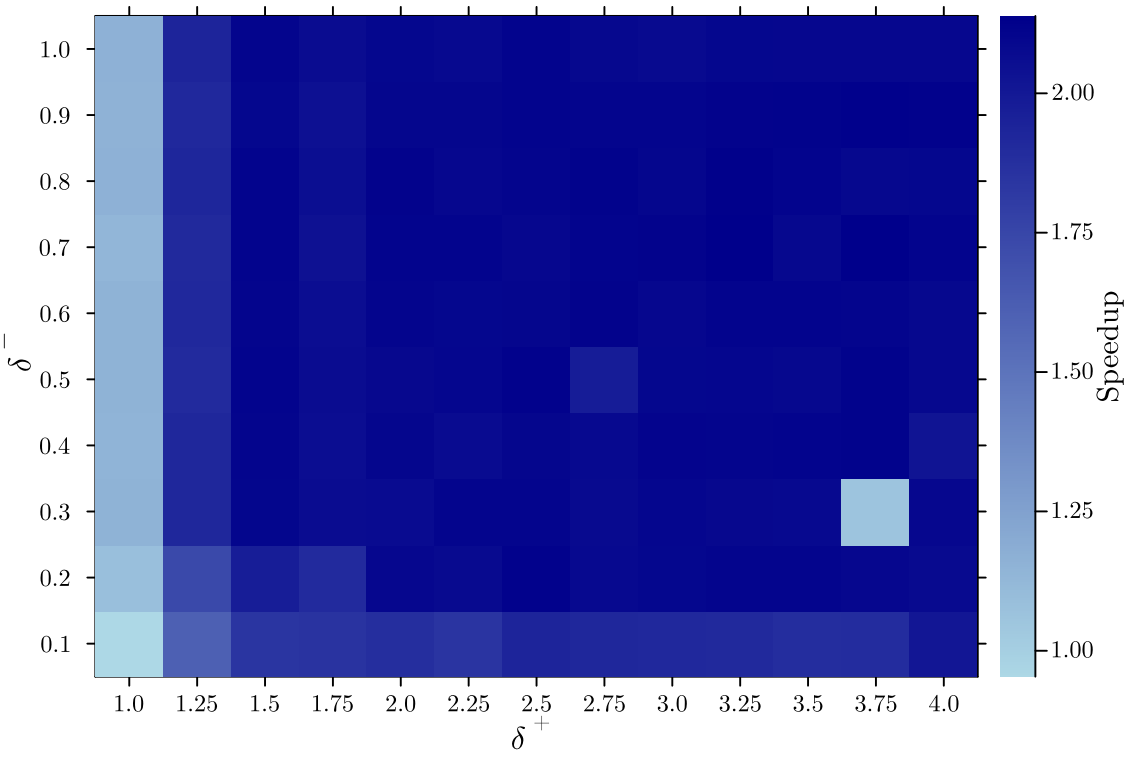

Figure 5.4: Speedup of AMoWi-2 on the Lorenz system as a function of the hopping (δ^-) and leaping (δ^+) parameters. For this problem, good speedup is achieved across a wide range of parameters, indicating that the strategy is robust. The lighter patch near $(\delta^+, \delta^-) \approx (3.75, 0.3)$ suggests a region of potential inefficiency.

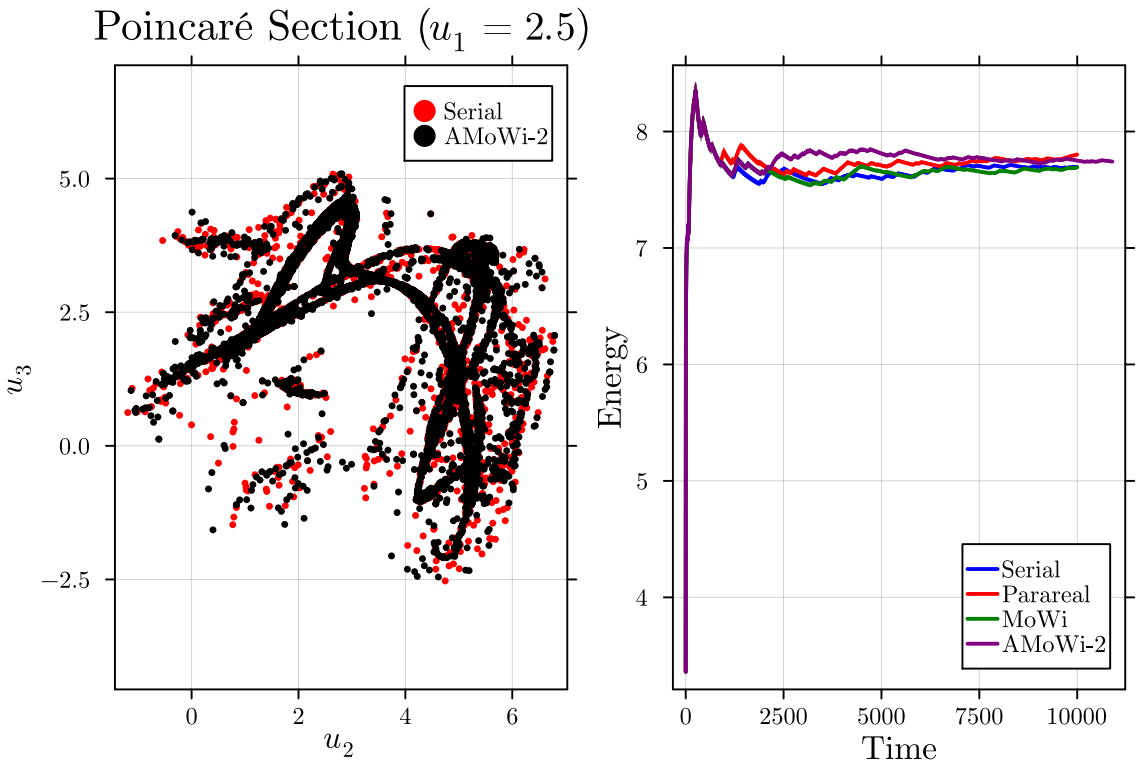

Figure 5.5: Comparison of AMoWi-2 with baselines on Lorenz-96. (left) Poincaré section at $u_1 = 2.5$, showing geometric fidelity to the serial solution. (right) Evolution of the cumulative mean energy. The AMoWi-2 solution (purple) converges to the same asymptotic value as the baselines, suggesting that adaptively varying the window shift does not bias the statistical sampling of the attractor.

Finally, we test AMoWi-2 on the Kuramoto–Sivashinsky equation. Figure 5.6 compares the standard MoWi solution (top) with the AMoWi-2 solution (bottom). The adaptive scheme successfully activates both leaping and hopping. However, we observe that the leaping quickly saturates at the safety ceiling imposed by the window length. Since convergence at this maximum shift value is consistently successful for this problem, the algorithm enters a stable regime of constant forward motion. Visually, this results in a solution structure similar to standard MoWi, but with a crucial operational difference: AMoWi-2 automatically locates and maintains the maximum permissible time shift, effectively integrating the system at the highest safe speed allowed by the time-parallel subroutine.

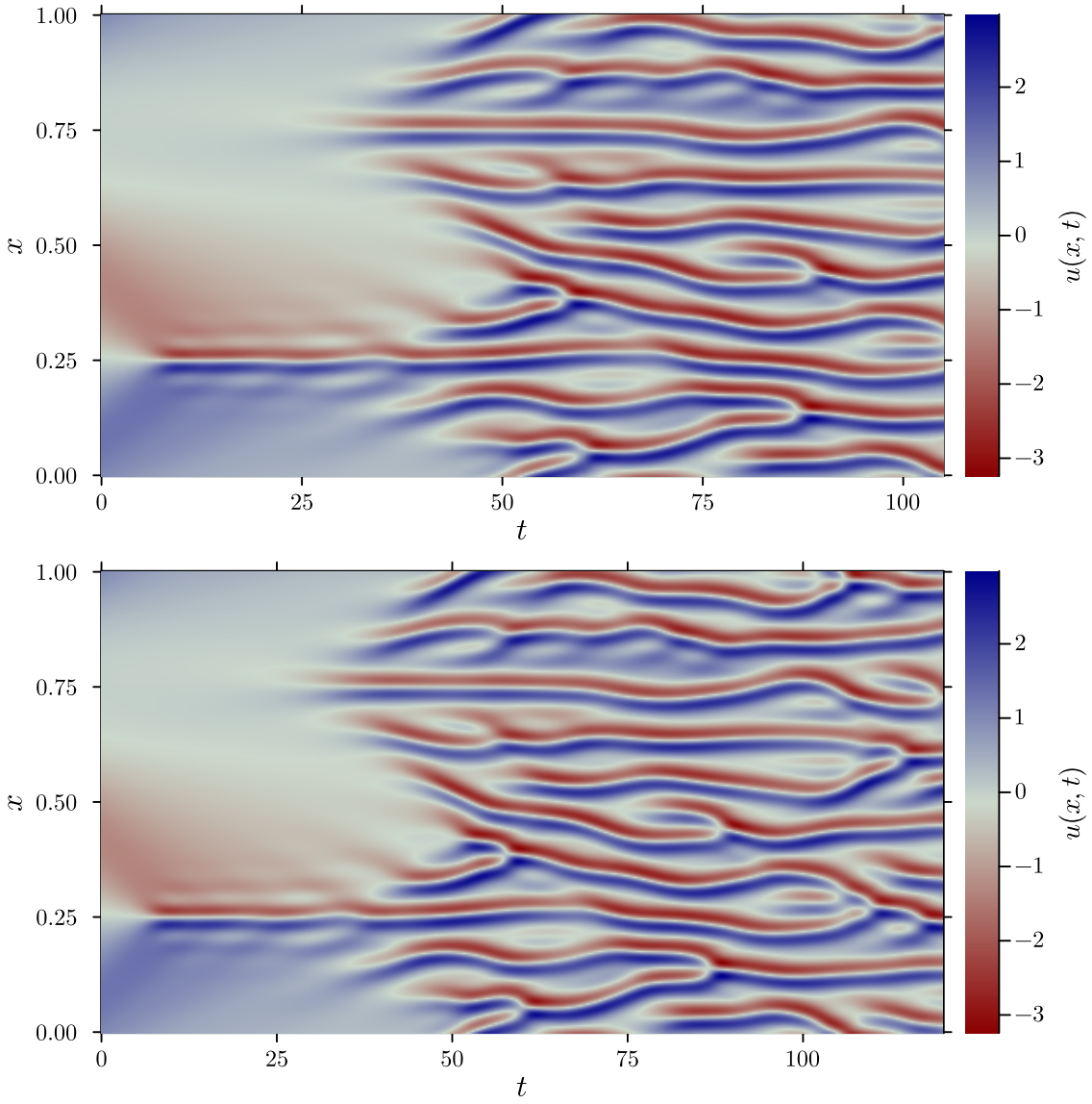

Figure 5.6: Comparison of AMoWi-2 and standard MoWi applied to the Kuramoto–Sivashinsky equation. (top) Standard MoWi. (bottom) AMoWi-2 with adaptive leaping.

5.3 AMoWi-3: Adapting the proximity function weights

In AMoWi-3, we keep both the window length and the shift fixed but adaptively adjust the weights in the time-parallel subroutine's proximity function

$$\psi(U) = \frac{1}{N} \sum_{n=1}^{N} w^{-n} \|U_n - F(U_{n-1})\|. \tag{5.13}$$

Note that the weights are w^{-n}, for which we consider the base weight to be $1/w$, for $w \geq 1$. This strategy is designed to make the time-parallel solver either more or less strict about enforcing continuity at the boundaries between time chunks. If the solver appears to be oversolving, we decrease the base weight (*zooming out*) to allow the subroutine to stop earlier. Conversely, if the window solve is successful, then we move on to a new window and increase the base weight (*zooming out*) to force tighter control over the discontinuities.

AMoWi-3 requires particular care. On one hand, our findings in Chapter 3 suggest that when a time-parallel routine like Parareal is oversolving, decreasing the base weight in the proximity function allows the algorithm to stop earlier and thus prevent it. On the other hand, if we are not careful, we end up actually stopping too early and getting undersolved window solutions. Also, because $\Delta\tau$ stay the same, the user must make sure from the start that their choice is feasible with the time-parallel method.

Finally, like for AMoWi-1 and AMoWi-2, we may want to zoom in and out at different rates. To distinguish these two operations, we introduce separate parameters: δ^- for zooming out (decreasing the base weight) and δ^+ for zooming in (increasing the base weight).

5.3.1 Analysis

To establish theoretical guarantees, we use the same style of restart analysis introduced in AMoWi-1. Specifically, whenever a window does not converge, we discard the partially converged solution and restart from the coarse solution over the same interval, but with updated weights in ψ. Although this approach is somewhat pessimistic compared to practical implementations – which would reuse as much of the previously computed solution as possible – it simplifies the convergence proofs.

Let $\delta > 1$, and $\delta^- = 1/\delta$ and $\delta^+ = \delta$ be the factors by which we increase and decrease the proximity-function base weight. After a failed convergence, a restarts triggers, we multiply the current base weight by δ^-, so that

$$w_{m,r} = \delta\, w_{m,r-1}. \tag{5.14}$$

This makes the subroutine looser, so that it reduces the penalty on discontinuities across subintervals. Intuitively, if Parareal (or any other time-parallel method) struggles to meet the tolerance, loosening the proximity measure should help enforce convergence.

The goal of this analysis is to show that it is possible to guarantee that every window converges within the inner ball, so that MoWi stays on track.

Theorem 5.3.1. Consider AMoWi-3 with Parareal as a subroutine. Each window is guaranteed to converge (and lead to a window shift) after at most

$$\iota = \left\lceil \log_\delta \left(\frac{C_N}{cr} \psi(U^0) \right) + K \log_\delta(Ch^2) \right\rceil \tag{5.15}$$

restarts.

Proof. We mirror the proof of Theorem 5.1.1. After each unsuccessful convergence, we divide the current base weight of the proximity function by $\delta > 1$, in order to make it less sensitive to the time-chunk discontinuities. Thus, after ι restarts, the weight is $\delta^\iota w$. Substituting this into $\psi(U^K)$ effectively divides the measured discontinuities by $\delta^\iota w$, which yields

$$\psi_\delta(U^K) = \frac{1}{N} \sum_{n=1}^{N} (\delta^\iota w)^{-n} \|U_n^K - F(U_{n-1}^K)\| \leq \frac{\psi(U^K)}{\delta^\iota}. \tag{5.16}$$

For a window to have converged correctly, we require $\psi_\delta(U^K) \leq r$, where r is the preset tolerance. Hence, we bound instead (5.16), which is satisfied after at most

$$\iota = \left\lceil \log_\delta \left(\frac{\psi(U^K)}{r} \right) \right\rceil \tag{5.17}$$

restarts. To make a parallel with (5.3), we go one step further: because (5.5) applies in the case of Theorem 5.3.1 too, we can even be more conservative and set ι as in (5.15). □

Notice the $1/(2K)$ factor when comparing (5.3) to (5.15). This seems to suggest that AMoWi-1 should require far less many restarts than AMoWi-3, but it might just be an artefact of our proof. Still, in both cases we have that ι scales logarithmically.

5.3.2 Experiments

We conduct a series of experiments to evaluate the performance and accuracy of AMoWi-3, following the same methodology as used for AMoWi-1 and AMoWi-2, with adjustments appropriate to this context.

We begin by identifying the region in parameter space where AMoWi-3 achieves the best performance. Specifically, we determine the optimal values of δ^+ and δ^- by comparing wall-clock times for the same total integration time. This analysis will inform our choice of parameters for more complex studies later on.

As before, we use the Lorenz equations as our test problem, comparing standard MoWi with AMoWi-3 by plotting the speedup achieved for different parameter values. We show our results in Figure 5.7. The darkest regions (best speedup) appear around $\delta^- \approx 4$ (top of the plot) and $\delta^+ \approx 1$ (close to the right edge). This suggests that higher δ^- and δ^+ values yield better performance.

Next, we rigorously verify that AMoWi-3 maintains statistical accuracy using the Lorenz-96 equations. We employ the same two metrics: geometrical fidelity via Poincaré sections and ergodic convergence via the energy functional. The left panel of Figure 5.8 displays a Poincaré section at $u_1 = 2.5$. The AMoWi-3 solution (black points) densely covers the same regions of the attractor as the serial baseline (red points), confirming that the adaptive relaxation of tolerance does not distort the attractor's geometry.

The right panel of Figure 5.8 illustrates the evolution of the system's energy, $E(t)$. As with the previous strategies, we compare the cumulative time-average of the ensemble mean for the adaptive solver against the baselines. The plot confirms that the AMoWi-3 solution (purple curve) tracks the Serial (blue), Parareal (red), and standard MoWi (green) solutions almost perfectly, converging to the same ergodic limit (≈ 7.8). This result is particularly important for AMoWi-3: it demonstrates that adaptively "zooming out" (loosening the convergence constraint) to prevent oversolving does not compromise the long-term statistical fidelity of the simulation.

Finally, we test AMoWi-3 on the Kuramoto–Sivashinsky equation. Figure 5.9 shows that the solution obtained with adaptive weight adjustment (bottom) maintains fidelity to the solution from standard MoWi (top). The adaptive zooming correctly identifies regions of varying difficulty, relaxing the convergence tolerance where appropriate without sacrificing the overall statistical accuracy of the simulation. This confirms the strategy's applicability to PDE-based problems.

Finally, we test AMoWi-3 on the Kuramoto–Sivashinsky equation. Figure 5.9 shows that the solution obtained with adaptive weight adjustment (bottom) is visually identical to the solution from standard MoWi (top). Although the adaptive mechanism is working, the convergence checks for the time-parallel subroutine are consistently successful. As a result, while the internal error metric evolves, the algorithm never triggers a restart. Since AMoWi-3 maintains fixed τ and $\Delta\tau$, and the convergence

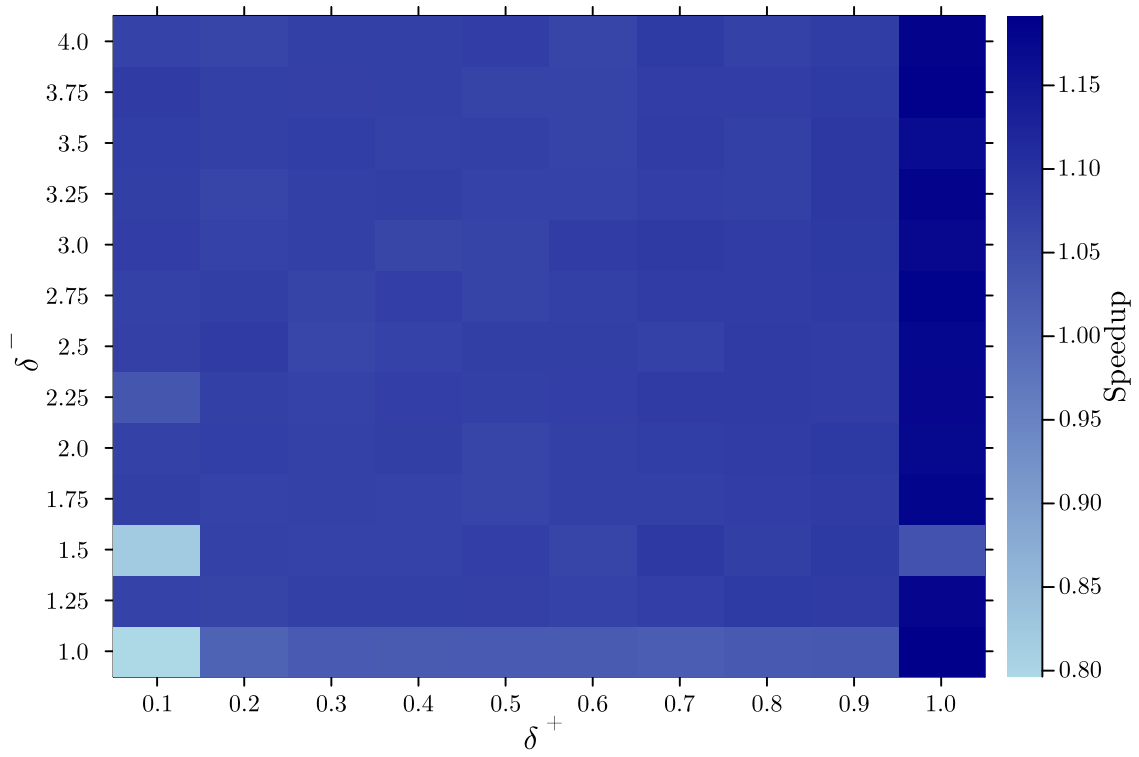

Figure 5.7: Speedup of AMoWi-3 on the Lorenz system as a function of the zooming-in (δ^+) and zooming-out (δ^-) parameters. The results suggest that a more aggressive zooming-out strategy (larger δ^-) combined with a conservative zooming-in strategy ($\delta^+ \approx 1$) yields the best performance for this system.

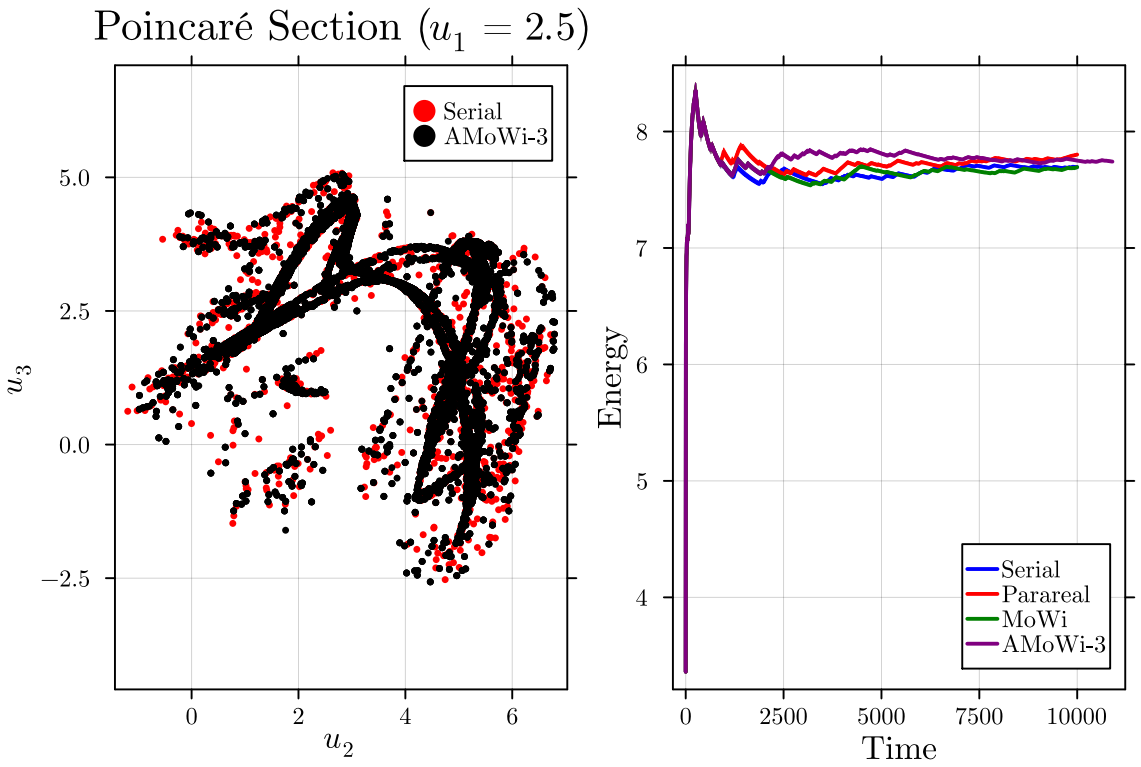

Figure 5.8: Comparison of AMoWi-3 with baselines on Lorenz-96. (left) Poincaré section at $u_1 = 2.5$. (right) Evolution of the cumulative mean energy. The AMoWi-3 curve (purple) tracks the baseline solutions closely, demonstrating that adaptively relaxing the proximity weights (that is, zooming out) maintains the long-term statistical fidelity of the simulation.

criteria are always met, the resulting solution trajectory is indistinguishable from that of standard MoWi. This confirms that the adaptive weighting strategy operates non-intrusively in stable regimes, modifying the convergence strictness without disrupting the successful integration of the dynamics.

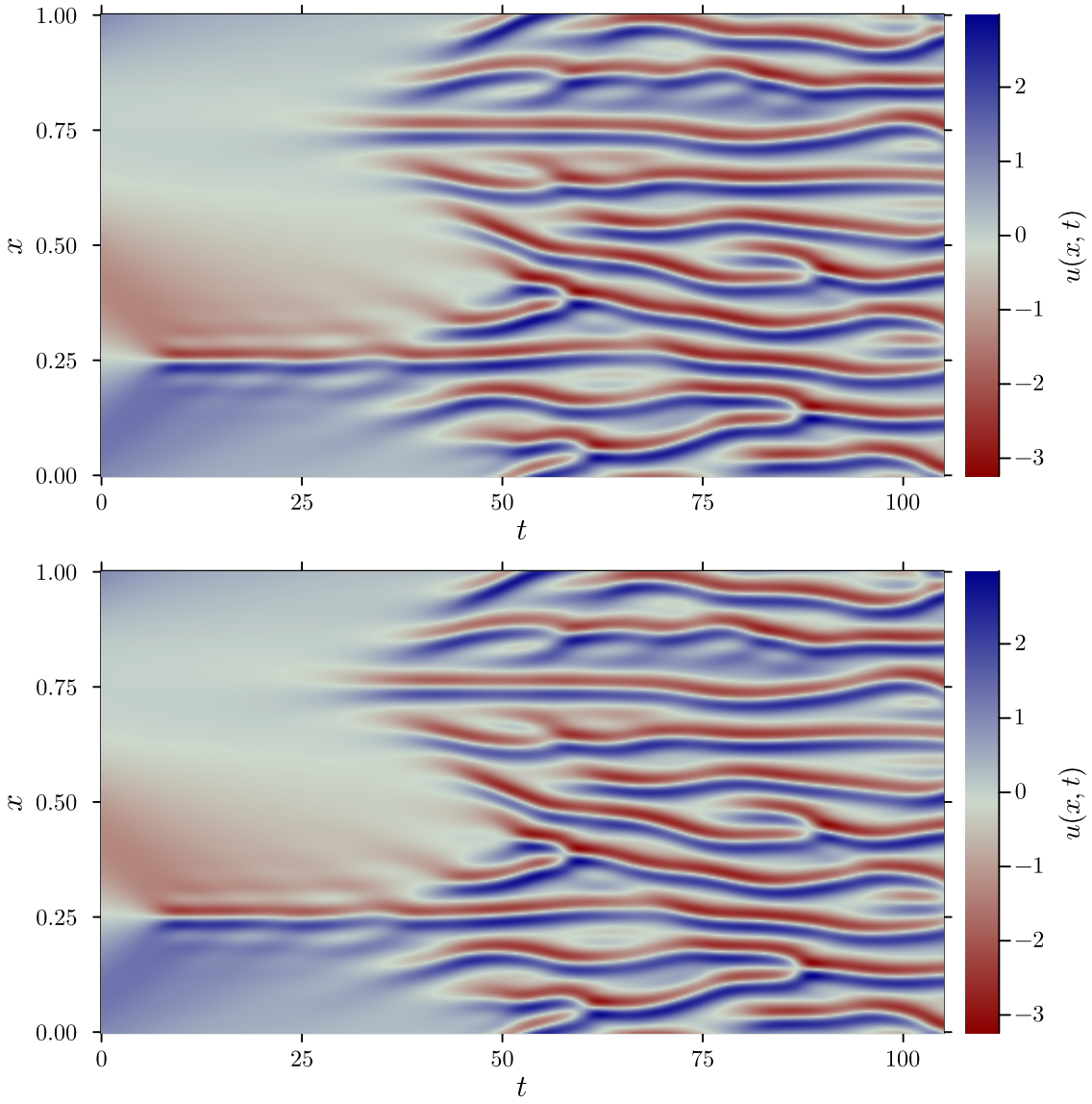

Figure 5.9: Comparison of AMoWi-3 and standard MoWi applied to the Kuramoto–Sivashinsky equation. (top) Standard MoWi. (bottom) AMoWi-3 with adaptive weight adjustment (zooming).

5.4 Combinations

Overall, each AMoWi variant is designed to strike a balance between speed and accuracy within a fixed budget. Our results show that adaptive strategies can reduce computational cost without sacrificing accuracy, but each comes with trade-offs and may require tuning for particular problems. A key open question is how to combine the three strategies (AMoWi-1, AMoWi-2, and AMoWi-3) into a single, unified approach. This challenge opens up several possible directions.

- A natural extension is to combine AMoWi-1 and AMoWi-2 into a strategy that adapts both the window length and the shift at the same time. Implementing this, however, is challenging in practice and may demand trial and error to identify good parameters. Developing a systematic way to coordinate these adjustments remains an open problem.

- Combining AMoWi-2 with AMoWi-3 introduces limits on how large $\Delta\tau$ can be, which depend on the base weight $1/w$ of the proximity function. The bound on $\Delta\tau$ given in Theorem 4.3.2 is conservative but explicitly tied to w. In principle, adjusting the leap size in AMoWi-2 could influence the parameters chosen in AMoWi-3. How these parameters interact remains an open question.

- Finally, it is natural to ask whether AMoWi-1, AMoWi-2, and AMoWi-3 can be merged into a single approach that adapts τ, $\Delta\tau$, and w at the same time. Doing so would mean balancing efficiency, convergence guarantees, and computational cost, and may call for new analytical tools to handle the interactions between these parameters.

Owing to time constraints, we have not explored these combinations and leave them as open problems for future research.

Chapter 6

Conclusion

In this thesis, we asked how chaotic dynamical systems can be simulated efficiently over long time spans using time-parallel methods. Our work was motivated by fusion research, where predicting turbulent plasma behaviour requires long and computationally expensive simulations. We identified a gap in the literature, whereby although many time-parallel methods have been proposed, their theoretical analysis usually applies only to linear problems. As a result, there is much scope for improvement in these methods when they are applied to the systems that matter most in fusion research.

To tackle this challenge, we developed a solution in three parts. First, we built a new theoretical framework for studying how time-parallel methods converge on nonlinear problems. On this basis, we made the moving-window (MoWi) algorithm, a new method designed specifically for the long-term integration of chaotic systems. Finally, we began looking at adaptive extensions of this algorithm (AMoWi) to make it robust and efficient when the system's chaotic behaviour changes over time.

This final chapter sums up our main contributions, outlines the limits of our work, and points to promising directions for future research.

6.1 Our contributions

In this thesis, we make four main contributions to the literature of parallel-in-time methods.

First, we addressed the lack of convergence theory for time-parallel methods on nonlinear systems. We introduced a new framework by modelling the algorithm's iterative process as a contraction mapping. The key innovation is a shift from asymptotic analysis to finite-time guarantees. We achieved this through our outer and inner ball concept, which links the initial error, the convergence factor, and a

fixed iteration budget K. This guarantees that if the initial guess lies within the outer ball, the method is guaranteed to reach the desired tolerance (the inner ball) in a predictable number of steps. A central part of this framework is the proximity function ψ, a chaos-aware stopping rule that prioritizes short-term accuracy while allowing for the trajectory divergence inherent in chaotic systems.

Building on this foundation, we developed MoWi, which tackles the problem of long-time integration by breaking the domain into a series of smaller, overlapping windows, and using a time-parallel method as a subroutine on each. The overlaps ensure that a good initial guess can be passed from one window to the next, speeding up convergence. The theoretical core of this work was our tracking analysis, which uses the outer and inner ball framework to show that MoWi stays on track. Both our cost analysis and numerical experiments confirmed that MoWi can outperform a standard Parareal implementation in the context of long-time chaotic problems.

Recognizing that the dynamics of chaotic systems can vary over time, we introduced three adaptive strategies to make MoWi more efficient and robust. These strategies, which we collectively refer to as adaptive MoWi (AMoWi), adjust key parameters on the fly:

1. AMoWi-1 adapts the window length τ, stretching it in predictable regions and shrinking it in highly chaotic ones.

2. AMoWi-2 adapts the window shift $\Delta\tau$, taking large steps forward when dynamics are stable and smaller ones when they are not.

3. AMoWi-3 adapts the proximity function weight w, tightening or loosening the convergence tolerance to suit the local dynamics.

Numerical experiments showed that each strategy achieves speedup while preserving statistical accuracy.

Lastly, we implemented all algorithms and experiments in two open-source Julia packages, `NSDERungeKutta.jl` and `NSDETimeParallel.jl`. These packages provide a practical, extensible tool that allows other researchers to use, test, and build on our work.

6.2 Limitations

A critical reflection on this work highlights several limitations. Below, we summarise the main limitations of our work, which in turn motivate our proposals for future research.

First, the convergence guarantees for the adaptive strategies in Chapter 5 were established under simplified, idealised models (for example, assuming both fine and coarse step sizes shrink proportionally in AMoWi-1). These proofs offer useful qualitative insight, but a fully rigorous analysis of the implemented algorithms remains an open problem.

Second, the convergence factor β derived in Chapter 3 is based on worst-case estimates of Jacobian norms. Since this comes from a linearisation of the nonlinear problem, the resulting predictions tend to be overly pessimistic.

Third, MoWi and its adaptive variants introduce several new hyperparameters, such as the initial window size τ, the shift $\Delta\tau$, and the adaptive factors δ^+ and δ^-. Our experiments identified parameter ranges for specific test problems, but this thesis does not offer a universal, automated way to choose them in advance. Their best values are likely to depend on the particular dynamics of the system under study.

Fourth, the software we developed, while adequate for validating our theoretical findings, is a proof of concept rather than production-level high-performance code. As noted in Chapter 2, it does not use advanced scheduling techniques and is therefore subject by parallel bottlenecks and communication overheads not accounted for in our idealised cost models.

Finally, the analysis and algorithms in this thesis were developed and tested with Parareal as the core time-parallel subroutine. Although the framework is intended to be general, its application and performance with more advanced methods, such as MGRIT or PFASST, have not been explored.

Despite these gaps, our theoretical framework proved to be a powerful tool. Its main aim was not to deliver exact quantitative predictions, but to guide the design of the algorithms and capture their qualitative behaviour and scaling trends. In this, it succeeded, directly enabling the development of MoWi and AMoWi.

6.3 Future work

The contributions and limitations of this work point to several promising directions for future research.

The most natural next step is to combine the three AMoWi strategies into a single, fully autonomous algorithm that adapts the window length, shift, and proximity weight at the same time. This would require developing a heuristic or control-theoretic approach to manage the complex interplay between these parameters.

Another key test of our framework's generality would be to apply it to other time-parallel algorithms, such as MGRIT or PFASST. Extending the MoWi and AMoWi ideas to these multilevel methods could unlock significant performance gains, especially on large-scale parallel hardware.

Beyond this, the ultimate goal of this research is to speed up real-world simulations. A major next step would be to integrate the AMoWi algorithm into a production-level physics code, such as a gyrokinetic solver for plasma turbulence in tokamaks. This would bring challenges of code complexity, machine architecture, and coupling with existing spatial parallelism.

Finally, the selection of optimal hyperparameters for AMoWi is a difficult task, and one that could be well-suited to machine learning. A promising direction for future research is to train a reinforcement learning agent or neural network to adjust the adaptive parameters $(\tau, \Delta\tau, w)$ in real time, using diagnostics of the simulation state. This could deliver performance beyond what is possible with a fixed heuristic.

In conclusion, in this thesis we have shown that by reframing convergence in a finite-time, chaos-aware setting, we can design robust and efficient algorithms for the long-term integration of chaotic systems. The work offers both the theoretical tools and a practical blueprint for new time-parallel strategies for chaotic dynamics.

Bibliography

[1] http://parallel-in-time.org/references/. (Visited on 05/12/2025).

[2] https://julialang.org/benchmarks/. (Visited on 12/05/2025).

[3] https://github.com/giancarloantonucci/NSDERungeKutta.jl. (Visited on 05/12/2025).

[4] https://github.com/giancarloantonucci/NSDETimeParallel.jl. (Visited on 05/12/2025).

[5] U. M. Ascher, S. J. Ruuth and R. J. Spiteri. 'Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations'. In: *Applied Numerical Mathematics*. Special Issue on Time Integration 25.2 (Nov. 1997), pp. 151–167. ISSN: 0168-9274. DOI: 10.1016/S0168-9274(97)00056-1.

[6] E. Aubanel. 'Scheduling of tasks in the parareal algorithm'. In: *Parallel Computing* 37.3 (Mar. 2011), pp. 172–182. ISSN: 0167-8191. DOI: 10.1016/j.parco.2010.10.004.

[7] G. Bal. "On the Convergence and the Stability of the Parareal Algorithm to Solve Partial Differential Equations". In: *Domain Decomposition Methods in Science and Engineering*. Ed. by T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, T. Schlick, R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. Xu. Vol. 40. Lecture Notes in Computational Science and Engineering. Berlin, Heidelberg: Springer, 2005, pp. 425–432. ISBN: 978-3-540-26825-3. DOI: 10.1007/3-540-26825-1_43.

[8] A. T. Barker. 'A minimal communication approach to parallel time integration'. In: *International Journal of Computer Mathematics* 91.3 (Mar. 2014), pp. 601–615. ISSN: 0020-7160. DOI: 10.1080/00207160.2013.800193.

[9] A. Bellen and M. Zennaro. 'Parallel algorithms for initial-value problems for difference and differential equations'. In: *Journal of Computational and Applied Mathematics* 25.3 (May 1989), pp. 341–350. ISSN: 0377-0427. DOI: 10.1016/0377-0427(89)90037-X.

[10] G. Benettin, L. Galgani, A. Giorgilli, and J.-M. Strelcyn. "Lyapunov Characteristic Exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. Part 1: Theory". In: *Meccanica* 15.1 (Mar. 1980), pp. 9–20. ISSN: 1572-9648. DOI: 10.1007/BF02128236.

[11] G. Benettin, L. Galgani, A. Giorgilli, and J.-M. Strelcyn. "Lyapunov Characteristic Exponents for smooth dynamical systems and for hamiltonian systems; A method for computing all of them. Part 2: Numerical application". In: *Meccanica* 15.1 (Mar. 1980), pp. 21–30. ISSN: 1572-9648. DOI: 10.1007/BF02128237.

[12] J. Bezanson, A. Edelman, S. Karpinski and V. B. Shah. 'Julia: A Fresh Approach to Numerical Computing'. In: *SIAM Review* 59.1 (Mar. 2017), pp. 65–98. ISSN: 0036-1445. DOI: 10.1137/141000671.

[13] S. Blanes and F. Casas. *A Concise Introduction to Geometric Numerical Integration*. Boca Raton: Chapman and Hall/CRC, Nov. 2017. ISBN: 978-1-315-37206-8. DOI: 10.1201/b21563.

[14] M. Bolten, D. Moser, and R. Speck. "A multigrid perspective on the parallel full approximation scheme in space and time". In: *Numerical Linear Algebra with Applications* 24.6 (Dec. 2017), e2110. ISSN: 1099-1506. DOI: 10.1002/nla.2110.

[15] M. Bolten, D. Moser, and R. Speck. "Asymptotic convergence of the parallel full approximation scheme in space and time for linear problems". In: *Numerical Linear Algebra with Applications* 25.6 (Dec. 2018), e2208. ISSN: 1070-5325. DOI: 10.1002/nla.2208.

[16] A. Brandt. "Multi-level adaptive solutions to boundary-value problems". In: *Mathematics of Computation* 31.138 (Apr. 1977), pp. 333–390. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-1977-0431719-X.

[17] J. C. Bronski and T. N. Gambill. "Uncertainty estimates and L2 bounds for the Kuramoto–Sivashinsky equation". In: *Nonlinearity* 19.9 (July 2006), p. 2023. ISSN: 0951-7715. DOI: 10.1088/0951-7715/19/9/002.

[18] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 2016. ISBN: 978-1-119-12150-3. DOI: 10.1002/9781119121534.

[19] C. Cartis and R. Hauser. *A new perspective on the complexity of interior point methods for linear programming*. Tech. rep. University of Oxford, Mar. 2007. URL: https://ora.ox.ac.uk/objects/uuid:7c3c586c-7c90-467b-b5a3-8a19b7fe8e6d (visited on 01/23/2025).

[20] P. Chartier and B. Philippe. "A parallel shooting technique for solving dissipative ODE's". In: *Computing* 51.3 (Sept. 1993), pp. 209–236. ISSN: 1436-5057. DOI: 10.1007/BF02238534.

[21] F. Chen, J. S. Hesthaven, Y. Maday, and A. S. Nielsen. "An Adjoint Approach for Stabilizing the Parareal Method". In: *Comptes Rendus de l'Académie des Sciences*. Série A, Sciences Mathématiques (Sept. 2015). ISSN: 0249-6291.

[22] F. F. Chen. *Introduction to Plasma Physics and Controlled Fusion*. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-22308-7 978-3-319-22309-4. DOI: 10.1007/978-3-319-22309-4. (Visited on 04/27/2026).

[23] A. J. Christlieb, C. B. Macdonald and B. W. Ong. 'Parallel High-Order Integrators'. In: *SIAM Journal on Scientific Computing* 32.2 (Jan. 2010), pp. 818–835. ISSN: 1064-8275. DOI: 10.1137/09075740X.

[24] A. Clarke, C. Davies, D. Ruprecht, S. Tobias and J. S. Oishi. 'Performance of parallel-in-time integration for Rayleigh Bénard convection'. In: *Computing and Visualization in Science* 23.1 (Sept. 2020), p. 10. DOI: 10.1007/s00791-020-00332-3.

[25] B. Cohen, J. Krommes, W. Tang, and M. Rosenbluth. "Non-linear saturation of the dissipative trapped-ion mode by mode coupling". In: *Nuclear Fusion* 16.6 (Dec. 1976), p. 971. ISSN: 0029-5515. DOI: 10.1088/0029-5515/16/6/009.

[26] A. Colagrossi, S. Marrone, P. Colagrossi, and D. Le Touzé. "Da Vinci's observation of turbulence: A French-Italian study aiming at numerically reproducing the physics behind one of his drawings, 500 years later". In: *Physics of Fluids* 33.11 (Nov. 2021), p. 115122. ISSN: 1070-6631. DOI: 10.1063/5.0070984.

[27] J. Cortial and C. Farhat. "A time-parallel implicit method for accelerating the solution of non-linear structural dynamics problems". In: *International Journal for Numerical Methods in Engineering* 77.4 (2009), pp. 451–470. ISSN: 1097-0207. DOI: 10.1002/nme.2418.

[28] P. Cvitanović, R. L. Davidchack and E. Siminos. 'On the State Space Geometry of the Kuramoto–Sivashinsky Flow in a Periodic Domain'. In: *SIAM Journal on Applied Dynamical Systems* 9.1 (2010), pp. 1–33. DOI: 10.1137/070705623. eprint: https://doi.org/10.1137/070705623.

[29] G. G. Dahlquist. "A special stability problem for linear multistep methods". In: *BIT Numerical Mathematics* 3.1 (Mar. 1963), pp. 27–43. ISSN: 1572-9125. DOI: 10.1007/BF01963532.

[30] R. S. Dembo, S. C. Eisenstat and T. Steihaug. 'Inexact Newton Methods'. In: *SIAM Journal on Numerical Analysis* 19.2 (1982), pp. 400–408. DOI: 10.1137/0719025. eprint: https://doi.org/10.1137/0719025.

[31] L. Dieci, M. S. Jolly and E. S. Van Vleck. 'Numerical Techniques for Approximating Lyapunov Exponents and Their Implementation'. In: *Journal of Computational and Nonlinear Dynamics* 6.1 (Sept. 2010), p. 011003. ISSN: 1555-1415. DOI: 10.1115/1.4002088.

[32] V. A. Dobrev, T. Kolev, N. A. Petersson and J. B. Schroder. 'Two-Level Convergence Theory for Multigrid Reduction in Time (MGRIT)'. In: *SIAM Journal on Scientific Computing* 39.5 (Jan. 2017), S501–S527. ISSN: 1064-8275. DOI: 10.1137/16M1074096.

[33] J. Dongarra, S. Gottlieb, and W. T. C. Kramer. "Race to Exascale". In: *Computing in Science & Engineering* 21.1 (Jan. 2019), pp. 4–5. ISSN: 1558-366X. DOI: 10.1109/MCSE.2018.2882574.

[34] J. Dongarra and P. Luszczek. "TOP500". In: *Encyclopedia of Parallel Computing*. Ed. by D. Padua. Boston, MA: Springer US, 2011, pp. 2055–2057. ISBN: 978-0-387-09766-4. DOI: 10.1007/978-0-387-09766-4_157.

[35] E. J. Doyle, W. A. Houlberg, Y. Kamada, V. Mukhovatov, T. H. Osborne, A. Polevoi, G. Bateman, J. W. Connor, J. G. Cordey, T. Fujita, X. Garbet, T. S. Hahm, L. D. Horton, A. E. Hubbard, F. Imbeaux, F. Jenko, J. E. Kinsey, Y. Kishimoto, J. Li, T. C. Luce, Y. Martin, M. Ossipenko, V. Parail, A. Peeters, T. L. Rhodes, J. E. Rice, C. M. Roach, V. Rozhansky, F. Ryter, G. Saibene, R. Sartori, A. C. C. Sips, J. A. Snipes, M. Sugihara, E. J. Synakowski, H. Takenaga, T. Takizuka, K. Thomsen, M. R. Wade, H. R. Wilson, ITPA Transport Physics Topical Group, ITPA Confinement Database and Modelling Topical Group, and ITPA Pedestal and Edge Topical Group. "Chapter 2: Plasma confinement and transport". In: *Nuclear Fusion* 47.6 (June 2007), S18. ISSN: 0029-5515. DOI: 10.1088/0029-5515/47/6/S02.

[36] A. Dutt, L. Greengard, and V. Rokhlin. "Spectral Deferred Correction Methods for Ordinary Differential Equations". In: *BIT Numerical Mathematics* 40.2 (June 2000), pp. 241–266. ISSN: 1572-9125. DOI: 10.1023/A:1022338906936.

[37] J. .-. Eckmann and D. Ruelle. 'Ergodic theory of chaos and strange attractors'. In: *Reviews of Modern Physics* 57.3 (July 1985), pp. 617–656. DOI: 10.1103/RevModPhys.57.617.

[38] R. Eichhorn, S. J. Linz and P. Hänggi. 'Transformation invariance of Lyapunov exponents'. In: *Chaos, Solitons & Fractals* 12.8 (June 2001), pp. 1377–1383. ISSN: 0960-0779. DOI: 10.1016/S0960-0779(00)00120-X.

[39] W. R. Elwasif, S. S. Foley, D. E. Bernholdt, L. A. Berry, D. Samaddar, D. E. Newman and R. Sanchez. 'A dependency-driven formulation of parareal: parallel-in-time solution of PDEs as a many-task application'. In: *Proceedings of the 2011 ACM international workshop on Many task computing on grids and supercomputers*. MTAGS '11. New York, NY, USA: Association for Computing Machinery, Nov. 2011, pp. 15–24. ISBN: 978-1-4503-1145-8. DOI: 10.1145/2132876.2132883.

[40] M. Emmett and M. Minion. 'Toward an efficient parallel in time method for partial differential equations'. In: *Communications in Applied Mathematics and Computational Science* 7.1 (Mar. 2012), pp. 105–132. ISSN: 2157-5452. DOI: 10.2140/camcos.2012.7.105.

[41] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan and J. B. Schroder. 'Parallel Time Integration with Multigrid'. In: *SIAM Journal on Scientific Computing* 36.6 (Nov. 2014), pp. C635–C661. ISSN: 1064-8275. DOI: 10.1137/130944230.

[42] R. D. Falgout, T. A. Manteuffel, B. O'Neill and J. B. Schroder. 'Multigrid Reduction in Time for Nonlinear Parabolic Problems: A Case Study'. In: *SIAM Journal on Scientific Computing* 39.5 (Jan. 2017), S298–S322. ISSN: 1064-8275. DOI: 10.1137/16M1082330.

[43] C. Farhat and M. Chandesris. "Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications". In: *International Journal for Numerical Methods in Engineering* 58.9 (2003), pp. 1397–1434. ISSN: 1097-0207. DOI: 10.1002/nme.860.

[44] C. Farhat, J. Cortial, C. Dastillung, and H. Bavestrello. "Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses". In: *International Journal for Numerical Methods in Engineering* 67.5 (2006), pp. 697–724. ISSN: 1097-0207. DOI: 10.1002/nme.1653.

[45] C. Foias, M. S. Jolly, I. Kukavica, and E. S. Titi. "The Lorenz equation as a metaphor for the Navier-Stokes equations". In: *Discrete and Continuous Dynamical Systems* 7.2 (Apr. 2001), pp. 403–429. ISSN: 1078-0947. DOI: 10.3934/dcds.2001.7.403.

[46] G. Gallavotti and V. Lucarini. "Equivalence of Non-equilibrium Ensembles and Representation of Friction in Turbulent Flows: The Lorenz 96 Model". In: *Journal of Statistical Physics* 156.6 (Sept. 2014), pp. 1027–1065. ISSN: 1572-9613. DOI: 10.1007/s10955-014-1051-6.

[47] M. Gander and L. Halpern. "Absorbing boundary conditions for the wave equation and parallel computing". In: *Mathematics of Computation* 74.249 (2005), pp. 153–176. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-04-01635-7.

[48] M. J. Gander. "50 Years of Time Parallel Time Integration". In: *Multiple Shooting and Time Domain Decomposition Methods*. Ed. by T. Carraro, M. Geiger, S. Körkel, and R. Rannacher. Vol. 9. Contributions in Mathematical and Computational Sciences. Cham: Springer International Publishing, 2015, pp. 69–113. ISBN: 978-3-319-23321-5. DOI: 10.1007/978-3-319-23321-5_3.

[49] M. J. Gander. "A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations". In: *Numerical Linear Algebra with Applications* 6.2 (Mar. 1999), pp. 125–145. ISSN: 1099-1506. DOI: 10.1002/(SICI)1099-1506(199903)6:2<125::AID-NLA152>3.0.CO;2-4.

[50] M. J. Gander. 'Optimized Schwarz Methods'. In: *SIAM Journal on Numerical Analysis* 44.2 (Jan. 2006), pp. 699–731. ISSN: 0036-1429. DOI: 10.1137/S0036142903425409.

[51] M. J. Gander and S. Güttel. 'PARAEXP: A Parallel Integrator for Linear Initial-Value Problems'. In: *SIAM Journal on Scientific Computing* 35.2 (Jan. 2013), pp. C123–C142. ISSN: 1064-8275. DOI: 10.1137/110856137.

[52] M. J. Gander, S. Güttel, and M. Petcu. "A Nonlinear ParaExp Algorithm". In: *Domain Decomposition Methods in Science and Engineering XXIV*. Ed. by P. E. Bjørstad, S. C. Brenner, L. Halpern, H. H. Kim, R. Kornhuber, T. Rahman, and O. B. Widlund. Vol. 125. Lecture Notes in Computational Science and Engineering. Cham: Springer International Publishing, 2018, pp. 261–270. ISBN: 978-3-319-93873-8. DOI: 10.1007/978-3-319-93873-8_24.

[53] M. J. Gander and E. Hairer. "Nonlinear Convergence Analysis for the Parareal Algorithm". In: *Domain Decomposition Methods in Science and Engineering XVII*. Ed. by U. Langer, M. Discacciati, D. E. Keyes, O. B. Widlund, and W. Zulehner. Vol. 60. Lecture Notes in Computational Science and Engineering. Berlin, Heidelberg: Springer, 2008, pp. 45–56. ISBN: 978-3-540-75199-1. DOI: 10.1007/978-3-540-75199-1_4.

[54] M. J. Gander and L. Halpern. "Time Parallelization for Nonlinear Problems Based on Diagonalization". In: *Domain Decomposition Methods in Science and Engineering XXIII*. Ed. by C.-O. Lee, X.-C. Cai, D. E. Keyes, H. H. Kim, A. Klawonn, E.-J. Park, and O. B. Widlund. Vol. 116. Lecture Notes in Computational Science and Engineering. Cham: Springer International Publishing, 2017, pp. 163–170. ISBN: 978-3-319-52389-7. DOI: 10.1007/978-3-319-52389-7_15.

[55] M. J. Gander, L. Halpern, J. Rannou, and J. Ryan. "A Direct Time Parallel Solver by Diagonalization for the Wave Equation". In: *SIAM Journal on Scientific Computing* 41.1 (Jan. 2019), A220–A245. ISSN: 1064-8275. DOI: 10.1137/17M1148347.

[56] M. J. Gander, L. Halpern, J. Ryan, and T. T. B. Tran. "A Direct Solver for Time Parallelization". In: *Domain Decomposition Methods in Science and Engineering XXII*. Ed. by T. Dickopf, M. J. Gander, L. Halpern, R. Krause, and L. F. Pavarino. Vol. 104. Lecture Notes in Computational Science and Engineering. Cham: Springer International Publishing, 2016, pp. 491–499. ISBN: 978-3-319-18827-0. DOI: 10.1007/978-3-319-18827-0_50.

[57] M. J. Gander, F. Kwok, and H. Zhang. "Multigrid interpretations of the parareal algorithm leading to an overlapping variant and MGRIT". In: *Computing and Visualization in Science* 19.3 (July 2018), pp. 59–74. ISSN: 1433-0369. DOI: 10.1007/s00791-018-0297-y.

[58] M. J. Gander and T. Lunet. *Time Parallel Time Integration*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 2024. ISBN: 978-1-61197-801-8. DOI: 10.1137/1.9781611978025.

[59] M. J. Gander and S. Vandewalle. 'Analysis of the Parareal Time-Parallel Time-Integration Method'. In: *SIAM Journal on Scientific Computing* 29.2 (Jan. 2007), pp. 556–578. ISSN: 1064-8275. DOI: 10.1137/05064607X.

[60] E. Giladi and H. B. Keller. "Space-time domain decomposition for parabolic problems". In: *Numerische Mathematik* 93.2 (Dec. 2002), pp. 279–313. ISSN: 0945-3245. DOI: 10.1007/s002110100345.

[61] A. Goddard and A. Wathen. "A note on parallel preconditioning for all-at-once evolutionary PDEs". In: *Verlag der Österreichischen Akademie der Wissenschaften* 51 (June 2019), pp. 135–150. DOI: 10.1553/etna_vol51s135.

[62] W. Hackbusch. 'Parabolic multi-grid methods'. In: *Computing Methods in Applied Sciences and Engineering VI*. Ed. by R. Glowinski and J.-L. Lions. June 1985, pp. 189–197.

[63] J. S. Hadamard. *Lectures on Cauchy's Problem in Linear Partial Differential Equations*. Vol. 18. Yale University Press, 1923.

[64] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*. Vol. 14. Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer, 1996. ISBN: 978-3-642-05220-0 978-3-642-05221-7. DOI: 10.1007/978-3-642-05221-7.

[65] E. Hairer, G. Wanner, and S. P. Nørsett. *Solving Ordinary Differential Equations I*. Vol. 8. Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer, 1993. ISBN: 978-3-540-56670-0 978-3-540-78862-1. DOI: 10.1007/978-3-540-78862-1.

[66] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. 6th ed. Computer Architecture and Design. Morgan Kaufmann, Dec. 2017. ISBN: 978-0-12-811905-1.

[67] F. L. Hinton and R. D. Hazeltine. 'Theory of plasma transport in toroidal confinement systems'. In: *Reviews of Modern Physics* 48.2 (Apr. 1976), pp. 239–308. DOI: 10.1103/RevModPhys.48.239.

[68] M. Hochbruck and A. Ostermann. "Exponential integrators". In: *Acta Numerica* 19 (2010), pp. 209–286. ISSN: 0962-4929. DOI: 10.1017/S0962492910000048.

[69] A. P. Hooper and R. Grimshaw. 'Nonlinear instability at the interface between two viscous fluids'. In: *The Physics of Fluids* 28.1 (Jan. 1985), pp. 37–45. ISSN: 0031-9171. DOI: 10.1063/1.865160.

[70] Y. Horibe and N. Ogura. "Deuterium content as a parameter of water mass in the ocean". In: *Journal of Geophysical Research (1896-1977)* 73.4 (Feb. 1968), pp. 1239–1249. ISSN: 2156-2202. DOI: 10.1029/JB073i004p01239.

[71] G. Horton and S. Vandewalle. 'A Space-Time Multigrid Method for Parabolic Partial Differential Equations'. In: *SIAM Journal on Scientific Computing* 16.4 (July 1995), pp. 848–864. ISSN: 1064-8275. DOI: 10.1137/0916050.

[72] A. J. Howse, H. D. Sterck, R. D. Falgout, S. MacLachlan, and J. Schroder. "Parallel-In-Time Multigrid with Adaptive Spatial Coarsening for The Linear Advection and Inviscid Burgers Equations". In: *SIAM Journal on Scientific Computing* 41.1 (Jan. 2019), A538–A565. ISSN: 1064-8275. DOI: 10.1137/17M1144982.

[73] E. Kazantsev. 'Local Lyapunov exponents of the quasi-geostrophic ocean dynamics'. In: *Applied Mathematics and Computation* 104.2 (Sept. 1999), pp. 217–257. ISSN: 0096-3003. DOI: 10.1016/S0096-3003(98)10078-4.

[74] A. N. Kolmogorov. “Dissipation of energy in the locally isotropic turbulence”. Trans. by V. Levin. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 434.1890 (1991). Ed. by J. C. R. Hunt, O. M. Phillips, and D. Williams, pp. 15–17. DOI: 10.1098/rspa.1991.0076. Trans. of A. H. Колмогоров. «Рассеяние энергии при локально изотропной турбулентности». В: *Доклады Академии Наук СССР* 32 (1941), с. 19—21.

[75] A. N. Kolmogorov. “The local structure of turbulence in incompressible viscous fluid for very large Reynolds number”. Trans. by V. Levin. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 434.1890 (1991). Ed. by J. C. R. Hunt, O. M. Phillips, and D. Williams, pp. 9–13. DOI: 10.1098/rspa.1991.0075. Trans. of A. H. Колмогоров. «Локальная структура турбулентности в несжимаемой вязкой жидкости при очень больших числах Рейнольдса». В: *Доклады Академии Наук СССР* 30 (1941), с. 299—303.

[76] G. L. Kooij, M. A. Botchev and B. J. Geurts. ‘A block Krylov subspace implementation of the time-parallel Paraexp method and its extension for nonlinear partial differential equations’. In: *Journal of Computational and Applied Mathematics* 316 (May 2017), pp. 229–246. DOI: 10.1016/j.cam.2016.09.036.

[77] Y. Kuramoto and T. Tsuzuki. ‘On the Formation of Dissipative Structures in Reaction-Diffusion Systems: Reductive Perturbation Approach’. In: *Progress of Theoretical Physics* 54.3 (Sept. 1975), pp. 687–699. ISSN: 0033-068X. DOI: 10.1143/PTP.54.687.

[78] F. Kwok and B. W. Ong. “Schwarz Waveform Relaxation with Adaptive Pipelining”. In: *SIAM Journal on Scientific Computing* 41.1 (Jan. 2019), A339–A364. ISSN: 1064-8275. DOI: 10.1137/17M115311X.

[79] R. E. LaQuey, S. M. Mahajan, P. H. Rutherford and W. M. Tang. ‘Nonlinear Saturation of the Trapped-Ion Mode’. In: *Physical Review Letters* 34.7 (Feb. 1975), pp. 391–394. DOI: 10.1103/PhysRevLett.34.391.

[80] J. D. Lawson. “Some Criteria for a Power Producing Thermonuclear Reactor”. In: *Proceedings of the Physical Society. Section B* 70.1 (Jan. 1957), p. 6. ISSN: 0370-1301. DOI: 10.1088/0370-1301/70/1/303.

[81] C. Li and G. Chen. ‘Estimating the Lyapunov exponents of discrete systems’. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 14.2 (June 2004), pp. 343–346. ISSN: 1054-1500. DOI: 10.1063/1.1741751.

[82] C. Li and X. Xia. ‘On the bound of the Lyapunov exponents for continuous systems’. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 14.3 (Sept. 2004), pp. 557–561. ISSN: 1054-1500. DOI: 10.1063/1.1768911.

[83] P. C. Liewer. “Measurements of microturbulence in tokamaks and comparisons with theories of turbulence and anomalous transport”. In: *Nuclear Fusion* 25.5 (May 1985), p. 543. ISSN: 0029-5515. DOI: 10.1088/0029-5515/25/5/004.

[84] J.-L. Lions, Y. Maday and G. Turinici. 'Résolution d'EDP par un schéma en temps «pararéel»'. In: *Comptes Rendus de l'Académie des Sciences*. Série I - Mathematics 332.7 (Apr. 2001), pp. 661–668. ISSN: 0764-4442. DOI: 10.1016/S0764-4442(00)01793-6.

[85] E. N. Lorenz. "Deterministic Nonperiodic Flow". In: *Journal of the Atmospheric Sciences* 20.2 (Mar. 1963), pp. 130–141. ISSN: 0022-4928, 1520-0469. DOI: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

[86] E. N. Lorenz. 'Reply to comment by L.-S. Yao and D. Hughes'. In: *Tellus A: Dynamic Meteorology and Oceanography* 60.4 (May 2008), pp. 806–807. DOI: 10.1111/j.1600-0870.2007.00302.x.

[87] E. Lorenz. "Predictability: a problem partly solved". In: *Seminar on Predictability, 4-8 September 1995*. Vol. 1. Shinfield Park, Reading: ECMWF, 1995, pp. 1–18. DOI: 10.1017/CBO9780511617652.004.

[88] A. M. Lyapunov. "The general problem of the stability of motion". Trans. by A. T. Fuller. In: *International Journal of Control* 55.3 (Mar. 1992), pp. 531–773. Trans. of A. M. LIAPOUNOFF. « Problème général de la stabilité du mouvement ». Trad. par É. M. DAVAUX. In : *Annales de la Faculté des Sciences de Toulouse*. 2e sér. 9 (1907), p. 203-474. Trans. of A. M. Ляпунов. «Общая задача об устойчивости движения». Дис. ... док. Императорский Харьковский университет, 1892.

[89] Y. Maday. "The 'Parareal in Time' Algorithm". In: *Substructuring Techniques and Domain Decomposition Methods*. Ed. by F. Magoulès. Vol. 24. Computational Science, Engineering & Technology. Stirlingshire, UK: Saxe-Coburg Publications, May 2010, pp. 19–44. ISBN: 978-1-874672-33-3. DOI: 10.4203/csets.24.2.

[90] Y. Maday and E. M. Rønquist. "Parallelization in time through tensor-product space–time solvers". In: *Comptes Rendus. Mathématique* 346.1-2 (2008), pp. 113–118. ISSN: 1778-3569. DOI: 10.1016/j.crma.2007.09.012.

[91] Y. Maday and G. Turinici. "A parareal in time procedure for the control of partial differential equations". In: *Comptes Rendus. Mathématique* 335.4 (2002), pp. 387–392. ISSN: 1778-3569. DOI: 10.1016/S1631-073X(02)02467-6.

[92] J. C. Maxwell. *Matter and Motion*. London: Society for Promoting Christian Knowledge, 1876.

[93] E. McDonald, J. Pestana and A. Wathen. 'Preconditioning and Iterative Solution of All-at-Once Systems for Evolutionary Partial Differential Equations'. In: *SIAM Journal on Scientific Computing* 40.2 (Mar. 2018), A1012–A1033. ISSN: 1064-8275. DOI: 10.1137/16M1062016.

[94] U. Miekkala and O. Nevanlinna. 'Convergence of Dynamic Iteration Methods for Initial Value Problems'. In: *SIAM Journal on Scientific and Statistical Computing* 8.4 (July 1987), pp. 459–482. ISSN: 0196-5204. DOI: 10.1137/0908046.

[95] W. L. Miranker and W. Liniger. “Parallel methods for the numerical integration of ordinary differential equations”. In: *Mathematics of Computation* 21.99 (July 1967), pp. 303–320. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-1967-0223106-8.

[96] K. Miyamoto. *Fundamentals of plasma physics and controlled fusion*. Tech. rep. NIFS-PROC–88. Toki: National Institutes of Natural Sciences, Japan, June 2011. (Visited on 04/27/2026).

[97] J. M. Nese. ‘Quantifying local predictability in phase space’. In: *Physica D: Nonlinear Phenomena* 35.1 (Apr. 1989), pp. 237–250. ISSN: 0167-2789. DOI: 10.1016/0167-2789(89)90105-X.

[98] J. Nievergelt. ‘Parallel methods for integrating ordinary differential equations’. In: *Communications of the ACM* 7.12 (Dec. 1964), pp. 731–733. ISSN: 0001-0782. DOI: 10.1145/355588.365137.

[99] B. W. Ong, R. D. Haynes and K. Ladd. ‘Algorithm 965: RIDC Methods: A Family of Parallel Time Integrators’. In: *ACM Trans. Math. Softw.* 43.1 (Aug. 2016), 8:1–8:13. ISSN: 0098-3500. DOI: 10.1145/2964377.

[100] B. W. Ong and J. B. Schroder. ‘Applications of time parallelization’. In: *Computing and Visualization in Science* 23.1 (Sept. 2020), p. 11. ISSN: 1433-0369. DOI: 10.1007/s00791-020-00331-4.

[101] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, 1970. ISBN: 978-0-12-528550-6.

[102] I. V. Oseledets. «The multiplicative ergodic theorem. Lyapunov characteristic numbers of dynamical systems». In: *Transactions of the Moscow Mathematical Society* 19 (1968), pp. 197–231. Trans. of И. В. Оселедец. «Мультипликативная эргодическая теорема. Характеристические показатели Ляпунова динамических систем». В: *Издательство Московского университета* 19 (1968), с. 179—210.

[103] F. Otto. ‘Optimal bounds on the Kuramoto–Sivashinsky equation’. In: *Journal of Functional Analysis* 257.7 (Oct. 2009), pp. 2188–2245. ISSN: 0022-1236. DOI: 10.1016/j.jfa.2009.01.034.

[104] L. Pareschi and G. Russo. “Implicit–Explicit Runge–Kutta Schemes and Applications to Hyperbolic Systems with Relaxation”. In: *Journal of Scientific Computing* 25.1 (Oct. 2005), pp. 129–155. ISSN: 1573-7691. DOI: 10.1007/s10915-004-4636-4.

[105] Y. B. Pesin. ‘Characteristic Lyapunov exponents and smooth ergodic theory’. In: *Russian Mathematical Surveys* 32.4 (Aug. 1977), pp. 55–114. DOI: 10.1070/RM1977v032n04ABEH001639. Trans. of Я. Б. Песин. ‘Характеристические показатели Ляпунова и гладкая эргодическая теория’. In: *Успехи математических наук* 32.4 (Aug. 1977), pp. 55–112.

[106] J. H. Poincaré. *Science and Method*. Trans. by F. Maitland. Thomas Nelson, 1914. Trans. of E. Flammarion, éd. *Science et Méthode*. Bibliothèque de philosophie scientifique. Paris : Ernest Flammarion, 1908.

[107] A. Quarteroni, A. Valli, A. Quarteroni, and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Numerical Mathematics and Scientific Computation. Oxford, New York: Oxford University Press, May 1999. ISBN: 978-0-19-850178-7.

[108] V. Rao and A. Sandu. 'An adjoint-based scalable algorithm for time-parallel integration'. In: *Journal of Computational Science*. Empowering Science through Computing + BioInspired Computing 5.2 (Mar. 2014), pp. 76–84. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2013.03.004.

[109] J. Regier, A. Miller, J. McAuliffe, R. Adams, M. Hoffman, D. Lang, D. Schlegel, and M. Prabhat. "Celeste: Variational inference for a generative model of astronomical images". In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. PMLR, June 2015, pp. 2095–2103.

[110] J. M. Reynolds-Barredo, D. E. Newman and R. Sanchez. 'An analytic model for the convergence of turbulent simulations time-parallelized via the parareal algorithm'. In: *Journal of Computational Physics* 255 (Dec. 2013), pp. 293–315. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2013.08.028.

[111] J. M. Reynolds-Barredo, D. E. Newman, R. Sanchez, D. Samaddar, L. A. Berry and W. R. Elwasif. 'Mechanisms for the convergence of time-parallelized, parareal turbulent plasma simulations'. In: *Journal of Computational Physics* 231.23 (Oct. 2012), pp. 7851–7867. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2012.07.028.

[112] L. F. Richardson. *Weather prediction by numerical processes*. London: Cambridge University Press, 1922.

[113] A. Rizvi and K. C. Hale. *A Look at Communication-Intensive Performance in Julia*. Sept. 2021. DOI: 10.48550/arXiv.2109.14072. arXiv: 2109.14072 [cs]. (Visited on 28/04/2026).

[114] K. Rupp. *Microprocessor Trend Data (2022)*. 2022. URL: https://github.com/karlrupp/microprocessor-trend-data (visited on 09/03/2025).

[115] D. Ruprecht and R. Krause. 'Explicit parallel-in-time integration of a linear acoustic-advection system'. In: *Computers & Fluids* 59 (Apr. 2012), pp. 72–83. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2012.02.015.

[116] D. Ruprecht. "Wave propagation characteristics of Parareal". In: *Computing and Visualization in Science* 19.1 (June 2018), pp. 1–17. ISSN: 1433-0369. DOI: 10.1007/s00791-018-0296-z.

[117] D. Ruprecht, R. Speck, and R. Krause. "Parareal for Diffusion Problems with Space- and Time-Dependent Coefficients". In: *Domain Decomposition Methods in Science and Engineering XXII*. Ed. by T. Dickopf, M. J. Gander, L. Halpern, R. Krause, and L. F. Pavarino. Vol. 104. Lecture Notes in Computational Science and Engineering. Cham: Springer International Publishing, 2016, pp. 371–378. ISBN: 978-3-319-18827-0. DOI: 10.1007/978-3-319-18827-0_37.

[118] P. Saha, J. Stadel and S. Tremaine. 'A Parallel Integration Method for Solar System Dynamics'. In: *The Astronomical Journal* 114 (July 1997), p. 409. ISSN: 0004-6256. DOI: 10.1086/118485.

[119] D. Samaddar, D. E. Newman and R. Sánchez. 'Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm'. In: *Journal of Computational Physics* 229.18 (Sept. 2010), pp. 6558–6573. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2010.05.012.

[120] D. Sheen, I. Sloan, and V. Thomée. "A parallel method for time-discretization of parabolic problems based on contour integral representation and quadrature". In: *Mathematics of Computation* 69.229 (Jan. 2000), pp. 177–195. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-99-01098-4.

[121] G. I. Sivashinsky and D. M. Michelson. 'On Irregular Wavy Flow of a Liquid Film Down a Vertical Plane'. In: *Progress of Theoretical Physics* 63.6 (June 1980), pp. 2112–2114. ISSN: 0033-068X. DOI: 10.1143/PTP.63.2112.

[122] G. I. Sivashinsky. "Nonlinear analysis of hydrodynamic instability in laminar flames—I. Derivation of basic equations". In: *Acta Astronautica* 4.11 (Nov. 1977), pp. 1177–1206. ISSN: 0094-5765. DOI: 10.1016/0094-5765(77)90096-0.

[123] B. S. Southworth. "Necessary Conditions and Tight Two-level Convergence Bounds for Parareal and Multigrid Reduction in Time". In: *SIAM Journal on Matrix Analysis and Applications* 40.2 (Apr. 2019), pp. 564–608. ISSN: 0895-4798. DOI: 10.1137/18M1226208.

[124] R. Speck, D. Ruprecht, R. Krause, M. Emmett, M. Minion, M. Winkel and P. Gibbon. 'A massively space-time parallel N-body solver'. In: *SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. Nov. 2012, pp. 1–11. DOI: 10.1109/SC.2012.6.

[125] G. A. Staff and E. M. Rønquist. "Stability of the Parareal Algorithm". In: *Domain Decomposition Methods in Science and Engineering*. Ed. by T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, T. Schlick, R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. Xu. Berlin, Heidelberg: Springer, 2005, pp. 449–456. ISBN: 978-3-540-26825-3. DOI: 10.1007/3-540-26825-1_46.

[126] J. Steiner, D. Ruprecht, R. Speck, and R. Krause. "Convergence of Parareal for the Navier-Stokes Equations Depending on the Reynolds Number". In: *Numerical Mathematics and Advanced Applications - ENUMATH 2013*. Ed. by A. Abdulle, S. Deparis, D. Kressner, F. Nobile, and M. Picasso. Lecture Notes in Computational Science and Engineering. Cham: Springer International Publishing, 2015, pp. 195–202. ISBN: 978-3-319-10705-9. DOI: 10.1007/978-3-319-10705-9_19.

[127] A. M. Stuart and A. R. Humphries. *Dynamical Systems and Numerical Analysis*. Cambridge Monographs on Applied and Computational Mathematics. Aug. 1996. ISBN: 978-0-521-49672-8.

[128] E. Süli and D. F. Mayers. *An Introduction to Numerical Analysis*. Cambridge: Cambridge University Press, 2003. ISBN: 978-0-521-00794-8. DOI: 10.1017/CBO9780511801181. (Visited on 01/23/2025).

[129] E. Tadmor. 'The Well-Posedness of the Kuramoto–Sivashinsky Equation'. In: *SIAM Journal on Mathematical Analysis* 17.4 (July 1986), pp. 884–893. ISSN: 0036-1410. DOI: 10.1137/0517063.

[130] R. Temam. *Infinite-Dimensional Dynamical Systems in Mechanics and Physics*. Ed. by F. John, J. E. Marsden and L. Sirovich. 1st ed. Vol. 68. Applied Mathematical Sciences. New York, NY: Springer US, 1988. ISBN: 978-1-4684-0315-2 978-1-4684-0313-8. DOI: 10.1007/978-1-4684-0313-8.

[131] *TOP500 List (November 2022)*. Nov. 2022. (Visited on 09/03/2025).

[132] C. V. Tran. 'The number of degrees of freedom of three-dimensional Navier–Stokes turbulence'. In: *Physics of Fluids* 21.12 (Dec. 2009), p. 125103. ISSN: 1070-6631. DOI: 10.1063/1.3276295.

[133] C. V. Tran and L. Blackbourn. 'Number of degrees of freedom of two-dimensional turbulence'. In: *Physical Review E* 79.5 (May 2009), p. 056308. DOI: 10.1103/PhysRevE.79.056308.

[134] C. V. Tran and X. Yu. 'Bounds for the number of degrees of freedom of incompressible magnetohydrodynamic turbulence in two and three dimensions'. In: *Physical Review E* 85.6 (June 2012), p. 066323. DOI: 10.1103/PhysRevE.85.066323.

[135] W. Tucker. 'The Lorenz attractor exists'. In: *Comptes Rendus de l'Académie des Sciences*. Série I - Mathematics 328.12 (June 1999), pp. 1197–1202. ISSN: 0764-4442. DOI: 10.1016/S0764-4442(99)80439-X.

[136] D. L. van Kekem and A. E. Sterk. 'Travelling waves and their bifurcations in the Lorenz-96 model'. In: *Physica D: Nonlinear Phenomena* 367 (Mar. 2018), pp. 38–60. ISSN: 0167-2789. DOI: 10.1016/j.physd.2017.11.008.

[137] J. Wesson. *Tokamaks*. 4th ed. International Series of Monographs on Physics. Oxford, New York: Oxford University Press, Dec. 2011. ISBN: 978-0-19-959223-4.

[138] R. W. Wittenberg and P. Holmes. 'Scale and space localization in the Kuramoto–Sivashinsky equation'. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 9.2 (June 1999), pp. 452–465. ISSN: 1054-1500. DOI: 10.1063/1.166419.

[139] G. R. Wood and B. P. Zhang. "Estimation of the Lipschitz constant of a function". In: *Journal of Global Optimization* 8.1 (Jan. 1996), pp. 91–103. ISSN: 1573-2916. DOI: 10.1007/BF00229304.

[140] L. C. Woods. *Physics of Plasmas*. Wiley-VCH, Dec. 2003. ISBN: 978-3-527-40461-2. DOI: 10.1002/9783527618064.

[141] Y. Wu and the FDS Team. "Conceptual design and testing strategy of a dual functional lithium–lead test blanket module in ITER and EAST". In: *Nuclear Fusion* 47.11 (Oct. 2007), p. 1533. ISSN: 0029-5515. DOI: 10.1088/0029-5515/47/11/015.

[142] S. Yoden and M. Nomura. "Finite-Time Lyapunov Stability Analysis and Its Application to Atmospheric Predictability". In: *Journal of the Atmospheric Sciences* 50.11 (June 1993), pp. 1531–1543. ISSN: 0022-4928, 1520-0469. DOI: 10.1175/1520-0469(1993)050<1531:FTLSAA>2.0.CO;2.

[143] S. Zelik. *Attractors. Then and now*. Aug. 2022. DOI: 10.48550/arXiv.2208.12101. arXiv: 2208.12101v1.

[144] C. Ziehmann, L. A. Smith and J. Kurths. 'Localized Lyapunov exponents and the prediction of predictability'. In: *Physics Letters A* 271.4 (July 2000), pp. 237–251. ISSN: 0375-9601. DOI: 10.1016/S0375-9601(00)00336-4.