Time-parallelisation techniques with applications to plasma simulation

Federico Danieli
Trinity College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity term 2020/2021

Statement of Originality

Unless otherwise indicated, all the work in this thesis has been done by the author. Chapter 3 is currently under review for publication in the *SIAM Journal on Scientific Computing*. A manuscript illustrating the results in Chapter 4 is in preparation. Chapter 5 is under review for publication in the *Electronic Transaction on Numerical Analysis* journal. Chapter 6 has been published in the *Numerical Linear Algebra with Applications* journal [32].

Abstract

Parallel-in-time (PinT) methods have become an increasingly popular tool in the numerical solution of time-dependent Partial Differential Equations (PDEs), in light of their potential for extracting additional concurrency when spatial parallelism saturates. Particularly complex applications, such as the ones involving plasma simulations for fusion research, would benefit greatly from the implementation of such methods, given the large computational time usually necessary for the numerical solution of their governing equations.

In the first part of this thesis, we develop the concept of Space-Time Block Preconditioning (STBP), a technique for simplifying the task of time-parallelising the solution of a whole system of PDEs by reducing it to that of inverting space-time operators acting on single variables only, which are more amenable to be tackled by PinT algorithms. This technique borrows from block preconditioning strategies developed for spatial operators in the framework of time-stepping algorithms, and extends them to the whole space-time setting. After illustrating the guiding principles behind STBP, we measure its effectiveness with applications in fluid dynamics and magnetohydrodynamics. In both cases, the resulting preconditioners show excellent scaling properties with respect to mesh refinement, and their applications come at little-to-no overhead cost in terms of iterations to convergence with respect to their time-stepping counterpart, thus providing evidence for their potential parallel efficiency.

Ultimately, however, the parallel performance of a space-time block preconditioner depends on that of its internal components, which still rely on already-existing PinT methods. This might become an issue when dealing with advection-dominated or hyperbolic equations, as classical PinT algorithms have shown to struggle in this case. For this reason, the second part of this thesis is dedicated to investigating the causes behind this degradation in performance, as well as to study more recent PinT approaches for overcoming this limitation.

Acknowledgements

My gratitude goes first and foremost to my academic supervisor, **Andrew Wathen**: even after working with Andy for so long, I have barely managed to scratch the surface of his intuition of linear algebra, and I can only hope to make the best out of his teaching. Close second is my industrial supervisor, **Debasmita Samaddar**: Debbie's enthusiasm, support, and knack for organisation are qualities I will always aspire to.

My Viva experience was made particularly enjoyable by the examiners, **Matthias Bolten** and **Patrick Farrell**. The discussion ensued during the final exam and their valuable input contributed greatly towards the quality of this thesis, as well as in directing future work.

Also deserving my sincerest thanks are my collaborators: **Ben Southworth**, who has been an inexhaustible source of interesting ideas, and **Scott MacLachlan**, whose guidance and supervision have been priceless. Special thanks go to the people who made InFoMM possible, too: **Chris**, **Colin**, **Jonathan**, **Laura**, **Amanda**, and **Sarah**.

And then there are all the people who accompanied me in this experience, making it one of the most memorable. First of all, Cohort 3: a big hug goes to **Mel**, who will always be my favourite. Another one to **Ali**, **Ana**, and **Attila**, (and I am quite lucky their names come first alphabetically, so I can pretend that is the real reason why I put them there, and **Raquel** will not get offended), together with **Clint**, **Jonathan**, **Joe**, **Kris**, **Ollie**, **Scott**, and our undercover agent **Helen**. I want to thank **Matteo** for all the coffee, and apologise to the rest of my officemates, **Florian**, **Gonzalo** and **Nicolas**, for all the singing. Special thanks go to **Yifi**, whose support and affection have accompanied me throughout the hardest times of my DPhil. A big thank you also to **Anna**, **Giuseppe**, **Arno**, **Sam**, **Matilde**, **Lili** and **Diana**, who made my stay in Oxford and in Italy more lighthearted (especially during the pandemic), and to all the amazing people

I have engaged in interesting discussions with during my DPhil, and from whom I have learnt so much.

Finally, my parents, **Patrizia** e **Lucio**, and my brother, **Matteo**: they might never understand what this thesis is about, but for sure they understood all the effort that went into it.

Contents

Chapter 1

Introduction

In the framework of this project, we examine strategies to effectively employ parallel algorithms for the numerical solution of the equations describing the behaviour of plasma and electrically charged fluids in general. The particularity of the algorithms investigated lies in the way concurrency is achieved: the evolution of the solution is tracked simultaneously at different instants, owing to a process called *time parallelisation*.

We begin by providing a general introduction to the two main topics treated in this thesis, namely time parellelisation and plasma simulation, and then proceed to define the principal goal of this research work.

1.1 Importance of parallel algorithms

The growing complexity of systems arising from the solution of *Partial Differential Equations* (PDEs) demands ever increasing computational power in order to tackle such problems numerically in a reasonable amount of time. Unfortunately, the frequency of computations that can be carried out on a single processor unit is limited (see Fig. 1.1), and represents a fundamental upper bound on the efficiency of serial algorithms. To overcome this limit, much attention has been directed towards the implementation of parallel algorithms, which provide an alternative to sheer computational power to speed up computations.

Some standard parallelisation methods involve the decomposition of the spatial domain of the target PDE into smaller subdomains, where some computations are carried out independently [147]. Within this framework, a certain amount of information must be exchanged between the subdomains, so that the solution could be matched at the interfaces, usually in an iterative fashion. However, this causes additional overhead with respect to solving the PDE serially, so that increasing the number of

subdomains above a certain limit becomes detrimental to the purpose of speeding up the computation, and spatial parallelisation is said to *saturate*. Moreover, systems of *Initial Value Problems* (IVPs) for *Ordinary Differential Equations* (ODEs) do not present a spatial domain to subdivide in the first place, preventing the use of such techniques. Indeed, the temporal evolution of IVPs (for both ODEs and PDEs) is classically resolved via *time-stepping*, where the solution at each instant is recovered using information from the previous ones, in an inherently sequential fashion.

Nonetheless, the temporal domain is still available for parallelisation, and one can think of applying similar methodologies as for the spatial domain in order to concurrently recover the solution at various instants. Possibly the first person behind this intuition is Nievergelt, who speculated on such opportunity as early as in 1964 [110] and proposed a scheme that eventually gave birth to a whole new category of techniques for the solution of IVPs, namely *multiple shooting methods*. The additional challenge of parallelising over the temporal domain (rather than over the spatial one), stems from the causality principle: information flows forward in time, so it might seem counterintuitive to apply a parallel algorithm to a phenomenon which is sequential in nature. This notwithstanding, in recent years a renewed interest sparked the research on techniques for time parallelisation [55], brought forth both by the

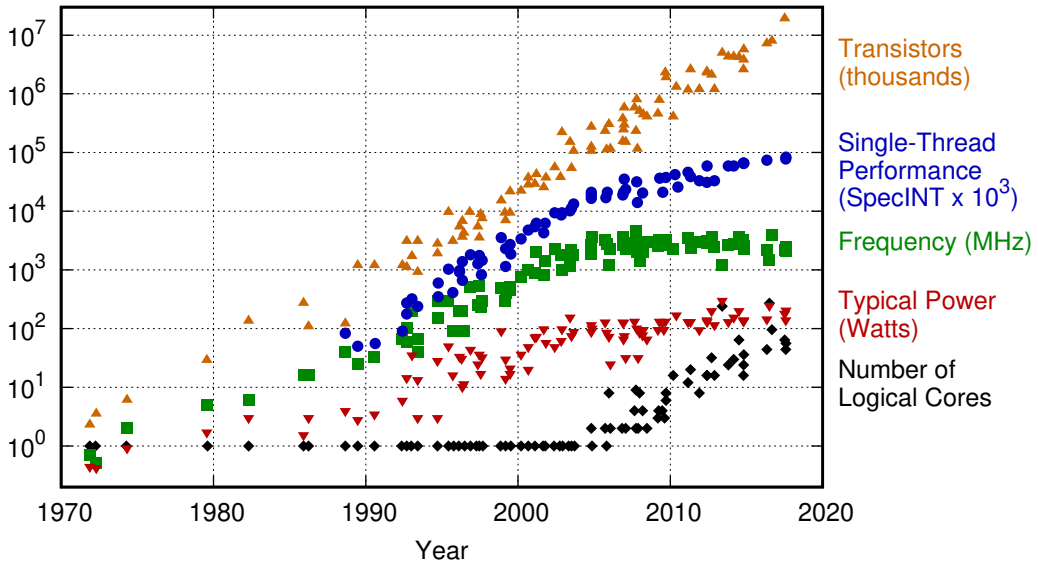


Figure 1.1: Evolution of microprocessors performance. In recent years, while the clock frequency of a single processor (green squares) has plateaued, the number of cores per processors (black diamonds) has been steadily increasing, which reflects the focus on parallelisation over power. Source: [122, Creative Commons Attribution 4.0 International Public License]

increased availability of parallel computational power (with exascale computing being the next target at the time of writing [36]), and by the development of *Parareal* [92], which represents the first modern parallel-in-time (PinT) algorithm.

1.2 Relevance of parallel computing in plasma simulations

Plasma is one of the fundamental states of matter, and can be considered as an *ionised* fluid, whose charges are relatively free to move independently [66]. As a consequence, its dynamics can internally generate electromagnetic fields, and is subject to interactions with externally imposed ones. Plasma plays a fundamental role in fusion energy research, since the conditions at which fusion occurs require the nuclear fuel to be in this particular state, at least for the standard methods in which fusion is achieved nowadays. Designing a stable, efficient fusion reactor would represent a major step towards solving the problem of generating clean energy from a durable and sustainable source. However, one of the greatest challenges in this endeavour consists in controlling the behaviour of the plasma within the reactor in an effective manner.

One way to achieve this is by imposing strong magnetic fields, which force the plasma into specific trajectories, in what is called *magnetic confinement*. Unfortunately, confined plasma often shows turbulent behaviour, characterised by the build-up of instabilities [106, Chap. 6-7 and 12-13]. It is precisely the growth of these instabilities that poses the main obstacle towards effective confinement, as these eventually resolve in sudden bursts occurring at the outer region of the plasma (namely *Edge-Localised Modes*, or ELMs). Therefore, controlling them is of utmost importance: if left unchecked, in fact, they can cause damage to the interior wall of the reactor, with consequent release of impurities within the plasma, and a drastic reduction in the temperature of the core (see Fig. 1.2), which might end up stopping the fusion reaction altogether.

In order to design an effective magnetic confinement, fusion researchers resort in large measure to numerical simulations. Given the complexity of the PDEs describing plasma behaviour, however, recovering their approximate solutions is a computationally demanding task, which needs parallel computing to be completed in a timely fashion. For this purpose, time parallelisation represents an appealing technique to further reduce computational time once more classical, spatial parallelisation methods have reached saturation and capped their potential for speedup.

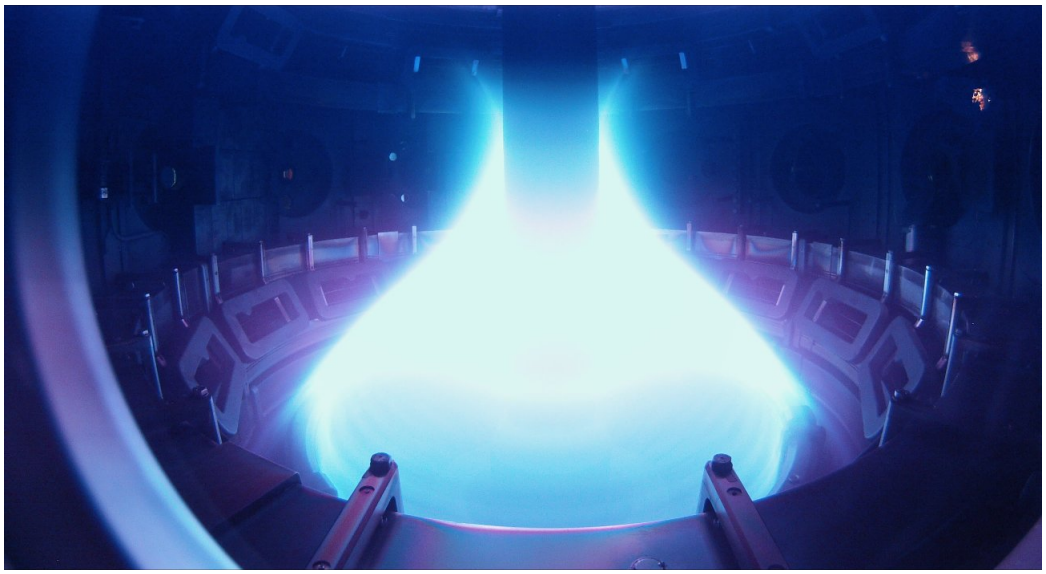


Figure 1.2: Picture of the *Mega Ampère Spherical Tokamak* (MAST) fusion reactor in Culham, UK. Incandescent plasma is filling the vessel, but it can only be seen at the bottom due to heat loss towards the reactor walls, which makes it radiate in the visible spectrum of light.

Research on PinT methods for plasma simulation remains however scarce. To make matters worse, in the most extreme regimes the relevant equations describing plasma behaviour are dominated by transport phenomena and, in the compressible limit, become hyperbolic in nature. This represents an additional complication to parallelisation in time, as most classical PinT algorithms tend to suffer from performance degradation when applied to this specific class of PDEs. Regardless, the recent successes from the applications of time parallelisation techniques to plasma simulation [119, 128, 129] justify the strong interest in further investigating the properties of such algorithms, on top of possible extensions and improvements, fine-tuned for the applications of interest.

1.3 Motivation, scope, and structure of the thesis

The motivation behind our work comes from the industrial partner and sponsor for this project, the *Culham Centre for Fusion Energy* (CCFE). CCFE is one of the leading research centres involved in the development of the technology necessary to attain a sustained fusion reaction, and the host to the *Joint European Torus* (JET) and the *Mega Ampère Spherical Tokamak* (MAST), two of the most advanced experimental fusion reactors in the world [151]. The interest of CCFE in time-parallelisation tech-

niques stems from the discussion outlined in Sec. 1.2, and in the scope of our work we investigate the applicability of PinT methods to the acceleration of the numerical solution of plasma simulations. This defines our research purpose.

> *The work conducted in this thesis aims at developing effective time-parallelisation techniques with the potential of speeding-up the numerical solution of complex systems of PDEs, specifically of models which are of relevance to the field of magnetohydrodynamics predicting plasma behaviour. In addition, it investigates the causes behind the loss of performance of available PinT algorithms when applied to the solution of hyperbolic PDEs, together with possible alternatives to overcome these limitations.*

The remainder of the thesis is structured as follows.

Chapter 2 acts as an introduction to the field of parallelisation-in-time. There, we list some of the most renowned time-parallel algorithms developed in recent years, with the goal of providing the reader with an overview of state-of-the-art PinT methods. In addition to a general description of the schemes selected, we illustrate their applicability and potential for parallel efficiency by reporting some results available in the literature. In particular, we point out the difficulties some of these methods experience in their application for the solution of hyperbolic PDEs.

In Chap. 3, we introduce the novel concept of *Space-Time Block Preconditioning* (STBP), which represents the main technique developed in the framework of our work. This considers existing block preconditioning strategies, which have been studied for the acceleration of the solution of systems arising from the *spatial* discretisation of a target multi-variable PDEs, and extends them to tackle its simultaneous *space-time* discretisation. The main advantage of this approach stems from the fact that it renders the target system more prone to be time-parallelised using already-existing PinT methods for single-variable PDEs. To illustrate the guiding principles behind this strategy, we consider applications to incompressible fluid dynamics, and measure the effectiveness of our approach for several test-cases, with the goal of testing its potential for parallelisation.

The target application in our research is introduced in Chap. 4, where we expand upon the work conducted in Chap. 3, to consider model problems in magnetohydrodynamics (MHD). After a brief introduction to the relevant system of PDEs, we showcase the flexibility of our STBP strategy by designing a space-time block preconditioner for the more complex MHD system, following similar principles as in

the previous chapter. Even for this problem, we investigate the performance of the preconditioner when used to accelerate the solution of a few test-cases, aiming at evaluating its applicability for the solution of plasma simulations.

As the success of our STBP approach is ultimately reliant on the parallel effectiveness of PinT algorithms in accelerating the solution of single-variable PDEs, with Chap. 5 the focus of this thesis shifts to investigating the application of time-parallelisation methods to the challenging class of hyperbolic PDEs. The time-parallel scheme we investigate in this chapter is the MGRIT algorithm, whose performance we measure in its application to some nonlinear conservation laws. The main purpose is to shed some light on the causes behind the degradation in convergence speed observed when MGRIT is used for this class of problems.

An alternative approach for the time-parallel solution of hyperbolic problems is investigated in Chap. 6. There, we consider the use of a *circulant preconditioner* for accelerating the solution of monolithic space-time systems arising from the discretisation of linear, constant-coefficient wave equations. We provide theoretical and experimental justifications for the effectiveness of this preconditioner, which supports its applicability for the solution of this type of PDEs.

Our results are finally summarised in Chap. 7, where we also provide a list of possibilities to further expand the work in this thesis, as well as some open-ended questions which might represent interesting topics for future research.

Chapter 2

Time parallelisation

In this chapter, we present some of the most renowned modern algorithms for time parallelisation available in the literature. The methods described here have been chosen with the aim to provide a reasonable sample of the variety of guiding concepts that can inspire the design of PinT methods.

In order to better organise this exposition, we choose to subdivide the algorithms illustrated here into two groups, loosely based on the framework in which they operate. The schemes belonging to the first group, described in Sec. 2.1, consider a given time integration algorithm, and attempt to achieve parallelisation in time by introducing concurrency into its application; on the other hand, the schemes belonging to the second group, described in Sec. 2.2, consider the system arising from the simultaneous (space-)time discretisation of a differential equation, and focus on providing a parallel method for the solution of the whole system at once. We point out that this distinction is, to some extent, arbitrary, and indeed some of the methods listed here have often been categorised differently in the literature; nonetheless, we find it useful to operate this separation, in that it highlights two substantially different frameworks in which to attempt to achieve parallelisation in time.

As a caveat, the list presented here is by no means exhaustive, and it reports only a fraction of the different manners in which time parallelism can be achieved. For a broader overview, we refer to the excellent chapter on the topic by Gander, [55], which describes the evolution of PinT methods since their inception. A more recent review by Ong and Schroder, collecting performance results for time-parallelisation algorithms, can be found in [111].

2.1 Parallelisation of time-integration schemes

In aiding our presentation of this first class of PinT schemes, we introduce the generic target ODE,

$$\begin{cases} u'(t) = f\left(u(t), t\right) & t \in (T_0, T] \\ u(T_0) = \bar{u}^0, \end{cases} \tag{2.1}$$

whose integration in time we are interested in conducting in parallel. Here, $\bar{u}^0$ represents the initial condition, while the right-hand side $f : \mathbb{R} \times (T_0, T] \to \mathbb{R}$ is a given Lipschitz continuous function. We point out that in this introduction we focus on a scalar ODE purely for the purpose of simplifying our description, but the methods listed here can be seamlessly extended to systems thereof, including (but not limited to) PDEs discretised using a *method-of-lines* approach [130].

2.1.1 Parareal

The *Parareal* algorithm is one of the most famous PinT methods, also owing to the simplicity of its implementation. Its original formulation, which we follow here to provide a description of the scheme, was originally presented in [92] by Lions, Maday, and Turinici.

To apply the algorithm, first the temporal domain of (2.1) is split into N_p subdomains (or *time chunks*), $[T_p, T_{p+1}]$, for $p = 0, \dots, N_p - 1$, of size $\Delta T_p = T_{p+1} - T_p$ and with $T_{N_p} = T$. Parallelisation is then achieved by solving independently in each subdomain the Cauchy problems

$$\begin{cases} u_p'(t) = f(u_p(t), t) & t \in (T_p, T_{p+1}] \\ u_p(T_p) = \lambda_p \end{cases} \qquad p = 0, \dots, N_p - 1. \tag{2.2}$$

We immediately realise the main difficulty we need to overcome for this approach to be effective: the initial condition is only available for the first time chunk, $\lambda_0 = \bar{u}^0$, and to collect the others exactly we would need to integrate sequentially each of the (2.2) and impose $\lambda_p = u_{p-1}(T_p)$, thus reducing the problem to the serial case. To obviate this inconvenience, Parareal works in an iterative fashion: a guess is provided for the initial conditions λ_p^k necessary to solve the problems (2.2), and this guess is refined at each iteration k.

This is achieved by introducing two numerical solvers (or *propagators*). Classically, these come in the form of time-stepping routines, but they can more generally be identified as functions that map an initial condition λ to an approximate solution of (2.2) at the end of the corresponding time chunk. The two solvers have different

Algorithm 1: Parareal

```
   /* Initialisation:                                                    */
1  Set λ_0^0 = ū^0 ;
2  for p = 0 to N_p − 2 do
3      Set λ_{p+1}^0 = G_p(λ_p^0);    // G is used to get an initial guess for λ_p^0

   /* Main Parareal iterations:                                          */
4  for k = 0 to It_max do
       /* Parallel step:                                                 */
5      parfor p = k to N_p − 1 do
6          Compute F_p(λ_p^k);
       /* Serial (update) step:                                          */
7      Set λ_{k+1}^{k+1} = F_k(λ_k^k);                    // Update (2.3a)
8      for p = k + 1 to N_p − 2 do
9          Compute G_p(λ_p^{k+1});
10         Update λ_{p+1}^{k+1} = F_p(λ_p^k) + G_p(λ_p^{k+1}) − G_p(λ_p^k);    // Update (2.3b)
```

properties: the *fine* one, $\mathcal{F}_p$, is more precise and expensive, while the *coarse* one, $\mathcal{G}_p$, is cheaper to compute but potentially less accurate[1]. The fine solver is employed to solve the N_p Cauchy problems (2.2) in parallel at the desired level of accuracy; the task of the coarse solver is instead to quickly propagate forward in time corrections to the eventual discrepancies between the guessed initial conditions λ_p^k and the actual results from the integration on the previous time chunk, $\mathcal{F}_{p-1}(\lambda_{p-1}^k)$. This is achieved by applying the following update formula for the initial conditions:

$$\begin{cases} \lambda_{k+1}^{k+1} = \mathcal{F}_k(\lambda_k^k) & \text{(2.3a)} \\ \lambda_{p+1}^{k+1} = \mathcal{F}_p(\lambda_p^k) + \mathcal{G}_p(\lambda_p^{k+1}) - \mathcal{G}_p(\lambda_p^k), \quad p = k+1, \dots, N_p - 2, & \text{(2.3b)} \end{cases}$$

where an initial guess is given, *e.g.*, by a first serial application of the coarse solver $\lambda_p^0 = \mathcal{G}_p(\lambda_{p-1}^0)$, starting from the available initial condition $\lambda_0^0 = \bar{u}^0$. Pseudo-code for the corresponding algorithm is provided in Alg. 1, while a sketch of its behaviour is given in Fig. 2.1.

From (2.3a), we can see that at each iteration k, the initial condition for the $(k+1)$-th subdomain λ_{k+1} becomes exact and does not need to be updated any more. This reflects the fact that information is propagating forward in time and, by that iteration, the applications of the fine solver have caught up with that time chunk.

[1]The subscripts p remind us that the action of each solver can in general depend on the time chunk p we are integrating on, in particular if the right-hand side of (2.1) is time-dependent.

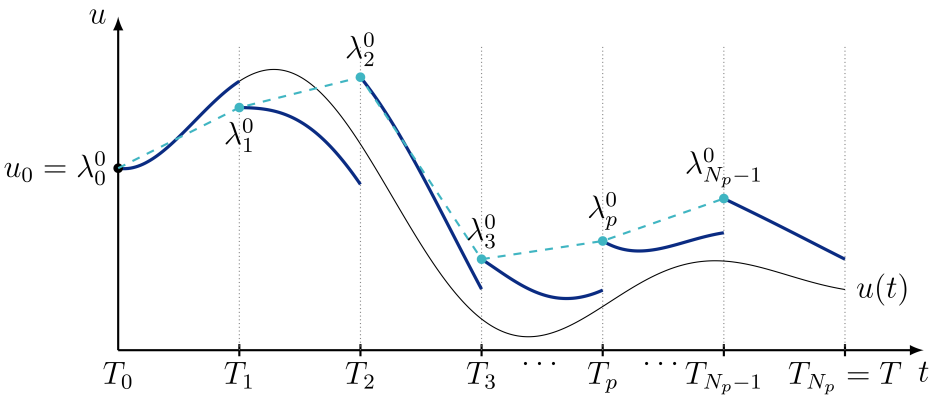


Figure 2.1: Example of the first iteration of the Parareal algorithm. To parallelise the computation of the solution $u(t)$ to (2.1) (black curve), we begin by splitting the domain into time chunks. An initial guess for the values of the solution at the beginning of each time chunk, λ_p^0, is collected using a first serial sweep of the coarse solver $\mathcal{G}$ (light blue), starting from the known value u_0. Using $\mathcal{F}$, integration is then conducted in parallel within each time subdomain (dark blue). At each iteration k, the k-th time chunk has achieved convergence. To update the other initial conditions, we perform a serial sweep of the coarse solver and sequentially apply formula (2.3b).

Indeed, after N_p iterations, the algorithm always converges to the solution one would obtain by serially applying the fine solver; for there to be any gain in employing Parareal, though, we should request the number of iterations to be much smaller than N_p. Moreover, (2.3b) states that the update needs to be carried out serially, since λ_{p+1}^{k+1} depends explicitly on λ_p^{k+1}; this, together with the fact that at least one fine integration needs to be performed, effectively caps the maximum parallel efficiency of the algorithm (intended as the ratio between time-to-solution in the serial and parallel case, divided by number of processors used). An alternative approach stemming from the attempt of improving upon this limitation is described in Sec. 2.1.2.

Since the inception of the algorithm, a number of interpretations have been given to the update formula (2.3). For example, following [54], consider the system

$$\begin{bmatrix} \lambda_0 \\ \lambda_1 - \mathcal{F}_0(\lambda_0) \\ \lambda_2 - \mathcal{F}_1(\lambda_1) \\ \vdots \\ \lambda_{N_p-1} - \mathcal{F}_{N_p-2}(\lambda_{N_p-2}) \end{bmatrix} = \begin{bmatrix} \bar{u}^0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.4}$$

Solving this system means finding the correct initial conditions λ_p so to match the result from the integration over the previous time chunk, $\mathcal{F}_p(\lambda_{p-1})$, which is the goal of the Parareal algorithm. Applying Newton iterations for solving (2.4) would give

rise to the following iterative update:

$$\begin{cases} \lambda_{k+1}^{k+1} = \mathcal{F}_k(\lambda_k^k) & \text{(2.5a)} \\ \lambda_{p+1}^{k+1} = \mathcal{F}_p(\lambda_p^k) + \left.\dfrac{\partial \mathcal{F}_p(\lambda)}{\partial \lambda}\right|_{\lambda=\lambda_p^k} \left(\lambda_p^{k+1} - \lambda_p^k\right), \quad p = k+1, \ldots, N_p - 2, & \text{(2.5b)} \end{cases}$$

where $\partial \mathcal{F}_p(\lambda)/\partial \lambda$ represents the sensitivity of the result from the p-th integrator with respect to variations in initial conditions. Comparing (2.5) with (2.3), we can view the application of the coarse solver as a way to provide an approximation for the sensitivities, which would classify Parareal as a quasi-Newton method [34, Chap. 6]. Similar in spirit is the interpretation provided in [96], which casts Parareal as a gradient descent algorithm applied to an optimal control problem, whose objective function directly relates to the norm of the residual in (2.4), or the one in [62], which sees the algorithm as a multiple shooting technique. Alternatively, in [96], as well as in [124], the algorithm is instead described as a preconditioned iterative method. All these different views on Parareal provide a varied framework for analysing its characteristics, which we proceed to describe in the following section.

2.1.1.1 Stability and issues with hyperbolic equations

Parareal is a relatively mature algorithm, whose properties have been studied extensively over the course of the last two decades. In this section we focus on reporting some results from the literature regarding the stability of the method, as well as on discussing the performance of the algorithm when applied to different classes of PDEs.

To this purpose, we introduce a convergence result presented in [62], recovered for the Dahlquist equation[2] ($f(u(t), t) = zu(t)$ in (2.1), with $z \in \mathbb{C}$), and which relies on one-step time-stepping algorithms as fine and coarse solvers to be used in the update formula (2.3). For simplicity, we consider time chunks of uniform size $\Delta T_p \equiv \Delta T$, so that $\mathcal{F}_p \equiv \mathcal{F}$ and $\mathcal{G}_p \equiv \mathcal{G}$ for each $p = 0, \ldots, N_p - 1$. Notice that in this simplified framework we can express the application of the solvers as multiplication by their corresponding *stability functions* [71, Chap. IV.3]: $\mathcal{G}u = \Phi_{\mathcal{G}}(z\Delta t/N_{\mathcal{G}})^{N_{\mathcal{G}}} u$, and $\mathcal{F}u = \Phi_{\mathcal{F}}(z\Delta t/N_{\mathcal{F}})^{N_{\mathcal{F}}} u$, where $N_{\mathcal{F}}$ and $N_{\mathcal{G}}$ are the number of time-steps per time chunk for fine and coarse solver, respectively. We also require that both solvers act on their region of absolute stability, given by $|\mathcal{F}| < 1$ and $|\mathcal{G}| < 1$. The bound is on the infinity norm of the error of the k-th Parareal iteration with respect to the result

[2] A somewhat more general framework for the convergence of Parareal is provided in [5], but we found the bound presented here better suited for describing the characteristics of Parareal outlined in this section.

from the serial application of $\mathcal{F}$, $\|\boldsymbol{e}^k\|_\infty = \max\{\mathcal{F}\bar{u}^0 - \lambda_1^k, \dots, \mathcal{F}^{N_p-1}\bar{u}^0 - \lambda_{N_p-1}^k\}$, and it reads

$$\|\boldsymbol{e}^k\|_\infty \leq |\mathcal{G} - \mathcal{F}|^k \min\left\{\binom{N_p-1}{k}, \left(\frac{1-|\mathcal{G}|^{N_p-1}}{1-|\mathcal{G}|}\right)^k\right\}\|\boldsymbol{e}^0\|_\infty. \tag{2.6}$$

The binomial coefficient reminds us once again that the algorithm always converges in a finite number of steps; moreover, it does so superlinearly as k increases. However, since our focus is on the regime $N_p \gg 1$ and $k \ll N_p$, to achieve fast convergence we need to ensure that

$$\frac{|\mathcal{G} - \mathcal{F}|}{1 - |\mathcal{G}|} < 1, \tag{2.7}$$

as to guarantee that the error decreases at each iteration. Loosely speaking, this condition suggests that the coarse solver should be at the same time dissipative enough (in order to keep the denominator reasonably large), and a good approximation of the fine solver (so that the numerator remains reasonably small). This is still feasible when the dynamics described by the target differential equation incorporate some form of smoothing, and indeed Parareal has proven to be effective in accelerating the solution of parabolic problems. Examples of successful applications for this class of PDEs are reported in [98] with a theoretical speedup with respect to sequential time-stepping of 22× using 50 cores, in [86] (7× speedup for 32 cores), in [20] (300× speedup with 1600 cores), and in [6] (6.25× speedup for 50 cores). When moving to transport-dominated equations, however, one can expect that ensuring (2.7) might become harder. In the case of pure transport, in fact, accurate solvers should preserve the energy within the system: as a consequence, the conditions on the coarse solver become even tighter, since this cannot be too dissipative.

It is possible to illustrate the difference in performance of Parareal when applied to a parabolic or hyperbolic problem with the example PDE

$$\begin{cases} \dfrac{\partial u}{\partial t}(x,t) = -v\dfrac{\partial u}{\partial x}(x,t) + \mu\dfrac{\partial^2 u}{\partial x^2}(x,t) & (x,t) \in [0,L] \times (0,T] \\ u(x,0) = \dfrac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} & x \in [0,L] \\ u(0,t) = u(L,t) & t \in (0,T] \end{cases}, \tag{2.8}$$

where, by varying its parameters, one can tweak the nature of the system from purely advective ($\mu = 0$) to purely diffusive ($v = 0$). The resulting convergence behaviour of the algorithm is compared in Fig. 2.2. While for the parabolic case the error diminishes relatively quickly, we can see how it remains substantially unchanged for the hyperbolic case, up until the very last iteration, when it must converge. Even

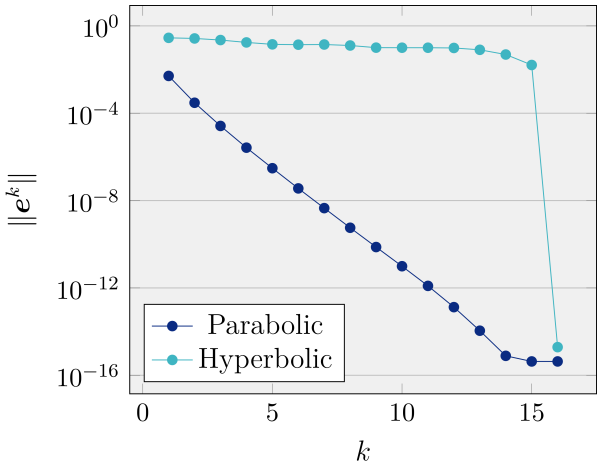


Figure 2.2: Evolution of the L^∞ norm of the error in both space and time for Parareal applied to (2.8), in the case $v = 0$, $\mu = 0.1$ (light blue), and $v = 1$, $\mu = 0$ (dark blue). Simulation parameters for both cases: *Implicit Euler* as coarse solver, exact fine solver, $N_p = T = 16$, $L = 2\pi$. Spectral discretisation in space with $N_x = 61$ modes.

worse scenarios, where the error spikes during the first $\sim N_p/2$ iterations to start decreasing only afterwards, are far from uncommon (see for example [62, Fig. 5.1]). Already in [143, Fig. 2 and 3] and [62, Fig. 4.1] it is illustrated how eigenvalues with large imaginary part tend to fall outside the stability region of Parareal for numerous choices of coarse/fine solvers, providing evidence for undermining the applicability of the vanilla version of this algorithm to systems where transport phenomena are dominant.

Investigating the causes for this instability, as well as identifying effective countermeasures, is in large part still a matter of research, and we address some of these issues in Chap. 5. The work conducted in [123,124] provides a useful intuition behind this undesirable behaviour, and connects it to the *local* nature of formula (2.3): when dealing with a vector unknown, (as it might arise from the spatial discretisation of a PDE), the update acts component-wise, so Parareal has no way to counteract errors stemming from potential relative shifts in the solution of the coarse solver with respect to that of the fine one. In solving hyperbolic problems, these occur whenever there is a mismatch in the wave propagation speed registered by the two solvers.

This notwithstanding, the literature presents a number of examples where Parareal is successful, even in the unfavourable transport-dominated case: for example, in [26], convergence of Parareal applied to Burgers' equation is preserved by projecting the solution back to an energy manifold after every iteration; in [38] the authors show the potential for Parareal in speeding up the solution to a hydrodynamic problem char-

acterised by a Reynolds number of $5 \cdot 10^4$; in [109], choosing a Roe-average Riemann solver seems to be key for fast convergence when solving the shallow-water equations, reaching a parallel efficiency of $\sim 35\%$. On top of this, numerous attempts at stabilising the algorithm are also reported, for example in [18, 19, 63], and are at the base of the development of alternative methods, such as the one presented in Sec. 2.1.3.

2.1.2 PFASST

The *Parallel Full Approximation Scheme in Space and Time* (PFASST) was first introduced by Emmett and Minion in [43]. Similarly to Parareal, this is an iterative method which employs solvers of different levels of accuracy in order to refine and propagate the solution along the time domain. In contrast to Parareal, however, the exchange of information between the levels is performed using a *Full Approximation Scheme* (FAS), a concept originally developed for multigrid techniques [12, Chap. 6]. Moreover, in its original formulation, the numerical integrators used at each level belong to the class of *Spectral Deferred Correction* (SDC) iterative methods [37], rather than being time-stepping schemes. The combination of SDC and FAS represents the core of the PFASST algorithm: the method in fact makes use of both techniques in order to propagate information about the evolution of the solution forward in time, as well as to refine it in parallel to an increased level of accuracy.

More in detail, as with the Parareal algorithm described in Sec. 2.1.1, the temporal domain of (2.1) is separated into N_p chunks (2.2), each assigned to a different processor. Within each chunk $p = 0, \dots, N_p - 1$, two different discretisations of the time domain are introduced, $\left\{t^G_{m,p}\right\}_{m=0}^{N_p^G-1} \subset [T_p, T_{p+1}] \supset \left\{t^F_{m,p}\right\}_{m=0}^{N_p^F-1}$, where $N_p^G < N_p^F$, so that the superscript G corresponds to the coarser level, and F to the finer one. We further request the first and last point of each time chunk to be included in these discretisations, *i.e.*, $t^G_{0,p} = t^F_{0,p} = T_p$ and $t^G_{N_p^G-1,p} = t^F_{N_p^F-1,p} = T_{p+1}$, $\forall p$. The two levels communicate between each other using a pair of interpolation/restriction operators, respectively denoted as I_G^F and R_F^G, which map solutions from coarse to fine level and vice-versa; an SDC solver is also associated to each level. For each main iteration k of PFASST, we define the approximate fine solution within the p-th time chunk at the i-th SDC sub-iteration as:

$$\tilde{\boldsymbol{u}}_p^{k,i} = \left[\tilde{u}^{k,i}_{0,p}, \dots, \tilde{u}^{k,i}_{N_p^F-1,p}\right]^T \approx \boldsymbol{u}_p = \left[u\left(t^F_{0,p}\right), \dots, u\left(t^F_{N_p^F-1,p}\right)\right]^T. \tag{2.9}$$

In complete analogy, the coarse-level approximate solution is denoted as:

$$\tilde{\boldsymbol{y}}_p^{k,i} = \left[\tilde{y}^{k,i}_{0,p}, \dots, \tilde{y}^{k,i}_{N_p^G-1,p}\right]^T \approx \boldsymbol{y}_p^{k} = \left[y^k\left(t^G_{0,p}\right), \dots, y^k\left(t^G_{N_p^G-1,p}\right)\right]^T. \tag{2.10}$$

Upon initialisation of the PFASST algorithm, the initial condition $\bar{u}^0$ is broadcast to all processors p and restricted to the coarse level, in order to build

$$\tilde{\boldsymbol{u}}_{p,IN} = \left[\bar{u}^0, \dots, \bar{u}^0\right]^T \quad \text{and} \quad \tilde{\boldsymbol{y}}^0_{p,IN} = R^G_F\left(\tilde{\boldsymbol{u}}_{p,IN}\right), \tag{2.11}$$

so to provide the initial guess for the coarse SDC solver. Here, the subscript $()_{,IN}$ identifies quantities computed during the initialisation phase. Then, in a similar fashion to the Parareal algorithm, the initial conditions are serially updated using the coarse SDC solver, and propagated forward. For each processor, this amounts to iteratively applying the SDC update a total of N^G_{SDC} times, and sending the last component of the recovered solution to the following processor, which will then substitute it to the initial component of its own solution before applying the coarse solver in its turn. Contrarily to Parareal, however, in PFASST the processors $p > 0$ do not need to remain idle while waiting for the updated initial conditions to reach them, and can instead start applying SDC sweeps straight away. Thus, at the end of the initialisation phase, each processor p has completed a total of $(p+1)N^G_{SDC}$ SDC iterations and hence recovered $\tilde{\boldsymbol{y}}^{(p+1)N^G_{SDC}}_{p,IN}$. The initialisation terminates by interpolating the coarse solution back to the fine level, providing the initial conditions for the fine SDC sweep:

$$\tilde{\boldsymbol{u}}^{0,0}_p = \tilde{\boldsymbol{u}}_{p,IN} + I^F_G\left(\tilde{\boldsymbol{y}}^{(p+1)N^G_{SDC}}_p - \tilde{\boldsymbol{y}}^0_p\right). \tag{2.12}$$

At each main iteration k of the PFASST algorithm, each processor independently performs a number N^F_{SDC} of SDC sweeps at the fine level, in order to refine the approximate solution to the target ODE. Updated information from the previous processor is then collected and restricted to the coarse level, where it is quickly propagated in time using the coarse SDC sweeps. The initial guess on the coarse solution is given by the restriction on the coarse grid of the current value of the approximate fine solution, but with the first component modified. This component is in fact recovered from the previous time-chunk:

$$\tilde{\boldsymbol{y}}^{k,0}_p = R^G_F\left(\left[\tilde{u}^{k+1,0}_{N^F_{p-1}-1,p-1}, \tilde{u}^{k,N^F_{SDC}}_{1,p}, \dots, \tilde{u}^{k,N^F_{SDC}}_{N^F_p-1,p}\right]^T\right). \tag{2.13}$$

Upon termination of the coarse SDC sweeps, the resulting correction is interpolated back to fine level and used to retrieve the starting point for the next PFASST iteration:

$$\tilde{\boldsymbol{u}}^{k+1,0}_p = \left[\tilde{u}^{k+1,0}_{N^F_{p-1}-1,p-1}, \tilde{u}^{k,N^F_{SDC}}_{1,p}, \dots, \tilde{u}^{k,N^F_{SDC}}_{N^F_p-1,p}\right]^T + I^F_G\left(\tilde{\boldsymbol{y}}^{k,N^G_{SDC}}_p - \tilde{\boldsymbol{y}}^{k,0}_p\right). \tag{2.14}$$

The iteration ends by sending the last component $\tilde{u}^{k+1,0}_{M^F_p-1,p}$ to the following time-chunk, before starting the new SDC sweeps at the fine level. The complete pseudo-code for the algorithm described is given in Alg. 2.

While this description is limited to the use of a pair of fine/coarse solvers, this framework can be extended to a complete hierarchy of discretisation grids. Even more generally, if we are solving PDEs rather than ODEs, the refinement level could vary in the space domain as well as in the time domain. In that case, some adjustments can be applied to the description of the algorithm in order to improve its speed, giving priority to the forward propagation of information, before performing the remaining tasks: that is to say, trying to reduce as much as possible the computational time between the `receive` and `send` instructions (lines 16 and 22 of Alg. 2 respectively).

Algorithm 2: PFASST

`/* Initialisation: */`

1 **parfor** $p = 0$ *to* $N_p - 1$ **do**

2 Broadcast initial conditions $\bar{u}^0$;

3 Restrict to coarse level and recover $\tilde{\boldsymbol{y}}^0_{p,IN}$; `// Formula (2.11)`

4 Apply pN^G_{SDC} coarse SDC sweeps to recover $\tilde{\boldsymbol{y}}^{pN^G_{SDC}}_{p,IN}$;

5 **if** $p > 0$ **then**

6 Receive the last component of $\tilde{\boldsymbol{y}}^{pN^G_{SDC}}_{p-1,IN}$ from processor $p-1$;

7 Substitute it to first component of $\tilde{\boldsymbol{y}}^{pN^G_{SDC}}_{p,IN}$;

8 Apply N^G_{SDC} additional SDC sweeps to recover $\tilde{\boldsymbol{y}}^{(p+1)N^G_{SDC}}_{p,IN}$;

9 **if** $p < N_p - 1$ **then**

10 Send last component of $\tilde{\boldsymbol{y}}^{(p+1)N^G_{SDC}}_{p,IN}$ to processor $p+1$;

11 Interpolate correction at fine level to recover $\tilde{\boldsymbol{u}}^{0,0}_p$; `// Formula (2.14)`

`/* Main PFASST iterations: */`

12 **parfor** $k = 0$ *to* $It_{\max}$ **do**

`/* Parallel step: */`

13 **for** $p = 0$ *to* $N_p - 1$ **do**

14 Apply N^F_{SDC} fine SDC sweeps to recover $\tilde{\boldsymbol{u}}^{k,N^F_{SDC}}_p$;

15 **if** $p > 0$ **then**

16 Receive the last component of $\tilde{\boldsymbol{u}}^{k+1,0}_{p-1}$ from processor $p-1$;

17 Substitute it to first component of $\tilde{\boldsymbol{u}}^{k,N^F_{SDC}}_p$;

18 Restrict fine solution to coarse and recover $\tilde{\boldsymbol{y}}^{k,0}_p$; `// Formula (2.13)`

19 Apply N^G_{SDC} coarse SDC sweeps to recover $\tilde{\boldsymbol{y}}^{k,N^G_{SDC}}_p$;

20 Interpolate correction at fine level to recover $\tilde{\boldsymbol{u}}^{k+1,0}_p$; `// Formula (2.14)`

21 **if** $p < N_p - 1$ **then**

22 Send last component of $\tilde{\boldsymbol{u}}^{k+1,0}_p$ to processor $p+1$;

For example, rather than interpolating the *whole* coarse correction to the fine level before sending the final conditions to the following processor, we can focus on recovering the sole final component (in time) of the fine solution, propagating that one forward, and only afterwards completing the interpolation of the remaining components, as presented in the original paper [43]. Furthermore, there is no need to limit the propagation of information to the fine grid only: each level can dialogue with the corresponding level of the following processor, sending updated information forward as soon as it becomes available [10, 142]. While this increases the amount of communication between processors, it might help reducing computational time, depending on the architecture of the system. Apart from these small but relevant adjustments, however, the core of the algorithm remains substantially unchanged.

Similarly to Parareal, also PFASST has proven to be effective in applications to parabolic PDEs: for example, in [104] a speedup of 6-9× with 24 cores is reported, while 8× is achieved in [43] with 16 cores. Unfortunately, though, not even PFASST is immune to instabilities in applications to transport-dominated problems, and often incurs in similar issues as those presented in Sec. 2.1.1.1. However, a recent result on linear convergence [11], (owing to an interpretation of the algorithm in a multigrid framework [10]), indicates that opportunely choosing certain parameters in the algorithm can help dampen these undesirable effects.

2.1.3 ParaExp

The *Parallel Exponential Propagation* (ParaExp) scheme represents another efficient alternative for parallel-in-time integration of linear differential equations. Introduced in [56] by Gander and Güttel, the algorithm has proven to be particularly effective on systems featuring a forcing term which is challenging to integrate numerically. Based on the principle of superposition of effects, it splits the system into an autonomous and a non-autonomous part, and employs exponential integration in order to rapidly solve the former. Further details are presented next, following the description originally given in [56].

The target equation for the ParaExp scheme is similar to (2.1), but the right-hand side is simplified to

$$f(u(t), t) = au(t) + g(t), \tag{2.15}$$

where a is a scalar[3]. Within this framework, the function $g(t)$ represents the main difficulty towards integration: this could be because it is rapidly oscillating, render-

[3]The description still holds in the multi-dimensional case, substituting a by a linear operator $\mathcal{A}$.

ing the system stiff, or simply because it is particularly expensive to evaluate. In complete analogy to the other time-parallel algorithms presented in Sec. 2.1.1 and in Sec. 2.1.2, the ParaExp algorithm achieves a speedup by dividing the time domain of the differential equation into chunks, each assigned to a different processor. Each of the processors solves then two types of problem. The first one is defined as:

$$\begin{cases} v_p'(t) = & a v_p(t) + g(t), \quad t \in (T_p, T_{p+1}] \\ v_p(T_p) = & \delta_{p,0}\, \bar{u}^0 \end{cases} \qquad p = 0, \ldots, N_p - 1, \tag{2.16}$$

where the Kronecker delta $\delta_{p,0}$ indicates that homogeneous initial conditions are picked for each subdomain, except the very first one, where the only known initial condition for the solution, $u(T_0) = \bar{u}^0$, is imposed. Due to the presence of $g(t)$, integrating these problems is expected to be time consuming; nonetheless, all of (2.16) are independent of each other, which hints at parallelisation as a valuable option to save computational time. The second problem is given by:

$$\begin{cases} w_p'(t) = & a w_p(t), \quad t \in (T_p, T] \\ w_p(T_p) = & v_{p-1}(T_p) \end{cases} \qquad p = 1, \ldots, N_p - 1. \tag{2.17}$$

Similarly, all of (2.17) are also independent. However, in order to be solved, they require $v_{p-1}(T_p)$, *i.e.* the result from the integration of the corresponding sub-problem (2.16). Moreover, notice that this integration does not stop at the end of a time chunk, but continues until the end of the time domain T, which causes an imbalance in the amount of computation that each processor needs to perform. This issue can be addressed by adjusting the sizes of each time chunk, if an estimate of the amount of computations necessary to solve both (2.16) and (2.17) is provided. Still, this might be seen as a large overhead with respect to the serial solution; however, each of the Cauchy problems in (2.17) is prone to be solved efficiently using exponential integration, as described in [76].

Finally, once the problems (2.16) and (2.17) are solved, the complete solution at a given instant $t \in [T_p, T_{p+1}]$ is promptly recovered by superposition of effects:

$$u(t) = v_p(t) + \sum_{i=1}^{p} w_i(t). \tag{2.18}$$

An illustration of the actions performed by each processor is sketched in Fig. 2.3, while the pseudo-code for the algorithm is provided in Alg. 3.

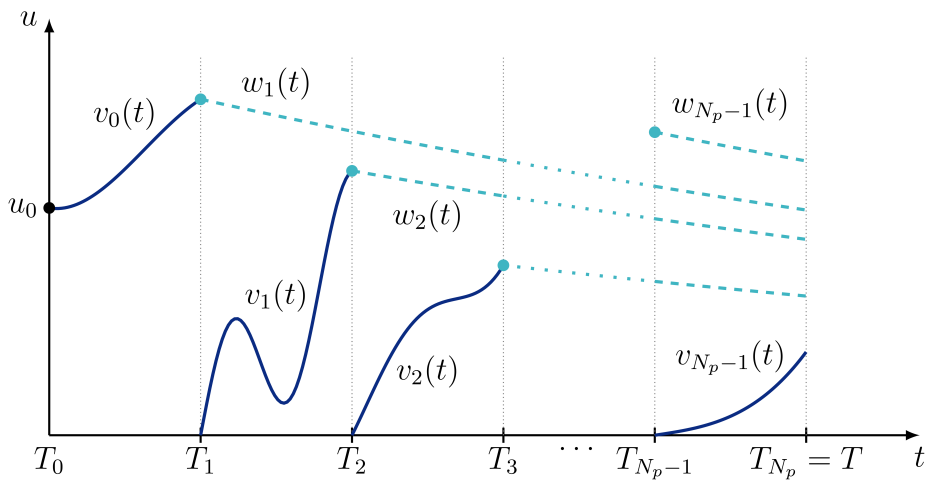


Figure 2.3: Schematic of the of the action of the ParaExp algorithm. Each processor p starts solving (2.16) on the corresponding time chunk, and recovers v_p (dark blue lines). This provides the initial condition (light blue dots), to proceed integrating (2.17) all the way until the end of the time domain (light blue dotted lines). Once $w_{p+1}(t)$ is also recovered, the complete solution can be attained simply summing all contributions.

2.1.3.1 Extension to nonlinear equations

If the linearity assumption breaks down, in general the principle of superposition of effects does not hold any more, and the procedure outlined in Alg. 3 fails to provide the correct solution to the target equation. However, one can hope to achieve convergence to the actual solution by iterating. In [57], this process is described and analysed in

Algorithm 3: Paraexp

```
   /* Initialisation:                                              */
1  Set v_0(T_0) = u_0;
2  parfor p = 1 to N_p − 1 do
3      Set v_p(T_p) = 0;

   /* Parallel steps:                                              */
4  parfor p = 0 to N_p − 1 do
5      Integrate v_p(t) in [T_p, T_{p+1}];          // Problem (2.16)
6      if p < N_p − 1 then
           /* Fast exponential integration:                        */
7          Set w_{p+1}(T_{p+1}) = v_p(T_{p+1});
8          Integrate w_{p+1}(t) in [T_{p+1}, T];    // Problem (2.17)
9  Recover the complete solution by superposition;  // Formula (2.18)
```

detail, giving rise to a ParaExp algorithm for nonlinear equations.

A different approach is instead pursued in [73], where the *Asymptotic Parareal* algorithm is introduced. This lies somewhat at the interface between the ParaExp and Parareal algorithms. It is intended for applications where the differential operator involved can be split into a linear and nonlinear part,

$$\frac{\partial u(t)}{\partial t} = \frac{1}{\varepsilon}\mathcal{L}u(t) + \mathcal{N}(u(t)), \tag{2.19}$$

whose linear component $\mathcal{L}$ encapsulates the effect of a transport phenomenon, while the nonlinear one $\mathcal{N}$ corresponds to diffusion. Moreover, it is assumed that advection occurs over a much smaller timescale (of order $\varepsilon \ll 1$) than diffusion. The main idea proposed in [73] is to employ a framework similar to that of Parareal, (alternating the use of a parallel fine integrator to recover an accurate solution and a serial coarse integrator to run an update), but one in which the coarse solver is chosen in such a way to exploit the separation of timescales. In particular, this consists of an exponential integrator that tracks exactly the evolution imposed by $\mathcal{L}$, while taking an appropriately averaged version of $\mathcal{N}$. It is possibly thanks to the very use of exponential integration that this method manages to circumvent many of the shortcomings of Parareal illustrated in Sec. 2.1.1.1: in fact, it allows to safely use a relatively large time-step for the coarse solver while still preserving an adequate level of accuracy, thus manufacturing a considerably more efficient algorithm: results in [73] show a speedup of 10× with respect to vanilla Parareal.

2.2 All-at-once methods

The schemes introduced in the previous sections employ an already existing algorithm for the integration of a time-dependent differential equation (be it a generic time-stepping routine for Parareal, SDC for PFASST, or exponential integration for ParaExp), which is left substantially unchanged, while focusing on implementing an overarching framework which allows for the introduction of concurrency in the recovery of the solution along the temporal domain. An alternative approach to time parallelisation consists of considering the monolithic system arising from the simultaneous space-time discretisation of a time-dependent partial differential equation, and developing techniques which tackle its solution concurrently as a whole, rather than proceeding instant-by-instant. In light of this, the class of algorithms following this strategy is often referred to as *all-at-once methods* (AAO)[4].

[4]We point out, however, that this term has been used with slightly different meanings throughout the literature.

All the techniques developed and studied in this thesis belong to this second class of PinT methods. In this section, we introduce the main ideas behind this approach, and we briefly illustrate a choice of AAO algorithms. As with the previous section, we introduce a model problem to help us in their description, and consider a generic time-dependent, linear, homogeneous PDE with constant coefficients, discretised in time using *Implicit Euler*. Temporal integration is performed by solving the following recurrence formula:

$$\frac{\boldsymbol{u}^k - \boldsymbol{u}^{k-1}}{\Delta t_k} + \mathcal{L}\boldsymbol{u}^k = \boldsymbol{0}, \qquad k = 1, \dots, N_t, \tag{2.20}$$

where the instants t_k identify the temporal grid $\{t_k\}_{k=0}^{N_t}$, with $\Delta t_k = t_k - t_{k-1}$, and starting from a given initial condition $\boldsymbol{u}^0 = \bar{\boldsymbol{u}}^0$. In (2.20), the term $\mathcal{L} \in \mathbb{R}^{N_x \times N_x}$ represents a discretisation of a spatial operator, where N_x is the number of unknowns in the spatial discretisation. The set of equations (2.20) can be collected into a single discrete space-time system:

$$\underbrace{\begin{bmatrix} \frac{I}{\Delta t_1} + \mathcal{L} & & & \\ -\frac{I}{\Delta t_2} & \frac{I}{\Delta t_2} + \mathcal{L} & & \\ & \ddots & \ddots & \\ & & -\frac{I}{\Delta t_{N_t}} & \frac{I}{\Delta t_{N_t}} + \mathcal{L} \end{bmatrix}}_{=:A} \underbrace{\begin{bmatrix} \boldsymbol{u}^1 \\ \boldsymbol{u}^2 \\ \vdots \\ \boldsymbol{u}^{N_t} \end{bmatrix}}_{=:\boldsymbol{u}} = \underbrace{\begin{bmatrix} \frac{\bar{\boldsymbol{u}}^0}{\Delta t_1} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{bmatrix}}_{=:\boldsymbol{g}}. \tag{2.21}$$

All-at-once methods attempt to solve the whole system (2.21) at once, either in a direct or an iterative fashion, and with the goal of exposing concurrency in its solution. Some examples of how this can be achieved are described in the following section.

Remark 1. We expect that the type of temporal discretisation chosen will play a major role in defining the final structure of the resulting space-time system. Particularly, (2.21) presents the distinctive block lower-triangular structure stemming from employing classical time-stepping schemes[5]. In a sense, time-stepping algorithms *must* produce such a structure, since they recover the solution at a given instant by using information only coming from the previous instants. Classical time-stepping can then be interpreted as solving (2.21) via block forward substitution, making use of the fact that the space-time matrix stemming from this temporal discretisation retains a block lower-triangular form.

[5]More in detail, the number of steps employed in the time-stepping routine determines the number of block sub-diagonals in the monolithic system. Since we use a *one*-step method for our example, the resulting space-time matrix contains only *one* block sub-diagonal. More complex examples are treated in Chap. 6.

2.2.1 ParaDiag

First introduced in [97], with its properties being analysed in [59], the ParaDiag algorithm aims at recovering the solution of the space-time system (2.21) via block diagonalisation. As a consequence, each of the blocks composing the resulting block diagonal matrix represents an independent system, whose solution can be trivially parallelised. Following [97], this procedure can be better illustrated by considering the tensor product representation of (2.21). To this purpose, introducing the discrete time-differentiation operator

$$\Phi \coloneqq \begin{bmatrix} \frac{1}{\Delta t_1} & & & \\ -\frac{1}{\Delta t_2} & \frac{1}{\Delta t_2} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{\Delta t_{N_t}} & \frac{1}{\Delta t_{N_t}} \end{bmatrix} \in \mathbb{R}^{N_t \times N_t}, \tag{2.22}$$

we can express the space-time system (2.21) as

$$A = \Phi \otimes I_{N_x} + I_{N_t} \otimes \mathcal{L}, \tag{2.23}$$

where $\otimes$ represents the Kronecker product, while I_{N_t} and I_{N_x} are identity matrices of size the number of temporal (N_t) and spatial (N_x) nodes, respectively. Given how Φ in this case is lower triangular, its eigenvalues are given by the elements in its diagonal, $\lambda_k(\Phi) = 1/\Delta t_k$; the matrix is hence diagonalisable if and only if all the time-steps δt_k are different from one another:

$$\Delta t_i \neq \Delta t_j \quad \forall i, j \in 1, \dots, N_t \quad \iff \quad \Phi = U \Lambda_\Phi U^{-1}, \quad \text{with } \Lambda_\Phi \coloneqq \operatorname*{diag}_{k=1,\dots,N_t} (\lambda_k(\Phi)), \tag{2.24}$$

where $U \in \mathbb{R}^{N_t \times N_t}$ denotes the matrix of eigenvectors of Φ. Under this assumption, the eigendecomposition of (2.22) can be recovered explicitly; moreover, U presents a lower-triangular structure [59, Thm. 3], making it relatively simple to invert via backward substitution. Enforcing diagonalisability, we can finally rewrite the space-time matrix A as

$$\begin{aligned} A &= (U \Lambda_\Phi U^{-1}) \otimes I_{N_x} + U U^{-1} \otimes \mathcal{L} \\ &= (U \otimes I_{N_x})(\Lambda_\Phi \otimes I_{N_x})(U^{-1} \otimes I_{N_x}) + (U \otimes I_{N_x})(I_{N_t} \otimes \mathcal{L})(U^{-1} \otimes I_{N_x}) \\ &= (U \otimes I_{N_x}) \underbrace{(\Lambda_\Phi \otimes I_{N_x} + I_{N_t} \otimes \mathcal{L})}_{= \operatorname{diag}_{k=1,\dots,N_t}(\lambda_k(\Phi) I_{N_x} + \mathcal{L})} (U^{-1} \otimes I_{N_x}), \end{aligned} \tag{2.25}$$

where we used the property of the Kronecker product $(ac) \otimes (bd) = (a \otimes b)(c \otimes d)$. Looking at the central factor in the formula above, we can see that the solution of

(2.21) reduces to the inversion of N_t independent spatial systems involving operators $\lambda_k(\Phi) I_{N_x} + \mathcal{L}$, which is what gives concurrency to this method. This procedure has been extended to applications with time-dependent parameters (*i.e.*, where the main operator $\mathcal{L}$ itself might vary with time), as well as to the nonlinear setting [58]. Interestingly, this method can be successfully applied also to the solution of wave equations: some results on the parallel capabilities of this method are reported in [60], where it is often reported a parallel efficiency of up to $\sim 90\%$.

Having to request irregular time-steps, as per (2.24), can be beneficial in specific frameworks[6], but is somewhat limiting in general, especially when considering that the condition number of the eigenvector matrix U is directly affected by the distribution of temporal nodes. However, work is under way in order to overcome this difficulty [60]. An alternative approach when uniform time-steps are employed instead is described in the following.

2.2.2 Space-time circulant preconditioning

Rather than using a direct solver to tackle (2.21), as done in Sec. 2.2.1, the method introduced in this section relies on Krylov-subspace iterative schemes [126], opportunely preconditioned with an operator whose application can be parallelised in time. To describe how this is achieved, consider again the operator in (2.21): if a uniform grid is employed for the temporal discretisation, this simplifies to

$$A \underset{\left(\begin{smallmatrix}\text{if} & \Delta t_k \equiv \Delta t \\ & \forall k = 1, \dots, N_t\end{smallmatrix}\right)}{=} \begin{bmatrix} \frac{I}{\Delta t} + \mathcal{L} & & \\ -\frac{I}{\Delta t} & \frac{I}{\Delta t} + \mathcal{L} & \\ & \ddots & \ddots \end{bmatrix}. \tag{2.26}$$

We notice that its structure further specialises: its blocks are constant down each diagonal, rendering A a block Toeplitz matrix. The authors in [101] suggest then to use a block circulant preconditioner in order to accelerate the convergence of iterative methods applied to the whole (2.26).

The idea of using circulant matrices as preconditioners for Toeplitz systems dates back to 1986 [145], and has proven to be remarkably effective: see for example [17, Chap. 2.5], or [108]. Circulant matrices can be seen as "wrapped around" Toeplitz matrices, where elements on a diagonal are also repeated N diagonals away (with N

[6] One such example is the classical heat equation: when tending to steady state, the rate of change of the solution decreases with time, so it makes sense to coarsen the temporal grid as time progresses.

being the size of the matrix), in the form

$$C = \begin{bmatrix} c_0 & c_{N-1} & \dots & c_2 & c_1 \\ c_1 & c_0 & c_{N-1} & & c_2 \\ \vdots & c_1 & \ddots & \ddots & \vdots \\ c_{N-2} & & \ddots & & c_{N-1} \\ c_{N-1} & c_{N-2} & \dots & c_1 & c_0 \end{bmatrix}. \tag{2.27}$$

A particularly favourable property of circulant matrices, is that they can be diagonalised via Fourier transform: introducing the *discrete Fourier transform* operator,

$$[F_N]_{m,n} \coloneqq \frac{1}{\sqrt{N}} e^{-\frac{2\pi \mathrm{i}}{N} mn}, \qquad m, n = 0, \dots, N-1, \tag{2.28}$$

we have that

$$C = F_N \Lambda_C F_N^*, \qquad \text{with} \qquad \Lambda_C \coloneqq \operatorname*{diag}_{k=0,\dots,N-1} ([F_N [C]_{-,0}]_k), \tag{2.29}$$

where Λ_C is the diagonal matrix containing the Fourier transform of the first column of C on its diagonal (which also identifies the eigenvalues of C). Systems involving C can then be effectively solved via *Fast Fourier Transform* (FFT).

This framework can be seamlessly extended to block circulant matrices, and applied to accelerate the solution of block Toeplitz systems such as (2.26): the preconditioner then assumes a form similar to that of (2.25), and its solution can be once again trivially parallelised in time[7]. We refer to Chap. 6, where we extensively make use of this method, for a more in-depth description. On top of this, the effectiveness of circulant preconditioning applied to space-time systems has been studied in detail in [101] and [65], and more recently in [93]. There, strong and weak scaling properties of the algorithm are discussed. In particular, the wall-clock time required by the method for some test-cases remains substantially constant as the number of processors used increases to up to 64, providing evidence for the optimal scalability of the algorithm.

Finally, we point out that, for constant-coefficient, linear problems which prescribe periodicity in time, the resulting space-time matrix is often already block circulant, namely

$$A_C = \begin{bmatrix} \frac{I}{\Delta t} + \mathcal{L} & & -\frac{I}{\Delta t} \\ -\frac{I}{\Delta t} & \frac{I}{\Delta t} + \mathcal{L} & \\ & \ddots & \ddots \end{bmatrix}, \tag{2.30}$$

[7]Indeed the approaches illustrated in Sec. 2.2.1 and 2.2.2 present some similarities, to the point that circulant preconditioning for AAO systems is described as a *variant* of ParaDiag in [60]. Given how the focus of the latter is more on the *preconditioning* aspect, however, we believe it is worth it to draw a distinction.

and the considerations on its diagonalisability can be applied directly to the solution of A: this has been exploited in [87] to design an efficient time-parallel solver for time-periodic problems, reporting a potential speedup of 500× when using 50 processors.

2.2.3 MGRIT

The *Multigrid Reduction in Time* (MGRIT) algorithm also aims at parallelising the solution of the monolithic system (2.21), but, as suggested by its name, it follows a *Multigrid Reduction* approach (MGR, [51, 120]), with the particularity that its coarsening procedure acts along the temporal domain. First proposed in [47], MGRIT is not the only multigrid-inspired algorithm for time parallelisation (see for example [48,61,77]), but is one of the most mature and widely studied. To introduce the idea behind this algorithm, we follow much of the original description in [47].

In accordance with the MGR approach, the N_t nodes corresponding to the temporal discretisation are split into two sets: C, consisting of *coarse* nodes, and F, consisting of *fine* ones. Both the components of the monolithic space-time discretisation $\boldsymbol{u}$ and the coefficients of A in (2.21), are rearranged accordingly, so that the matrix system can be factorised in the following way:

$$A = \begin{bmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{bmatrix} = \begin{bmatrix} I_{N_F} & 0 \\ A_{CF}A_{FF}^{-1} & I_{N_C} \end{bmatrix} \begin{bmatrix} A_{FF} & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I_{N_F} & A_{FF}^{-1}A_{FC} \\ 0 & I_{N_C} \end{bmatrix} \tag{2.31}$$

provided A_{FF} is invertible. Here, $I_{(*)}$ denotes the identity matrix of size $(*)$, with N_C and N_F identifying the number of coarse and fine nodes, respectively, so that $N_C + N_F = N_t$. The block S in the block diagonal matrix represents the coarse-nodes Schur complement,

$$S \coloneqq A_{CC} - A_{CF}A_{FF}^{-1}A_{FC}, \tag{2.32}$$

i.e., the ideal Petrov-Galerkin coarse grid operator [12, Chap. 7.7.3]. This factorisation allows us to separate the solution of the system over the F and C nodes: we have in fact

$$A^{-1} = \underbrace{\begin{bmatrix} -A_{FF}^{-1}A_{FC} \\ I_{N_C} \end{bmatrix}}_{=:I_F^C} S^{-1} \underbrace{\begin{bmatrix} A_{CF}A_{FF}^{-1} & I_{N_C} \end{bmatrix}}_{=:R_C^F} + \underbrace{\begin{bmatrix} A_{FF}^{-1} & 0 \\ 0 & 0 \end{bmatrix}}_{=:A_{FF,0}^{-1}}, \tag{2.33}$$

where we introduced the *ideal* restriction/interpolation operators R_C^F / I_F^C between the coarse and fine level, for which $S = R_C^F A I_F^C$ holds. To conclude the MGR description,

we use (2.33) to more conveniently express the error propagation operator associated with an *exact* 2-level multigrid method:

$$E \coloneqq I - A^{-1}A = \underbrace{\left(I - I_F^C S^{-1} R_C^F A\right)}_{=:E_C} \underbrace{\left(I - A_{FF,0}^{-1} A\right)}_{=:E_F}, \tag{2.34}$$

where we highlighted the factors responsible for error relaxation at fine and coarse level, E_F and E_C, respectively. This suggests that an iterative multigrid method can be built by substituting the components in (2.34), (and specifically the restriction/interpolation operators and the Schur complement, R_C^F, I_F^C and S), with approximate ones ($\tilde{R}_C^F$, $\tilde{I}_F^C$ and $\tilde{S}$).

Interestingly, Parareal itself can be interpreted as a special case of MGRIT, as shown in [47]. In fact, by choosing the injection operator $\tilde{R}_C^F \coloneqq \begin{bmatrix} 0 & I_F^C \end{bmatrix}$ instead of ideal restriction, and by substituting the Schur complement S with a space-time operator $\tilde{A}$ which represents time-stepping over the coarse nodes only, it can be proven that Parareal corresponds to a 2-level MGR scheme which applies a semi-coarsening procedure along the temporal domain only. Its associated error propagator is given by

$$E_{Parareal} \coloneqq \left(I - I_F^C \tilde{A}^{-1} \tilde{R}_C^F A\right) \left(I - A_{FF,0}^{-1} A\right). \tag{2.35}$$

This way MGRIT provides the great advantage of establishing a framework for interpreting Parareal as a MGR method, thus allowing the possibility to extend the algorithm using concepts borrowed from the multigrid technology[8]. Particularly, this framework enables the use of a whole hierarchy of coarse grids (in a true *multi*-grid spirit), as well as different approaches to error relaxation. For example, in [47] it is recommended to substitute the F-relaxation factor E_F in (2.34) with the FCF-relaxation term

$$E_{FCF} \coloneqq \left(I - A_{FF,0}^{-1} A\right) \left(I - I_F^C S^{-1} R_C^F A\right) \left(I - A_{FF,0}^{-1} A\right). \tag{2.36}$$

To explore these details further, we refer to Chap. 5, where we consider applications of MGRIT, and particularly to Sec. 5.2, for a more in-depth discussion on the actual choice of its approximated components $\tilde{R}_C^F$, $\tilde{I}_F^C$ and $\tilde{S}$.

To get an idea of the performance of MGRIT, an example on PDE-constrained optimisation reports a 19× speedup for 256 cores [69]; a more advanced application in fluid-structure interaction [74] shows a speedup of 18× with 32 cores.

[8] A different multigrid interpretation of Parareal is given in [62], and does share some similarities with the approach presented in MGRIT, albeit the latter uses more standard operators in its description.

The characteristic of applying coarsening along the time domain only, specific of the original MGRIT algorithm, has the advantage of producing a non-intrusive, *black-box* solver which often squeezes additional (temporal) parallelisation out of an already-developed time-marching scheme, provided the latter allows for some flexibility in the time-step chosen: this makes it closer in spirit to Parareal and the other algorithms in Sec. 2.1. Indeed, thanks to their similarity, one can exploit some of the convergence results derived for Parareal also in the MGRIT framework (and vice-versa: theory developed for MGRIT can be applied to Parareal as well), see for example [35, 139, 140]. Regrettably, this also suggests that some of the *limitations* of Parareal, and specifically those described in Sec. 2.1.1.1 in applications to hyperbolic systems, carry over to MGRIT as well. We address some of these issues in Chap. 5, and an attempt to overcome them is presented in [33]. In practice, however, once the connection with MGR has been established, one has the freedom of applying its machinery to produce coarsening strategies involving both spatial and temporal domains at once. The advantages this ensues are highlighted for instance in [78]. An algorithm which heavily relies on this feature is AIR, discussed in the next section.

2.2.4 AIR

Rooted on the same MGR concept as MGRIT, the *Approximation to Ideal Restriction* (AIR) technology was originally developed in an attempt to design an effective method for applying the multigrid framework to nonsymmetric systems, and quickly established itself as an effective PinT algorithm.

When applying multigrid to elliptic operators, the focus generally lies on the interpolation operator: this must be designed so that it can effectively map the correction from the coarse grid into the range of the fine-level relaxation (in other words, $\tilde{I}_F^C$ should be chosen so that it does not perturb the residual at the coarse nodes after coarse grid correction, since F-relaxation can usually take care of dampening the residual on the fine nodes); once this is achieved, if the target operator of the multigrid routine is self-adjoint, then restriction is usually taken as the adjoint of interpolation: $\tilde{R}_C^F = (\tilde{I}_F^C)^T$. On the other hand, with AIR the focus shifts to the restriction operator (as the name of the algorithm suggests). The role of restriction is complementary to that of interpolation, in that it should help defining a coarse-grid correction procedure which dampens error components falling outside the range of $\tilde{I}_F^C$. In [99] the case is made that for nonsymmetric systems, choosing $\tilde{R}_C^F$ in a way that it covers this role effectively (thus approximating the action of R_C^F in (2.33)), is particularly relevant in order to ensure convergence.

To achieve this, an MGR method is introduced in [99] for (block) lower-triangular matrices, as they might arise from the discretisation of advection equations, which relies on approximating the action of R_C^F, (and particularly the A_{FF}^{-1} term appearing in its C block (2.33)), via *Neumann series*: this represents the nAIR version of this algorithm. Unfortunately, the effectiveness of the approximation heavily relies on the lower-triangular structure of the target matrix, so that if this is disrupted (for example by introducing a diffusive term), the performance of the method degrades quickly. To counteract this, an alternative approach based on a *local* approximation of the effect of R_C^F (aptly dubbed lAIR) has been proposed in [100], which shows much more robustness when applied to the whole spectrum from advection- to diffusion-dominated problems.

Remarkably, AIR can be successfully applied to the solution of space-time systems, once we realise that a d-dimensional time-dependent advection(-diffusion) equation can be interpreted as a $d+1$ dimensional one, where time is yet another direction for advection. Moreover, the AIR procedure has the additional advantage of not relying on the target space-time system having the typical structure outlined in (2.21), which stems from a time-stepping scheme. As such, it can be successfully applied also to alternative approaches to temporal discretisation, such as space-time Finite Elements. This is showcased in [138], which also demonstrates the optimal parallel efficiency of the method. We make use of this algorithm in Chap. 3.

In the continuation of this thesis, we decide to focus on algorithms belonging to the AAO class described in Sec. 2.2. This preference is guided both by the potential they have shown over most algorithms in Sec. 2.1 when dealing with hyperbolic equations, and by the fact that they introduce a flexible framework where to operate our analysis, namely that of multigrid methods and/or preconditioning. Indeed, the technique developed in the following chapter takes inspiration precisely from block preconditioning strategies, which we extend to the whole space-time setting.

Chapter 3

Space-time block preconditioning

As mentioned in Sec. 2.2.3 and 2.2.4, the application of multigrid technologies to the simultaneous solution of space-time systems has been a guiding principle in designing successful parallel-in-time techniques [48,61,77], often providing significantly more parallelism and faster time-to-solution than algorithms listed in Sec. 2.1, not least because this approach naturally allows for adaptive mesh refinement in both space and time [138]. To our knowledge, however, most of the development of space-time MG techniques has been limited to applications involving single-variable (advection-)diffusion equations, and no multigrid methods have been developed and proven effective on space-time systems of PDEs. Indeed, effectively handling simultaneously both the complications introduced by inter-variable coupling, as well as the purely advective term in one dimension representing a time derivative, is beyond the capabilities of the current state-of-the-art.

In this chapter, we take a novel approach and introduce the concept of *space-time block preconditioning* (STBP) for the all-at-once solution of block-structured systems of PDEs. The main idea behind this method relies on considering well-developed spatial block preconditioning techniques, already used in accelerating the solution of sequential time-stepping or steady-state problems, and extending their application to the whole space-time setting. The approach described in [113,144], where the authors propose a preconditioner for a time-dependent optimal control problem, is similar in spirit to our work, as it exploits the block structure of the system at the space-time level. However, to our knowledge, similar concepts have not been applied before for the purpose of time parallelisation.

To better deliver and crystallise our description, we focus on the Oseen equations for illustrating the guiding principle behind the assembly of our space-time block preconditioner. This target problem is chosen both for the relevance of developing efficient time-parallelisation techniques for applications to fluid dynamics, due to the

typically high cost associated with their solution (see for example [152] for a discussion on the need for time parallelism in this field for emerging architectures), and because it serves as a building block for the more complex magnetohydrodynamics problem analysed in Chap. 4. Indeed, we believe that the underlying principle of space-time block preconditioning can be extended to other systems of PDEs for which effective spatial block preconditioners have been developed for the single time-step case (or for its stationary counterpart).

The block separation introduced by our preconditioner allows for great freedom in the choice of the solver for the velocity block, making it straightforward to apply, *e.g.*, space-time multigrid for single-variable equations as in [48, 77, 138], thus overcoming the issue described in the first paragraph. The proposed method builds on the *Pressure Convection-Diffusion* (PCD) preconditioner described in [40], and we show that similar principles to the ones that justified the validity of PCD can be generalised to the space-time setting, resulting in a space-time block preconditioner whose application is highly parallelisable in time.

The remainder of this chapter is organised as follows. Section 3.1 introduces our target problem, together with notation that will be used throughout. A derivation of the space-time block preconditioner representing our main contribution is given in Sec. 3.2, together with multiple theoretical justifications for its formulation. In Sec. 3.4 we investigate the effectiveness of the proposed preconditioner, testing its performance on four classical models in incompressible flow. Results indicate perfect scalability in refinement of spatial and temporal mesh spacing, perfect scalability in nonlinear Picard iteration count when applied to a nonlinear Navier-Stokes problem, and minimal overhead in terms of number of preconditioner applications compared with sequential time-stepping.

The work in this chapter stems from a collaboration with Ben S. Southworth: the findings have been collected into a manuscript, which at the time of writing is under review for publication in the *SIAM Journal on Scientific Computing*. A pre-print version of this work can be found in [31].

3.1 Problem definition

This section introduces notation that is used throughout this chapter. Here we describe a time-dependent version of the Oseen equations, as well as the discrete operators arising from its finite element discretisation.

Our target system is

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t}(\boldsymbol{x},t) + (\boldsymbol{w}(\boldsymbol{x},t)\cdot\nabla)\boldsymbol{u}(\boldsymbol{x},t) \\ \qquad -\mu\nabla^2\boldsymbol{u}(\boldsymbol{x},t) + \nabla p(\boldsymbol{x},t) &= \boldsymbol{f}(\boldsymbol{x},t) \qquad \text{in } \Omega\times(T_0,T], \\ \qquad\qquad \nabla\cdot\boldsymbol{u}(\boldsymbol{x},t) &= 0 \end{cases} \tag{3.1}$$

defined on a spatial domain $\Omega \subset \mathbb{R}^d$. Its unknowns are the d-dimensional vector field $\boldsymbol{u}(\boldsymbol{x},t)$, and the scalar field $p(\boldsymbol{x},t)$, representing velocity and pressure, respectively; μ is the viscosity coefficient, which we consider constant on Ω. The vector function $\boldsymbol{f}(\boldsymbol{x},t)$ is a given forcing term, while $\boldsymbol{w}(\boldsymbol{x},t)$ represents a given d-dimensional advection field. The special case where $\boldsymbol{w}(\boldsymbol{x},t) \equiv 0$ gives rise to a time-dependent version of Stokes equations:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t}(\boldsymbol{x},t) - \mu\nabla^2\boldsymbol{u}(\boldsymbol{x},t) + \nabla p(\boldsymbol{x},t) &= \boldsymbol{f}(\boldsymbol{x},t) \qquad \text{in } \Omega\times(T_0,T]; \\ \qquad\qquad \nabla\cdot\boldsymbol{u}(\boldsymbol{x},t) &= 0 \end{cases} \tag{3.2}$$

While most of the analysis conducted in this chapter considers the more general problem (3.1), we will at times refer to (3.2) for some simplifications. Conversely, by introducing the nonlinearity $\boldsymbol{w}(\boldsymbol{x},t) \equiv \boldsymbol{u}(\boldsymbol{x},t)$ we produce the Navier-Stokes equations for incompressible flow

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t}(\boldsymbol{x},t) + (\boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla)\boldsymbol{u}(\boldsymbol{x},t) \\ \qquad -\mu\nabla^2\boldsymbol{u}(\boldsymbol{x},t) + \nabla p(\boldsymbol{x},t) &= \boldsymbol{f}(\boldsymbol{x},t) \qquad \text{in } \Omega\times(T_0,T]. \\ \qquad\qquad \nabla\cdot\boldsymbol{u}(\boldsymbol{x},t) &= 0 \end{cases} \tag{3.3}$$

The Oseen system can then be interpreted as a linearisation of Navier-Stokes, implying that the analysis conducted here can be extended to more general frameworks: this is addressed more in detail in Sec. 3.4.3.

In order to provide closure to the definition of the initial-boundary value problem, we also equip (3.1) with an initial condition

$$\boldsymbol{u}(\boldsymbol{x},T_0) = \bar{\boldsymbol{u}}^0(\boldsymbol{x}) \quad \text{on } \Omega, \tag{3.4}$$

and the boundary conditions[1] (BC)

$$\boldsymbol{u}(\boldsymbol{x},t) = \boldsymbol{u}_D(\boldsymbol{s},t) \qquad \text{on } \Gamma_D^{\boldsymbol{u}}\times(T_0,T], \quad \text{and} \tag{3.5a}$$

$$\mu\nabla\boldsymbol{u}(\boldsymbol{s},t)\,\boldsymbol{n}(\boldsymbol{s}) - p(\boldsymbol{s},t)\,\boldsymbol{n}(\boldsymbol{s}) = \boldsymbol{g}(\boldsymbol{s},t) \qquad \text{on } \Gamma_N^{\boldsymbol{u}}\times(T_0,T], \tag{3.5b}$$

[1] Choosing the Neumann boundary conditions in the form (3.5b) ignores the contribution from the $\mu(\nabla\boldsymbol{u})^T$ term in the fluid stress tensor. This simplification is dangerous, since it results in a violation of the *objectivity principle* of continuum mechanics [91]. However, this choice of BC remains unfortunately common when dealing with the Laplace form of the unsteady Stokes equations (3.2), as it results in a system where the velocity components are decoupled (and indeed it has been used in [40] as well).

where we split the boundary $\partial\Omega$ into its Dirichlet and Neumann parts, respectively, with $\partial\Omega = \Gamma_D^{\boldsymbol{u}} \cup \Gamma_N^{\boldsymbol{u}}$ and $\Gamma_D^{\boldsymbol{u}} \cap \Gamma_N^{\boldsymbol{u}} = \varnothing$. In (3.5), $\boldsymbol{n}(\boldsymbol{s}) \in \mathbb{R}^d$ represents the outward-pointing normal to the boundary, while $\boldsymbol{u}_D(\boldsymbol{s}, t)$, $\boldsymbol{g}(\boldsymbol{s}, t)$, and $\bar{\boldsymbol{u}}^0(\boldsymbol{x})$ are some known functions.

At any given instant $t \in [T_0, T]$, the velocity unknown lives in the functional space [118, Chap. 2.4]

$$\boldsymbol{u}(\cdot, t) \in V(\Omega) \equiv \left(H^1_{\Gamma_D^{\boldsymbol{u}}}(\Omega)\right)^d, \tag{3.6}$$

while for the pressure we have

$$p(\cdot, t) \in Q(\Omega) \equiv L^2(\Omega), \quad \text{or} \quad Q(\Omega) \equiv L^2_0(\Omega) = \left\{ q(\boldsymbol{x}) \in L^2(\Omega) : \int_\Omega q(\boldsymbol{x})\, d\boldsymbol{x} = 0 \right\}, \tag{3.7}$$

where the zero-mean valued counterpart, $L^2_0(\Omega)$, is chosen if Dirichlet boundary conditions are imposed everywhere on the boundary of the domain, $\Gamma_D \equiv \partial\Omega$. In either case, having introduced eqs. (3.6) and (3.7), we can provide the weak formulation [118, Chap. 17.2] of (3.1): we seek $\boldsymbol{u}(\boldsymbol{x}, t)$ and $p(\boldsymbol{x}, t)$ such that

$$\begin{cases} \dfrac{\partial}{\partial t}\langle \boldsymbol{u}, \boldsymbol{v}\rangle + \mu \langle \nabla \boldsymbol{u}, \nabla \boldsymbol{v}\rangle & & & \\ \quad + \langle (\boldsymbol{w}\cdot\nabla)\boldsymbol{u}, \boldsymbol{v}\rangle - \langle p, \nabla\cdot\boldsymbol{v}\rangle &=& \langle \boldsymbol{f}, \boldsymbol{v}\rangle + \displaystyle\int_{\Gamma_N^{\boldsymbol{u}}} \boldsymbol{g}\cdot\boldsymbol{v}, & \forall \boldsymbol{v}(\boldsymbol{x}) \in V(\Omega)\ , \\ \quad - \langle q, \nabla\cdot\boldsymbol{u}\rangle &=& 0, & \forall q(\boldsymbol{x}) \in Q(\Omega) \end{cases} \tag{3.8}$$

for each instant $t \in (T_0, T]$, where $\langle \boldsymbol{a}(\boldsymbol{x}), \boldsymbol{b}(\boldsymbol{x})\rangle \coloneqq \int_\Omega \boldsymbol{a}(\boldsymbol{x})\cdot\boldsymbol{b}(\boldsymbol{x})\, d\boldsymbol{x}$ (or $\int_\Omega a(\boldsymbol{x})b(\boldsymbol{x})\, d\boldsymbol{x}$ when scalar quantities are involved) denotes the spatial scalar product. Notice we dropped the variables' dependence on $\boldsymbol{x}$ and t in order to contract notation.

3.1.1 Space-time discretisation

For the sake of simplicity, we discretise the temporal derivative in (3.8) using implicit Euler. To this end, we introduce a uniformly-spaced grid, $\{t_k\}$, over the temporal domain, where

$$t_k = T_0 + k\Delta t, \qquad \text{with} \qquad \Delta t = (T - T_0)/N_t \qquad \text{and} \qquad k = 0, \dots, N_t, \tag{3.9}$$

at which we seek to recover our numerical solution. Following a finite element approach [118, Chap. 4] [40, Chap. 10], we discretise (3.8) in space by introducing a triangulation of Ω of characteristic length Δx, on which we build finite-dimensional

approximations to both $V(\Omega)$ and $Q(\Omega)$, named $V_h(\Omega)$ and $Q_h(\Omega)$. These are identified by their basis functions

$$V_h(\Omega) \coloneqq \operatorname{span}\{\boldsymbol{\phi}_0(\boldsymbol{x}), \ldots, \boldsymbol{\phi}_{N_{\boldsymbol{u}}-1}(\boldsymbol{x})\}, \quad \text{and} \tag{3.10a}$$

$$Q_h(\Omega) \coloneqq \operatorname{span}\{\psi_0(\boldsymbol{x}), \ldots, \psi_{N_p-1}(\boldsymbol{x})\}, \tag{3.10b}$$

with $N_{\boldsymbol{u}}$ and N_p being the number of degrees of freedom associated with the velocity and pressure variables, respectively, so that vectors $\boldsymbol{v} \in \mathbb{R}^{N_{\boldsymbol{u}}}$ and $\boldsymbol{q} \in \mathbb{R}^{N_p}$ correspond to functions in $V_h(\Omega)$ and $Q_h(\Omega)$ via

$$\boldsymbol{v}(\boldsymbol{x}) = \sum_{j=0}^{N_{\boldsymbol{v}}-1} [\boldsymbol{v}]_j \, \boldsymbol{\phi}_j(\boldsymbol{x}) \qquad \text{and} \qquad q(\boldsymbol{x}) = \sum_{j=0}^{N_p-1} [\boldsymbol{q}]_j \, \psi_j(\boldsymbol{x}), \tag{3.11}$$

respectively. (In the remainder of the chapter, we will refer somewhat interchangeably to functions in our discrete spaces and their vector representations according to the bases (3.10).) With this, we assemble the Galerkin matrices, namely the discrete counterparts of the bilinear forms appearing in (3.8). In particular, we define the mass and stiffness matrices for the velocity variable, respectively,

$$[\mathcal{M}_{\boldsymbol{u}}]_{m,n=0}^{N_{\boldsymbol{u}}-1} \coloneqq \int_\Omega \boldsymbol{\phi}_m(\boldsymbol{x}) \cdot \boldsymbol{\phi}_n(\boldsymbol{x}) \, d\boldsymbol{x} \quad \text{and} \tag{3.12a}$$

$$[\mathcal{K}_{\boldsymbol{u}}]_{m,n=0}^{N_{\boldsymbol{u}}-1} \coloneqq \int_\Omega \nabla\boldsymbol{\phi}_m(\boldsymbol{x}) \colon \nabla\boldsymbol{\phi}_n(\boldsymbol{x}) \, d\boldsymbol{x}, \tag{3.12b}$$

the discrete (negative) divergence matrix, responsible for the coupling between velocity and pressure,

$$[\mathcal{B}]_{m,n=0}^{N_p-1,N_{\boldsymbol{u}}-1} \coloneqq -\int_\Omega \psi_m(\boldsymbol{x}) \nabla \cdot \boldsymbol{\phi}_n(\boldsymbol{x}) \, d\boldsymbol{x}, \tag{3.13}$$

and discretisations of the advection operator for each temporal instant t_k in (3.9):

$$[\mathcal{W}_{\boldsymbol{u},k}]_{m,n=0}^{N_{\boldsymbol{u}}-1} \coloneqq \int_\Omega \left((\boldsymbol{w}(\boldsymbol{x}, t_k) \cdot \nabla) \, \boldsymbol{\phi}_n(\boldsymbol{x}) \right) \cdot \boldsymbol{\phi}_m(\boldsymbol{x}) \, d\boldsymbol{x}. \tag{3.14}$$

Combining spatial and temporal discretisations gives rise to the following discrete linear system approximating (3.8):

$$\left\{ \begin{array}{rcl} \dfrac{1}{\Delta t}\mathcal{M}_{\boldsymbol{u}} \left(\boldsymbol{u}^k - \boldsymbol{u}^{k-1}\right) + \mu\mathcal{K}_{\boldsymbol{u}}\boldsymbol{u}^k + \mathcal{W}_{\boldsymbol{u},k}\boldsymbol{u}^k + \mathcal{B}^T\boldsymbol{p}^k & = & \boldsymbol{f}^k \\ \mathcal{B}\boldsymbol{u}^k & = & \boldsymbol{0} \end{array} \right. , \quad k = 1, \ldots N_t, \tag{3.15}$$

where $\boldsymbol{0}$ is an all-zero vector of size N_p. Here, $\left[\boldsymbol{u}^k\right]_j$ and $\left[\boldsymbol{p}^k\right]_j$ represent the coefficients corresponding to the j-th basis function for the velocity and pressure variables

in eqs. (3.10a) and (3.10b), respectively, evaluated at the k-th temporal instant; analogously, $[\boldsymbol{f}^k]_j$ identifies the L^2-projection of the right-hand side of (3.8) on the j-th basis function for $V_h(\Omega)$, at the instant t_k:

$$[\boldsymbol{f}^k]_j \coloneqq \int_\Omega \boldsymbol{f}(\boldsymbol{x}, t_k) \cdot \boldsymbol{\phi}_j(\boldsymbol{x})\, d\boldsymbol{x} + \int_{\Gamma_N^{\boldsymbol{u}}} \boldsymbol{g}(\boldsymbol{s}, t_k) \cdot \boldsymbol{\phi}_j(\boldsymbol{s})\, d\boldsymbol{s}. \tag{3.16}$$

We now have all the ingredients necessary to assemble the monolithic space-time system corresponding to (3.15). To simplify notation, we introduce the operators

$$\mathcal{F}_{\boldsymbol{u},k} \coloneqq \frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t} + \mathcal{W}_{\boldsymbol{u},k} + \mu \mathcal{K}_{\boldsymbol{u}}, \qquad k = 1, \dots N_t. \tag{3.17}$$

We then arrange all the equations of (3.15) in a single block matrix, as

$$\begin{bmatrix} \mathcal{F}_{\boldsymbol{u},1} & \mathcal{B}^T & & & \\ \mathcal{B} & & & & \\ -\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t} & \ddots & & & \\ & \ddots & & \mathcal{F}_{\boldsymbol{u},N_t} & \mathcal{B}^T \\ & & & \mathcal{B} & \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^1 \\ \boldsymbol{p}^1 \\ \vdots \\ \boldsymbol{u}^{N_t} \\ \boldsymbol{p}^{N_t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^1 \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{f}^{N_t} \\ \boldsymbol{0} \end{bmatrix}, \tag{3.18}$$

where, with a slight abuse of notation, we re-define $\boldsymbol{f}^1$ from (3.16) in order to include the initial condition (3.4),

$$\left[\boldsymbol{f}^1\right]_j \coloneqq \int_\Omega \left(\boldsymbol{f}(\boldsymbol{x}, t_1) + \frac{1}{\Delta t} \bar{\boldsymbol{u}}^0(\boldsymbol{x}) \right) \cdot \boldsymbol{\phi}_j(\boldsymbol{x})\, d\boldsymbol{x} + \int_{\Gamma_N^{\boldsymbol{u}}} \boldsymbol{g}(\boldsymbol{s}, t_1) \cdot \boldsymbol{\phi}_j(\boldsymbol{s})\, d\boldsymbol{s}. \tag{3.19}$$

Notice that the system in (3.18) is of the block lower-triangular structure typical of space-time discretisations [47, 62]; moreover, since we use a *one*-step method to approximate the time derivative, the system is made of *two* block diagonals, with the 2×2 blocks composing them highlighted in (3.18). Usually, the solution to (3.15) is recovered via a time-stepping routine, which corresponds to solving (3.18) directly via (block) forward substitution. There are several methods to accelerate this procedure, mostly focusing on accelerating the inversion of the main diagonal blocks

$$\mathcal{A}_k \coloneqq \begin{bmatrix} \mathcal{F}_{\boldsymbol{u},k} & \mathcal{B}^T \\ \mathcal{B} & \end{bmatrix}, \qquad k = 1, \dots, N_t, \tag{3.20}$$

by designing an optimal preconditioner (see for example [118, Chap. 17.8], or [40, Chap. 4] for an overview). The *pressure convection-diffusion* (PCD) preconditioner is one of the most successful among these [85, 137], preconditioning the block structure

of (3.20) by finding an accurate and computable approximation to the pressure Schur complement. Our work stems from the same principles as PCD, but considers the whole space-time system (3.18), rather than just the spatial diagonal blocks (3.20). Motivation and derivation of our preconditioner are described in the following section.

3.2 Space-time block preconditioning for incompressible flow

Rather than inverting (3.18) using sequential time-stepping (i.e., forward substitution), here we tackle the solution of the full space-time system (3.18) all at once by using preconditioned GMRES [127], with an appropriate space-time block preconditioner. In particular, we focus on developing a preconditioner that treats space and time together and in parallel, which may yield faster time-to-solution than block forward substitution when sufficient processors are available. In the remainder of this section, we show how we can effectively build on concepts developed in the study of the single time-step matrix (3.20) to design preconditioners that are effective in the full space-time setting.

To develop our preconditioner, we start by reordering the space-time operator in (3.18) to take the form

$$\left[\begin{array}{cccc|ccc} \mathcal{F}_{\boldsymbol{u},1} & & & & \mathcal{B}^T & & \\ -\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t} & \ddots & & & & \ddots & \\ & \ddots & \mathcal{F}_{\boldsymbol{u},N_t} & & & & \mathcal{B}^T \\ \hline \mathcal{B} & & & & & & \\ & \ddots & & & & & \\ & & \mathcal{B} & & & & \end{array}\right] \left[\begin{array}{c} \boldsymbol{u}^1 \\ \vdots \\ \boldsymbol{u}^{N_t} \\ \hline \boldsymbol{p}^1 \\ \vdots \\ \boldsymbol{p}^{N_t} \end{array}\right] = \left[\begin{array}{c} \boldsymbol{f}^1 \\ \vdots \\ \boldsymbol{f}^{N_t} \\ \hline \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{array}\right]. \tag{3.21}$$

This rearrangement favours a separation of variables based on the physical unknown they refer to, rather than on the individual time-steps. We further denote the blocks composing (3.21) by

$$F_{\boldsymbol{u}} \coloneqq \begin{bmatrix} \mathcal{F}_{\boldsymbol{u},1} & & \\ -\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t} & \ddots & \\ & \ddots & \mathcal{F}_{\boldsymbol{u},N_t} \end{bmatrix}, \qquad \text{and} \qquad B \coloneqq \operatorname*{diag}_{N_t}(\mathcal{B}), \tag{3.22}$$

where $\operatorname{diag}_N(*)$ identifies a block diagonal matrix with N blocks containing $(*)$. As in the single time-step case [40, Chap. 9.2], we exploit the block structure of (3.21), and

look for a right preconditioner for GMRES in the following block upper-triangular form:

$$P_T \coloneqq \begin{bmatrix} F_{\boldsymbol{u}} & B^T \\ & \tilde{S}_p \end{bmatrix} \quad \Longleftrightarrow \quad P_T^{-1} \coloneqq \begin{bmatrix} F_{\boldsymbol{u}}^{-1} & -F_{\boldsymbol{u}}^{-1} B^T \tilde{S}_p^{-1} \\ & \tilde{S}_p^{-1} \end{bmatrix}. \tag{3.23}$$

This is justified in [107], and stems from considering the block LU factorisation of matrix (3.21),

$$\begin{bmatrix} F_{\boldsymbol{u}} & B^T \\ B & \end{bmatrix} = \begin{bmatrix} I & \\ BF_{\boldsymbol{u}}^{-1} & I \end{bmatrix} \begin{bmatrix} F_{\boldsymbol{u}} & B^T \\ & S_p \end{bmatrix}, \tag{3.24}$$

and choosing P_T as an approximation of the rightmost factor: particularly, taking $\tilde{S}_p$ in (3.23) as an approximation of the space-time pressure Schur complement S_p

$$\tilde{S}_p \approx S_p \coloneqq -BF_{\boldsymbol{u}}^{-1} B^T. \tag{3.25}$$

Convergence of GMRES applied to the full space-time system and preconditioned by P_T^{-1} is then exactly the same as the convergence of GMRES applied to the Schur complement problem, preconditioned by $\tilde{S}_p^{-1}$ [141].

The key advantage of the reordering in (3.21) and the block preconditioning discussed above is that we have effectively decoupled the imposition of the Lagrangian constraint from the solution of the PDE. Applying the preconditioner then requires inverting $F_{\boldsymbol{u}}$ (that is, time-integrating the velocity field), and applying an approximate Schur complement inverse (3.25). The former represents the discretisation of a single-variable parabolic PDE, which is much simpler to solve in parallel in space-time than the fully-coupled system (3.21); indeed parallel-in-(space-)time algorithms have shown to behave well on such problems: see [48, 138], and more generally the discussion in Chap. 2. Moreover, we claim that the pressure Schur complement (3.25) can naturally be approximated in a time-parallel fashion. To derive such an approximation, we follow two approaches which have been an inspiration for designing efficient preconditioners for the single time-step case.

Remark 2. One could also consider a block diagonal preconditioner, but due to the nonsymmetry of eqs. (3.22) and (3.25), MINRES [112] cannot be applied, thus negating one of the primary benefits of using a preconditioner in this form. With GMRES, block diagonal preconditioners are typically expected to require twice as many iterations [141] as block triangular ones, and in this case offer only a marginal reduction in computational cost. Block lower-triangular preconditioning is another option, but numerical tests (not included here) have favoured using an upper-triangular preconditioner. The latter is also more suited for right-preconditioning [141], which is required

by flexible GMRES [125] (and flexible Krylov methods are necessary if we use an inner Krylov method to approximately invert $F_{\boldsymbol{u}}$). For these reasons, P_T (3.23) remains our preconditioner of choice.

3.2.1 Small commutator approach

In [137], an effective preconditioning strategy for (3.20) is recovered by starting from an assumption on the commuting of certain operators. Although our Schur complement (3.25) differs from the case considered there, as it involves a time-derivative, we can still follow a similar line of reasoning. Let us begin by characterising our space-time Schur complement in more detail. Firstly, we can explicitly write $F_{\boldsymbol{u}}^{-1}$ as[2]

$$F_{\boldsymbol{u}}^{-1} = \begin{bmatrix} \mathcal{F}_{\boldsymbol{u},1}^{-1} & & \\ \mathcal{F}_{\boldsymbol{u},2}^{-1}\left(\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\mathcal{F}_{\boldsymbol{u},1}^{-1}\right) & \mathcal{F}_{\boldsymbol{u},2}^{-1} & \\ \mathcal{F}_{\boldsymbol{u},3}^{-1}\left(\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\mathcal{F}_{\boldsymbol{u},2}^{-1}\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\mathcal{F}_{\boldsymbol{u},1}^{-1}\right) & \mathcal{F}_{\boldsymbol{u},3}^{-1}\left(\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\mathcal{F}_{\boldsymbol{u},2}^{-1}\right) & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix}. \tag{3.26}$$

Multiplying on the left and right by the block diagonal matrices B and B^T, respectively, results in inner operators (*i.e.*, blocks) of the form

$$\mathcal{B}\mathcal{F}_{\boldsymbol{u},k}^{-1}\left(\prod_{i=1}^{k-j}\left(\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\mathcal{F}_{\boldsymbol{u},k-i}^{-1}\right)\right)\mathcal{B}^T, \quad \text{with} \quad \begin{cases} k = 1,\dots,N_t \\ j = 1,\dots,k \end{cases}, \tag{3.27}$$

where the product is assumed to expand towards the right, that is $\prod_{i=0}^{k} a_i = a_0 \prod_{i=1}^{k-1} a_i$, and returns identity if the subscript is larger than the superscript. We seek to adequately approximate the operators appearing in (3.27): to do so, we follow [137] as well as [40, Chap. 9.2], and make the assumption that the reaction-advection-diffusion operator and the gradient operator approximately commute. For each $\mathcal{F}_{\boldsymbol{u},k}$ in (3.17), let us define $\mathcal{F}_{p,k}$, an equivalent reaction-advection-diffusion operator acting on the pressure field:

$$\mathcal{F}_{p,k} \coloneqq \frac{\mathcal{M}_p}{\Delta t} + \mathcal{W}_{p,k} + \mu\mathcal{K}_p, \qquad k = 1,\dots,N_t, \tag{3.28}$$

[2] If we were solving the time-dependent Stokes equations (that is, if $\boldsymbol{w}(\boldsymbol{x},t) \equiv 0$), the structure of (3.26) would be further simplified, as $F_{\boldsymbol{u}}$ would be block Toeplitz; the presence of a time-dependent advection field, instead, causes the main diagonal block to vary down the diagonal. A similar disruption would be caused by using a non-uniform temporal grid when discretising Stokes: in that context, the difference between the blocks down the same diagonal is given by the factor Δt_k by which we divide the mass matrix. A similar analysis as the one reported here can be applied to that case as well.

where $\mathcal{M}_p$, $\mathcal{K}_p$, and $\mathcal{W}_{p,k}$ play the same role as the mass, stiffness, and advection matrices ((3.12a), (3.12b), and (3.14)), but for the pressure variable, namely

$$[\mathcal{M}_p]_{m,n=0}^{N_p-1} \coloneqq \int_\Omega \psi_m(\boldsymbol{x})\psi_n(\boldsymbol{x})\, d\boldsymbol{x}, \tag{3.29a}$$

$$[\mathcal{K}_p]_{m,n=0}^{N_p-1} \coloneqq \int_\Omega \nabla\psi_m(\boldsymbol{x}) \cdot \nabla\psi_n(\boldsymbol{x})\, d\boldsymbol{x}, \quad \text{and} \tag{3.29b}$$

$$[\mathcal{W}_{p,k}]_{m,n=0}^{N_p-1} \coloneqq \int_\Omega \left(\boldsymbol{w}(\boldsymbol{x}, t_k) \cdot \nabla\right) \psi_n(\boldsymbol{x})\psi_m(\boldsymbol{x})\, d\boldsymbol{x}, \quad \text{for} \quad k = 1, \dots, N_t. \tag{3.29c}$$

More details on how to assemble these operators are provided in Sec. 3.2.1.2. Translating the commutation assumption to the discrete form introduced in Sec. 3.1, we have

$$\left(\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\right)\left(\mathcal{M}_p^{-1}\mathcal{F}_{p,j}\right) - \left(\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{F}_{\boldsymbol{u},j}\right)\left(\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\right) \approx 0 \tag{3.30}$$

for each $j = 1, \dots, N_t$. Here, rescaling by the corresponding mass matrices is necessary, since the Galerkin operators have the effect of mapping functions to *functionals*, integrating over the whole spatial domain.

To get closer to our desired operators (3.27), we left-multiply (3.30) by $\mathcal{F}_{\boldsymbol{u},j}^{-1}\mathcal{M}_{\boldsymbol{u}}$ and right-multiply by $\mathcal{F}_{p,j}^{-1}\mathcal{M}_p$, thus obtaining

$$\mathcal{F}_{\boldsymbol{u},j}^{-1}\mathcal{B}^T \approx \mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\mathcal{F}_{p,j}^{-1}\mathcal{M}_p. \tag{3.31}$$

Continuing to left-multiply by $\mathcal{F}_{\boldsymbol{u},j+i}^{-1}\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}$, increasing i until we reach $j+i=k$, using (3.31) for each $j+i$, and finally left-multiplying by $\mathcal{B}$, gives us

$$\begin{aligned}
\mathcal{F}_{\boldsymbol{u},j+1}^{-1}\left(\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\mathcal{F}_{\boldsymbol{u},j}^{-1}\right)\mathcal{B}^T \approx \quad & \underbrace{\frac{\mathcal{F}_{\boldsymbol{u},j+1}^{-1}}{\Delta t}\mathcal{B}^T}_{\approx\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\mathcal{F}_{p,j+1}^{-1}\frac{\mathcal{M}_p}{\Delta t}} \mathcal{F}_{p,j}^{-1}\mathcal{M}_p \\
\implies \quad \mathcal{B}\mathcal{F}_{\boldsymbol{u},k}^{-1}\left(\prod_{i=1}^{k-j}\left(\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\mathcal{F}_{\boldsymbol{u},k-i}^{-1}\right)\right)\mathcal{B}^T \approx \mathcal{B}\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\mathcal{F}_{p,k}^{-1}&\left(\prod_{i=1}^{k-j}\left(\frac{\mathcal{M}_p}{\Delta t}\mathcal{F}_{p,k-i}^{-1}\right)\right)\mathcal{M}_p,
\end{aligned} \tag{3.32}$$

for any desired index $k = 1, \dots, N_t$. This gives us an approximation for each of the operators in (3.27), and consequently identifies the following candidate for a reasonable approximation of the space-time pressure Schur complement (3.25):

$$\tilde{S}_p \coloneqq -\operatorname*{diag}_{N_t}(\mathcal{B}\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{B}^T)\begin{bmatrix} \mathcal{F}_{p,1}^{-1} & & \\ \mathcal{F}_{p,2}^{-1}\left(\frac{\mathcal{M}_p}{\Delta t}\mathcal{F}_{p,1}^{-1}\right) & \mathcal{F}_{p,2}^{-1} & \\ \mathcal{F}_{p,3}^{-1}\left(\frac{\mathcal{M}_p}{\Delta t}\mathcal{F}_{p,2}^{-1}\frac{\mathcal{M}_p}{\Delta t}\mathcal{F}_{p,1}^{-1}\right) & \mathcal{F}_{p,3}^{-1}\left(\frac{\mathcal{M}_p}{\Delta t}\mathcal{F}_{p,2}^{-1}\right) & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix}\operatorname*{diag}_{N_t}(\mathcal{M}_p). \tag{3.33}$$

The expression above can be simplified by noticing that the central matrix shares the same structure as (3.26). In fact, it is exactly the inverse of the block bi-diagonal matrix

$$F_p \coloneqq \begin{bmatrix} \mathcal{F}_{p,1} & & \\ -\frac{\mathcal{M}_p}{\Delta t} & \ddots & \\ & \ddots & \mathcal{F}_{p,N_t} \end{bmatrix}, \tag{3.34}$$

which can be interpreted as a system stemming from the implicit Euler discretisation of a time-dependent PDE, just like its velocity counterpart (3.22). Moreover, by making use of the discrete inf-sup condition, it has been shown [40, Chap. 3.5] that the operator $\mathcal{B}\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{B}^T$ in (3.33) is spectrally equivalent to a discrete Laplacian operator acting on the pressure field. In principle, this operator can differ from the one appearing in (3.28), hence we denote it as

$$\tilde{\mathcal{K}}_p \approx \mathcal{B}\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{B}^T, \tag{3.35}$$

and delay a more in-depth discussion on its definition to Sec. 3.2.1.2. Furthermore, to simplify notation, we define the following block diagonal matrices

$$M_p \coloneqq \underset{N_t}{\operatorname{diag}}(\mathcal{M}_p) \qquad \text{and} \qquad K_p \coloneqq \underset{N_t}{\operatorname{diag}}\left(\tilde{\mathcal{K}}_p\right), \tag{3.36}$$

which ultimately allows us to write our approximation to the space-time pressure Schur complement (3.33) as

$$\tilde{S}_p \coloneqq -K_p F_p^{-1} M_p \quad \Longleftrightarrow \quad \tilde{S}_p^{-1} \coloneqq -M_p^{-1} F_p K_p^{-1}. \tag{3.37}$$

3.2.1.1 Comparison with single time-step block preconditioner

As briefly remarked at the end of Sec. 3.1, the PCD preconditioner [137] which is used for the acceleration of the solution for the single time-step system (3.20), and which largely motivates our work, is given by

$$\mathcal{P}_{T,k} \coloneqq \begin{bmatrix} \mathcal{F}_{\boldsymbol{u},k} & \mathcal{B}^T \\ & \tilde{\mathcal{S}}_{p,k} \end{bmatrix} \quad \Longleftrightarrow \quad \mathcal{P}_{T,k}^{-1} \coloneqq \begin{bmatrix} \mathcal{F}_{\boldsymbol{u},k}^{-1} & -\mathcal{F}_{\boldsymbol{u},k}^{-1}\mathcal{B}^T\tilde{\mathcal{S}}_p^{-1} \\ & \tilde{\mathcal{S}}_{p,k}^{-1} \end{bmatrix}, \tag{3.38}$$

for each time-step $k = 1, \ldots, N_t$, where the single time-step pressure Schur complement, $\mathcal{S}_{p,k} \coloneqq -\mathcal{B}\mathcal{F}_{\boldsymbol{u},k}^{-1}\mathcal{B}^T$, is approximated as

$$\mathcal{S}_{p,k} \approx \tilde{\mathcal{S}}_{p,k} \coloneqq -\tilde{\mathcal{K}}_p\mathcal{F}_{p,k}^{-1}\mathcal{M}_p \quad \Longleftrightarrow \quad \tilde{\mathcal{S}}_{p,k}^{-1} \coloneqq -\mathcal{M}_p^{-1}\mathcal{F}_{p,k}\tilde{\mathcal{K}}_p^{-1}, \tag{3.39}$$

rendering it strikingly similar to (3.37). The sum of the computational costs associated with applying the pressure Schur complement approximation (3.39) at each

time-step is analogous to that of applying its space-time counterpart once; moreover, the application of (3.37) can be conducted in parallel over time naturally. We can convince ourselves of this by considering the block diagonal structure of the matrices in (3.37): applying the inverses of K_p and M_p requires solving N_t independent systems involving $\tilde{\mathcal{K}}_p$ and $\mathcal{M}_p$, akin to applying (3.39) for each of the N_t time-steps. The sole overhead with respect to sequential time-stepping comes from the *bi*-diagonal structure of F_p: the off-diagonal coupling between one time-step and the next must be accounted for for *each* application of the space-time preconditioner, rather than once per time-step. Nonetheless, this extra cost remains negligible compared to the other operations.

In light of these considerations, we conclude that the difference in performance between using (3.38) at each time-step, and tackling the whole space-time system at once with (3.23), is given by two factors: (i) the difference in the total number of GMRES iterations to convergence using the space-time block preconditioner (3.23), which we denote with N_{it}^{ST}, and the average number of GMRES iterations to convergence per time-step using (3.38), identified by N_{it}^{0}; and (ii) the computational time associated with inverting the velocity spatial operator $\mathcal{F}_{\boldsymbol{u},k}$ at each time-step versus inverting the whole space-time velocity block $F_{\boldsymbol{u}}$. We denote the latter as $C_{F_{\boldsymbol{u}}}$, while for the former we consider an average of $C_{\mathcal{F}_{\boldsymbol{u}}}$ per time-step, so that the total time is given by $C_{\mathcal{F}_{\boldsymbol{u}}} \cdot N_t$. We expect our approach to be competitive if

$$\frac{N_{it}^{ST}}{N_{it}^{0}} C_{F_{\boldsymbol{u}}} < C_{\mathcal{F}_{\boldsymbol{u}}} N_t. \tag{3.40}$$

The ratio N_{it}^{ST}/N_{it}^{0} defines the overhead cost of the all-at-one approach in terms of the total number of preconditioner applications. Numerical results in Sec. 3.4.4 indicate that this ratio is $\in (1, 2)$ for most cases, and only increasing to 3 in some extreme settings. Thus, the overhead in performing all-at-once block preconditioning in the space-time setting is quite marginal compared to standard spatial preconditioning in sequential time-stepping. The efficiency of our approach then directly depends on the speedup that can be obtained when solving a vector-based time-dependent (advection-)diffusion problem using PinT or all-at-once techniques. With a sufficiently large number of MPI processes available, however, PinT and all-at-once solutions to space-time (advection-)diffusion problems have shown significant speedups over sequential time-stepping, as outlined in Chap. 2.

We point out that in general our space-time approach is associated with a larger memory footprint than sequential time-stepping. In order to ensure that the parallel

application of (3.37) can be conducted in a time comparable to that of applying (3.39), in fact, in our implementation we assign one processor per time-step, and have each processor assemble all the relevant spatial operators. As a consequence, the required memory grows linearly with the number of processors employed. Even though this effectively caps the maximum system size of our simulations in Sec. 3.4, such characteristic is common among PinT methods, and we do not consider it as a particularly burdensome limitation. As outlined in Chap. 1, in fact, time-parallelisation is generally employed within scenarios where computational resources are not a limiting factor.

3.2.1.2 Defining the pressure operators

One issue associated with the definition of the pressure reaction-advection-diffusion operator (3.28) and the pressure Laplacian (3.35) involves the choice of BC to impose on the operators themselves. In [41] and [40, Chap. 9.2.2], it is suggested that the various $\mathcal{F}_{p,k}$ should be assembled as if Robin BC,

$$\mu \nabla p(\boldsymbol{s},t)\, \boldsymbol{n}(\boldsymbol{s}) - (\boldsymbol{w}(\boldsymbol{s},t)\cdot \boldsymbol{n}(\boldsymbol{s}))\, p(\boldsymbol{s},t) = 0, \tag{3.41}$$

were imposed everywhere, while $\tilde{\mathcal{K}}_p$ should have Dirichlet BC on the outflow boundary (that is, on $\boldsymbol{s} \in \partial\Omega : \boldsymbol{u}(\boldsymbol{s})\cdot\boldsymbol{n}(\boldsymbol{s}) > 0$), and Neumann BC everywhere else. In particular, this choice implies that $\mathcal{K}_p$ in (3.28) might differ from $\tilde{\mathcal{K}}_p$. However, prescribing the same BC for the assembly of both the pressure reaction-advection-diffusion operator and the pressure Laplacian has the advantage of further simplifying (3.37) for the Stokes equations (3.2). Under the assumption $\mathcal{K}_p \equiv \tilde{\mathcal{K}}_p$, in fact, this becomes

$$\tilde{S}_p^{-1} \underset{\left(\begin{smallmatrix}\text{if } \mathcal{K}_p \equiv \tilde{\mathcal{K}}_p \\ \boldsymbol{w} \equiv \boldsymbol{0}\end{smallmatrix}\right)}{=} - \begin{bmatrix} \frac{\mathcal{K}_p^{-1}}{\Delta t} + \mu \mathcal{M}_p^{-1} & & \\ & -\frac{\mathcal{K}_p^{-1}}{\Delta t} & \ddots \\ & & \ddots \end{bmatrix}. \tag{3.42}$$

In comparison to (3.37), we need to multiply by one fewer operator, and inverting $\mathcal{M}_p$ and $\mathcal{K}_p$ can be done independently. (Notice that a preconditioner in this form is reminiscent of the Cahouet-Chabard preconditioner [14].) Hence, keeping performance in mind, in our code we rather choose to assemble both the pressure Laplacian and the pressure reaction-diffusion operator considering homogeneous Dirichlet BC at the outflow of the domain, and homogeneous Neumann BC everywhere else, thus mimicking the implementation in the IFISS package [42] (available at [82]). Moreover, early experiments showed that the two approaches provide comparable results in terms of acceleration of convergence for the test-cases considered.

We also remark that, using the BC prescribed above, (3.29b) becomes singular when dealing with fully enclosed Stokes flows, for which $\boldsymbol{u}(\boldsymbol{s}) \cdot \boldsymbol{n}(\boldsymbol{s}) = 0$, $\forall \boldsymbol{s} \in \partial\Omega$. In this context, in fact, the operator mimics the action of a Laplacian with Neumann BC everywhere on the boundary, whose kernel contains all constant functions. The GMRES procedure remains robust in this case too [40, Chap. 9.3.5], but we need to ensure that the chosen solver for $\tilde{\mathcal{K}}_p$ can deal with singular matrices.

Finally, since with (3.7) we only require L^2-regularity in space for the pressure variable, the operator (3.29b) might be ill-defined, depending on the discretisation chosen. A solution to this issue when considering a discontinuous pressure field is detailed in [40, Chap. 9.2.1].

3.2.1.3 Small commutator approach focusing on the *divergence* operator

The most recent edition of [40] proposes to consider the commutator between the discrete *divergence* (rather than *gradient*) operator, and the reaction-diffusion operator. This provides a better framework for justifying the kind of BC to assign to the discrete operators defined over the pressure field (see also [41]). In our case, choosing this approach would translate to considering the commutator

$$\left(\mathcal{M}_p^{-1}\mathcal{F}_{p,k}\right)\left(\mathcal{M}_p^{-1}\mathcal{B}\right) - \left(\mathcal{M}_p^{-1}\mathcal{B}\right)\left(\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{F}_{\boldsymbol{u},k}\right) \approx 0, \tag{3.43}$$

as opposed to (3.30). Following steps similar to the derivation in (3.32), this would eventually lead to a definition of the pressure Schur complement approximation where the block diagonal operators containing the pressure stiffness and mass matrix are swapped with respect to (3.37), that is:

$$\tilde{S}_p \coloneqq -M_p F_p^{-1} K_p \quad \Longleftrightarrow \quad \tilde{S}_p^{-1} \coloneqq -K_p^{-1} F_p M_p^{-1}. \tag{3.44}$$

The operations necessary for its application are virtually the same as those for (3.37). However, numerical experiments have indicated that the definition (3.44) does not provide significant improvements over (3.37) in terms of convergence. Additionally, the form (3.37) is more prone to simplifications similar to the one illustrated in (3.42) when solving Oseen equations. Therefore, for the remainder of this chapter, we consider the gradient commutator (3.37).

3.2.2 Green's function approach

Alternatively to the small-commutator approach described in Sec. 3.2.1, another heuristic for recovering a good candidate preconditioner for the Oseen equations has

been proposed in [85]. It is based on mapping the operators appearing in (3.1) to Fourier space, finding the associated Green's tensor in Fourier domain, and taking inspiration from its form to recover an approximation of the Schur complement. Here we demonstrate that such principles can also be extended to the space-time setting, and, in fact, arrive at the same Schur-complement approximation as derived in (3.37), providing additional validation for our choice of preconditioner.

We start by arranging the operators appearing in (3.1) in tensor form,

$$\left[\begin{array}{c|c} \mathcal{D}_{\varphi} & \nabla \\ \hline -\nabla\cdot & 0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{u}(\boldsymbol{x},t) \\ \hline p(\boldsymbol{x},t) \end{array}\right] = \left[\begin{array}{c} \boldsymbol{f}(\boldsymbol{x},t) \\ \hline 0 \end{array}\right], \tag{3.45}$$

where $\mathcal{D}_{(*)}$ denotes the block diagonal operator responsible for applying $(*)$ to each component of a vector in $\mathbb{R}^d$, while φ is defined as

$$\varphi \coloneqq \frac{\partial}{\partial t} + \boldsymbol{w}\cdot\nabla - \mu\nabla^2. \tag{3.46}$$

The (tensor) operator associated to (3.1) is then given by

$$\mathcal{L} \coloneqq \left[\begin{array}{c|c} \mathcal{D}_{\varphi} & \nabla \\ \hline -\nabla\cdot & 0 \end{array}\right]. \tag{3.47}$$

We seek to recover a candidate approximation for the pressure Schur complement appearing in our preconditioner (3.23), by following the form of the Green's tensor $\mathcal{G}$ associated with (3.47), under the assumption of an advection field constant over the spatial domain (but not necessarily in time), $\boldsymbol{w}(\boldsymbol{x},t) \equiv \boldsymbol{w}(t)$. To this purpose, we can exploit the fact that the Fourier transform of the Green's tensor $\hat{\mathcal{G}}$ is given by the inverse of the Fourier transform of (3.47), that is, $\hat{\mathcal{G}} = \hat{\mathcal{L}}^{-1}$. The operator equivalent to (3.46) in Fourier space is given by

$$\hat{\varphi} = \frac{\partial}{\partial t} + \mathrm{i}\boldsymbol{w}\cdot\boldsymbol{k} + \mu|\boldsymbol{k}|^2, \tag{3.48}$$

where $\boldsymbol{k} = [k_0, \ldots, k_{d-1}]^T$ is the frequency vector, and i represents the imaginary unit; consequently, the Fourier equivalent of (3.47) is given by

$$\hat{\mathcal{L}} = \left[\begin{array}{c|c} \mathcal{D}_{\hat{\varphi}} & \mathrm{i}\boldsymbol{k} \\ \hline -\mathrm{i}\boldsymbol{k}\cdot & 0 \end{array}\right]. \tag{3.49}$$

Let $\hat{\sigma}_p$ denote the (2,2) Schur complement in Fourier space,

$$\hat{\sigma}_p \coloneqq \mathrm{i}\boldsymbol{k}^T\mathcal{D}_{\hat{\varphi}^{-1}}\mathrm{i}\boldsymbol{k} = -|\boldsymbol{k}|^2\hat{\varphi}^{-1}. \tag{3.50}$$

Then, using the classic formula for inversion of a 2×2 block operator based on a block LDU decomposition, we recover

$$\begin{aligned}\hat{\mathcal{G}} = \hat{\mathcal{L}}^{-1} &= \left[\begin{array}{c|c} \mathcal{D}_{\hat{\varphi}^{-1}} - \mathcal{D}_{\hat{\varphi}^{-1}} \boldsymbol{k} \hat{\sigma}_p^{-1} \boldsymbol{k}^T \mathcal{D}_{\hat{\varphi}^{-1}} & -\mathcal{D}_{\hat{\varphi}^{-1}} \mathrm{i}\boldsymbol{k} \hat{\sigma}_p^{-1} \\ \hline \hat{\sigma}_p^{-1} \mathrm{i}\boldsymbol{k}^T \mathcal{D}_{\hat{\varphi}^{-1}} & \hat{\sigma}_p^{-1} \end{array}\right] \\ &= \left(\hat{\varphi}|\boldsymbol{k}|^2\right)^{-1} \left[\begin{array}{c|c} \mathcal{D}_{|\boldsymbol{k}|^2} - \boldsymbol{k}\boldsymbol{k}^T & \mathrm{i}\boldsymbol{k}\hat{\varphi} \\ \hline -\mathrm{i}\boldsymbol{k}^T \hat{\varphi} & -\hat{\varphi}^2 \end{array}\right].\end{aligned} \tag{3.51}$$

We focus our attention on the bottom-right block of (3.51). Transforming back from Fourier space to physical space, and using the properties of Fourier transforms, namely the correspondence of the operators $|\boldsymbol{k}|^2 \leftrightarrow -\nabla^2$ and $-\boldsymbol{k}\boldsymbol{k}^T \leftrightarrow \nabla(\nabla\cdot)$, as well as the identity $\nabla \times (\nabla\times) = \nabla(\nabla\cdot) - \nabla^2$, we get an expression for the Green's tensor:

$$\mathcal{G} = \left[\begin{array}{c|c} (\nabla \times (\nabla\times))\mathcal{D}_{\mathcal{G}_{\hat{\varphi}\nabla^2}} & \nabla\mathcal{G}_{-\nabla^2} \\ \hline (-\nabla\cdot)\mathcal{D}_{\mathcal{G}_{-\nabla^2}} & -(\frac{\partial}{\partial t} + \boldsymbol{w}\cdot\nabla - \mu\nabla^2)\mathcal{G}_{-\nabla^2} \end{array}\right], \tag{3.52}$$

where $\mathcal{G}_{-\nabla^2}$ is the fundamental solution for the negative Laplacian operator, while $\mathcal{G}_{\varphi\nabla^2}$ is the fundamental solution for the operator $\nabla^2\left(\frac{\partial}{\partial t} + \boldsymbol{w}\cdot\nabla - \mu\nabla^2\right)$. Keeping only the bottom-right block and discarding the rest, we get that the space-time pressure Schur complement operator σ_p satisfies

$$\sigma_p^{-1} q \approx -\left(\left(\frac{\partial}{\partial t} + \boldsymbol{w}\cdot\nabla - \mu\nabla^2\right)\mathcal{G}_{-\nabla^2}\right) * q, \qquad \forall q \in Q(\Omega), \tag{3.53}$$

on which we base its approximation. Considering that $\mathcal{G}_{-\nabla^2} * q$ should represent the solution to a (negative) Laplacian defined on the pressure space, we can discretise it using $\tilde{\mathcal{K}}_p^{-1} q$ defined in (3.35). Analogously, the space-time operator acting on the pressure space can be discretised via $F_p \approx \frac{\partial}{\partial t} + \boldsymbol{w}\cdot\nabla - \mu\nabla^2$, introduced in (3.34).

Altogether, this analysis suggests that a discrete approximation to the space-time pressure Schur complement S_p should satisfy

$$S_p^{-1} \approx \tilde{S}_p^{-1} \coloneqq -M_p^{-1} F_p K_p^{-1}, \tag{3.54}$$

where M_p and K_p are defined as in (3.36). Note that here we rescaled the operator by the pressure mass matrix for similar reasons as discussed for (3.30); this rescaling was also applied to the single time-step case in [85]. We see that the Green's function heuristic followed in this section yields the same approximation to the space-time pressure Schur complement as found via the small-commutator approach in Sec. 3.2.1. This is not necessarily surprising, as both analyses in some sense rely on neglecting boundary conditions, but the fact that two approaches provide the same approximation (3.54) gives confidence in the validity of the proposed preconditioner.

3.3 Eigenvalues clustering

To provide an indication of the effectiveness of (3.23) as a preconditioner, we investigate how the eigenvalues of the resulting preconditioned system spread in the complex plane. Although eigenvalues are not necessarily indicative of fast convergence for GMRES, particularly for highly nonsymmetric systems [68], poor clustering of eigenvalues is likely to yield poor convergence, while nicely bounded and clustered eigenvalues are still often indicative of an effective preconditioning.

We are interested in solving the following generalised eigenvalue problem:

$$\begin{bmatrix} F_{\boldsymbol{u}} & B^T \\ B & \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \lambda \begin{bmatrix} F_{\boldsymbol{u}} & B^T \\ & \tilde{S}_p \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix}. \tag{3.55}$$

The system presents $\lambda = 1$ as a solution with multiplicity $N_{\boldsymbol{u}}$, corresponding to a choice of $\boldsymbol{p} = \boldsymbol{0}$; the remaining eigenvalues are given by satisfying

$$\left(\tilde{S}_p^{-1} B F_{\boldsymbol{u}}^{-1} B^T + \lambda I\right) \boldsymbol{p} = 0. \tag{3.56}$$

From this, we see that the general eigenvalues of (3.55) directly correspond to the eigenvalues of the matrix $-\tilde{S}_p^{-1} B F_{\boldsymbol{u}}^{-1} B^T$. Substituting our approximation to the space-time pressure Schur complement (3.37), we have that the operator of interest becomes

$$-\tilde{S}_p^{-1} B F_{\boldsymbol{u}}^{-1} B^T = M_p^{-1} F_p K_p^{-1} B F_{\boldsymbol{u}}^{-1} B^T. \tag{3.57}$$

This matrix is block lower-triangular, because F_p and $F_{\boldsymbol{u}}^{-1}$ are both block lower-triangular and the remaining factors are block diagonal. Thus, to find its eigenvalues, we just need to recover the eigenvalues of the blocks on the main diagonal. These blocks consist of the operators

$$\mathcal{M}_p^{-1} \mathcal{F}_{p,k} \tilde{\mathcal{K}}_p^{-1} \mathcal{B} \mathcal{F}_{\boldsymbol{u},k}^{-1} \mathcal{B}^T, \qquad \text{with} \qquad k = 1, \dots, N_t, \tag{3.58}$$

whose factors can be separated as

$$\left(\tilde{\mathcal{K}}_p \mathcal{F}_{p,k}^{-1} \mathcal{M}_p\right)^{-1} \qquad \text{and} \qquad \mathcal{B} \mathcal{F}_{\boldsymbol{u},k}^{-1} \mathcal{B}^T, \qquad \text{with} \qquad k = 1, \dots, N_t. \tag{3.59}$$

By comparing these two factors with the definitions of the single time-step pressure Schur complement $\mathcal{S}_{p,k}$ and of its PCD approximation $\tilde{\mathcal{S}}_{p,k}$, both provided in Sec. 3.2.1.1, we can convince ourselves that the eigenvalue distribution of the space-time preconditioned system is directly dependent on that of the single time-step preconditioned system, and in particular on the spectral equivalence of $\mathcal{S}_{p,k}$ and $\tilde{\mathcal{S}}_{p,k}$, which we investigate next.

In order to simplify the upcoming analysis, we limit ourselves to the Stokes case, and we substitute $\tilde{\mathcal{K}}_p$ with $\mathcal{K}_p$, following Sec. 3.2.1.2. This renders the operator on the left in (3.59) *Symmetric Positive Definite* (SPD), since

$$\tilde{\mathcal{K}}_p \mathcal{F}_p^{-1} \mathcal{M}_p \underset{\substack{\text{if } \mathcal{K}_p \equiv \tilde{\mathcal{K}}_p \\ \boldsymbol{w} \equiv \mathbf{0}}}{=} \left(\frac{\mathcal{K}_p^{-1}}{\Delta t} + \mu \mathcal{M}_p^{-1} \right)^{-1}, \tag{3.60}$$

where both $\mathcal{K}_p$ and $\mathcal{M}_p$ are SPD. Notice that since $\boldsymbol{w} \equiv \mathbf{0}$, we can drop the dependence on the index k and write $\mathcal{F}_p \equiv \mathcal{F}_{p,k}$ and $\mathcal{F}_{\boldsymbol{u}} \equiv \mathcal{F}_{\boldsymbol{u},k}$, for each k.

Under these assumptions, we can investigate the spectral equivalence of the factors in (3.59) by considering the *generalised singular value problem*[3]

$$\begin{bmatrix} & \mathcal{B}^T \\ \mathcal{B} & \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \sigma \begin{bmatrix} \mathcal{F}_{\boldsymbol{u}} & \\ & \mathcal{K}_p \mathcal{F}_p^{-1} \mathcal{M}_p \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix}. \tag{3.62}$$

In fact, we have that the following relationship holds between the generalised singular values and the Rayleigh quotient of the matrices in (3.59):

$$\frac{\left\langle \mathcal{B} \mathcal{F}_{\boldsymbol{u}}^{-1} \mathcal{B}^T \boldsymbol{q}, \boldsymbol{q} \right\rangle}{\left\langle \mathcal{K}_p \mathcal{F}_p^{-1} \mathcal{M}_p \boldsymbol{q}, \boldsymbol{q} \right\rangle} = \sigma^2 = \frac{\left\langle \mathcal{B}^T \mathcal{M}_p^{-1} \mathcal{F}_p \mathcal{K}_p^{-1} \mathcal{B} \boldsymbol{v}, \boldsymbol{v} \right\rangle}{\left\langle \mathcal{F}_{\boldsymbol{u}} \boldsymbol{v}, \boldsymbol{v} \right\rangle}, \tag{3.63}$$

for all $(\boldsymbol{v}, \boldsymbol{q}) \in V_h(\Omega) \times Q_h(\Omega)$ such that $\langle \mathcal{F}_{\boldsymbol{u}} \boldsymbol{v}, \boldsymbol{v} \rangle = \left\langle \mathcal{K}_p \mathcal{F}_p^{-1} \mathcal{M}_p \boldsymbol{q}, \boldsymbol{q} \right\rangle$. Proving their spectral equivalence is then reduced to bounding σ^2, *i.e.*, finding two scalars $\underline{\sigma}^2$ and $\bar{\sigma}^2$ such that

$$\underline{\sigma}^2 \leq \frac{\left\langle \mathcal{B} \mathcal{F}_{\boldsymbol{u}}^{-1} \mathcal{B}^T \boldsymbol{q}, \boldsymbol{q} \right\rangle}{\left\langle \mathcal{K}_p \mathcal{F}_p^{-1} \mathcal{M}_p \boldsymbol{q}, \boldsymbol{q} \right\rangle} = \sigma^2 = \frac{\left\langle \mathcal{B}^T \mathcal{M}_p^{-1} \mathcal{F}_p \mathcal{K}_p^{-1} \mathcal{B} \boldsymbol{v}, \boldsymbol{v} \right\rangle}{\left\langle \mathcal{F}_{\boldsymbol{u}} \boldsymbol{v}, \boldsymbol{v} \right\rangle} \leq \bar{\sigma}^2, \tag{3.64}$$

either $\forall \boldsymbol{q} \in Q_h(\Omega)$ or $\forall \boldsymbol{v} \in V_h(\Omega)$. To simplify notation in the following derivation, we introduce these norms for our discrete pressure space:

$$\|\boldsymbol{q}\|_p^2 = \left\langle \mathcal{K}_p \mathcal{F}_p^{-1} \mathcal{M}_p \boldsymbol{q}, \boldsymbol{q} \right\rangle \qquad \text{and} \qquad \|\boldsymbol{q}\|_0^2 = \left\langle \mathcal{M}_p \boldsymbol{q}, \boldsymbol{q} \right\rangle. \tag{3.65}$$

Notice $\|*\|_0$ is simply the L^2-norm. For the discrete velocity space, instead, we define

$$\|\boldsymbol{v}\|_{\boldsymbol{u}}^2 = \langle \mathcal{F}_{\boldsymbol{u}} \boldsymbol{v}, \boldsymbol{v} \rangle \qquad \text{and} \qquad \|\boldsymbol{v}\|_1^2 = \langle \mathcal{K}_{\boldsymbol{u}} \boldsymbol{v}, \boldsymbol{v} \rangle, \tag{3.66}$$

where $\|\boldsymbol{v}\|_1 \equiv \|\nabla \boldsymbol{v}(\boldsymbol{x})\|_0$, the H_1-semi-norm.

[3] This procedure is akin to the analysis performed in [40, Chap. 3.5] for the Stokes equations. In that case, the aim was to prove the spectral equivalence between the chosen preconditioner $\mathcal{M}_p$ and the pressure Schur complement $\mathcal{B}\mathcal{K}_{\boldsymbol{u}}^{-1}\mathcal{B}^T$: the generalised singular value problem considered for this purpose was

$$\begin{bmatrix} & \mathcal{B}^T \\ \mathcal{B} & \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \sigma \begin{bmatrix} \mathcal{K}_{\boldsymbol{u}} & \\ & \mathcal{M}_p \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix}. \tag{3.61}$$

In our case the structure is identical, but the operators involved take into account the extra terms coming from the temporal discretisation: the single time step pressure Schur complement is $\mathcal{B}\mathcal{F}_{\boldsymbol{u}}\mathcal{B}^T$, and the proposed approximation is $\mathcal{K}_p\mathcal{F}_p^{-1}\mathcal{M}_p$.

Upper bound. We first find $\bar{\sigma}^2$: starting from the Rayleigh quotient on the left of (3.64), we obtain

$$\frac{\left\langle \mathcal{B}\mathcal{F}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\boldsymbol{q},\boldsymbol{q}\right\rangle^{\frac{1}{2}}}{\left\langle \mathcal{K}_p\mathcal{F}_p^{-1}\mathcal{M}_p\boldsymbol{q},\boldsymbol{q}\right\rangle^{\frac{1}{2}}}=\frac{\left\langle \mathcal{F}_{\boldsymbol{u}}^{-\frac{1}{2}}\mathcal{B}^T\boldsymbol{q},\mathcal{F}_{\boldsymbol{u}}^{-\frac{1}{2}}\mathcal{B}^T\boldsymbol{q}\right\rangle^{\frac{1}{2}}}{\|\boldsymbol{q}\|_p}=\frac{|\left\langle \boldsymbol{q},\mathcal{B}\boldsymbol{v}\right\rangle|}{\|\boldsymbol{q}\|_p\|\boldsymbol{v}\|_{\boldsymbol{u}}}, \tag{3.67}$$

where we have substituted $\boldsymbol{v}=\mathcal{F}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\boldsymbol{q}$. We then focus on the right-most term, and notice its numerator equals $|\left\langle q(\boldsymbol{x}),\nabla\cdot\boldsymbol{v}(\boldsymbol{x})\right\rangle|$, where $q(\boldsymbol{x})$ and $\boldsymbol{v}(\boldsymbol{x})$ are the functions corresponding to the vectors $\boldsymbol{q}$ and $\boldsymbol{v}$, according to (3.11). Using Cauchy-Schwarz,

$$|\left\langle q(\boldsymbol{x}),\nabla\cdot\boldsymbol{v}(\boldsymbol{x})\right\rangle|\leq\|q(\boldsymbol{x})\|_0\|\nabla\cdot\boldsymbol{v}(\boldsymbol{x})\|_0, \tag{3.68}$$

and moreover, we have

$$\|\nabla\cdot\boldsymbol{v}(\boldsymbol{x})\|_0^2\leq\|\nabla\cdot\boldsymbol{v}(\boldsymbol{x})\|_0^2+\|\nabla\times\boldsymbol{v}(\boldsymbol{x})\|_0^2=\|\nabla\boldsymbol{v}(\boldsymbol{x})\|_0^2=\|\boldsymbol{v}\|_1^2, \tag{3.69}$$

which allows us to write

$$\frac{|\left\langle \boldsymbol{q},\mathcal{B}\boldsymbol{v}\right\rangle|}{\|\boldsymbol{q}\|_p\|\boldsymbol{v}\|_{\boldsymbol{u}}}\leq\max_{\boldsymbol{q}\in Q_h(\Omega)}\frac{\|\boldsymbol{q}\|_0}{\|\boldsymbol{q}\|_p}\max_{\boldsymbol{v}\in V_h(\Omega)}\frac{\|\boldsymbol{v}\|_1}{\|\boldsymbol{v}\|_{\boldsymbol{u}}}, \tag{3.70}$$

at which point finding the upper bound for (3.64) becomes an exercise in proving equivalence of norms. For the velocity field, we have:

$$\|\boldsymbol{v}\|_1^2=\left\langle \mathcal{K}_{\boldsymbol{v}}\boldsymbol{v},\boldsymbol{v}\right\rangle\leq\max\left\{1,\mu^{-1}\right\}\left\langle\left(\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}+\mu\mathcal{K}_{\boldsymbol{u}}\right)\boldsymbol{v},\boldsymbol{v}\right\rangle=\max\left\{1,\mu^{-1}\right\}\|\boldsymbol{v}\|_{\boldsymbol{u}}^2. \tag{3.71}$$

For the pressure field, the analysis requires more care. Firstly, we can write

$$\frac{\|\boldsymbol{q}\|_0^2}{\|\boldsymbol{q}\|_p^2}=\frac{\left\langle \mathcal{M}_pq,q\right\rangle}{\left\langle \mathcal{K}_p\mathcal{F}_p^{-1}\mathcal{M}_pq,q\right\rangle}\leq\frac{\bar{\lambda}(\mathcal{M}_p)}{\underline{\lambda}\left(\mathcal{K}_p\mathcal{F}_p^{-1}\mathcal{M}_p\right)}=\bar{\lambda}(\mathcal{M}_p)\bar{\lambda}\left(\mathcal{K}_p^{-1}\mathcal{F}_p\mathcal{M}_p^{-1}\right), \tag{3.72}$$

where $\underline{\lambda}(*)$ and $\bar{\lambda}(*)$ respectively denote the smallest and largest eigenvalues of matrix $(*)$, and we once again exploit the fact that the matrices involved are SPD. We also notice that

$$\lambda\left(\mathcal{K}_p^{-1}\mathcal{F}_p\mathcal{M}_p^{-1}\right)=\lambda\left(\frac{\mathcal{K}_p^{-1}}{\Delta t}+\mu\mathcal{M}_p^{-1}\right)\subseteq\left[\frac{\Delta t^{-1}}{\bar{\lambda}(\mathcal{K}_p)}+\frac{\mu}{\bar{\lambda}(\mathcal{M}_p)},\frac{\Delta t^{-1}}{\underline{\lambda}(\mathcal{K}_p)}+\frac{\mu}{\underline{\lambda}(\mathcal{M}_p)}\right]. \tag{3.73}$$

The final ingredient we need to recover an expression for $\bar{\sigma}^2$, is a bound on the eigenvalues of the Galerkin mass and stiffness matrices: for $\Omega\in\mathbb{R}^2$, these are given by

$$\begin{array}{rcccll}\underline{c}_{\mathcal{M}}^{(*)}\Delta x^2 & \leq & \lambda(\mathcal{M}_{(*)}) & \leq & \bar{c}_{\mathcal{M}}^{(*)}\Delta x^2 & \text{[40, Prop.1.30]},\\ \underline{c}_{\mathcal{K}}^{(*)}\Delta x^2 & \leq & \lambda(\mathcal{K}_{(*)}) & \leq & \bar{c}_{\mathcal{K}}^{(*)} & \text{[40, Thm.1.33]}\ ,\end{array} \tag{3.74}$$

where $\underline{c}_{\mathcal{M}}$, $\bar{c}_{\mathcal{M}}$, $\underline{c}_{\mathcal{K}}$, and $\bar{c}_{\mathcal{K}}$ are constants depending on the domain and the type of discretisation; these bounds hold both for pressure and velocity, $(*) = p$ or $(*) = \boldsymbol{u}$, and are valid for a quasi-uniform discretisation defined on a regular 2D domain, with characteristic mesh size Δx. Substituting (3.74) for the pressure variable in (3.72) gives the upper bound for the equivalence of the norms introduced in (3.65):

$$\frac{\|\boldsymbol{q}\|_0^2}{\|\boldsymbol{q}\|_p^2} \leq \bar{c}_{\mathcal{M}}^p \Delta x^2 \left(\frac{1}{\Delta t} \frac{1}{\underline{c}_{\mathcal{K}}^p \Delta x^2} + \frac{\mu}{\underline{c}_{\mathcal{M}}^p \Delta x^2} \right) = \frac{1}{\Delta t} \frac{\bar{c}_{\mathcal{M}}^p}{\underline{c}_{\mathcal{K}}^p} + \frac{\mu \bar{c}_{\mathcal{M}}^p}{\underline{c}_{\mathcal{M}}^p}. \tag{3.75}$$

The right bound on (3.64) is finally recovered by combining eqs. (3.71) and (3.75)

$$\frac{\left\langle \mathcal{B}\mathcal{F}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\boldsymbol{q}, \boldsymbol{q} \right\rangle}{\left\langle \mathcal{K}_p\mathcal{F}_p^{-1}\mathcal{M}_p\boldsymbol{q}, \boldsymbol{q} \right\rangle} \leq \max\{\mu, 1\} \left(\frac{1}{\mu \Delta t} \frac{\bar{c}_{\mathcal{M}}^p}{\underline{c}_{\mathcal{K}}^p} + \frac{\bar{c}_{\mathcal{M}}^p}{\underline{c}_{\mathcal{M}}^p} \right) = \bar{\sigma}^2. \tag{3.76}$$

Lower bound. To find an expression for the lower bound $\underline{\sigma}^2$, we make use of the discrete inf-sup condition

$$0 < \gamma \leq \min_{\boldsymbol{q} \neq \mathbf{1}} \max_{\boldsymbol{v} \neq \mathbf{0}} \frac{|\langle \boldsymbol{q}, \mathcal{B}\boldsymbol{v} \rangle|}{\|\boldsymbol{v}\|_1 \|\boldsymbol{q}\|_0}, \tag{3.77}$$

where $\mathbf{1}$ denotes a vector filled with 1 (representing a FE function constant over the whole domain). Starting from (3.67) and using (3.77) we obtain

$$\frac{\left\langle \mathcal{B}\mathcal{F}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\boldsymbol{q}, \boldsymbol{q} \right\rangle^{\frac{1}{2}}}{\left\langle \mathcal{K}_p\mathcal{F}_p^{-1}\mathcal{M}_p\boldsymbol{q}, \boldsymbol{q} \right\rangle^{\frac{1}{2}}} \underset{(\boldsymbol{v} = \mathcal{F}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\boldsymbol{q})}{=} \frac{|\langle \boldsymbol{q}, \mathcal{B}\boldsymbol{v} \rangle|}{\|\boldsymbol{q}\|_p \|\boldsymbol{v}\|_{\boldsymbol{u}}} \geq \min_{\boldsymbol{q} \neq \mathbf{1}} \max_{\boldsymbol{v} \neq \mathbf{0}} \frac{|\langle \boldsymbol{q}, \mathcal{B}\boldsymbol{v} \rangle|}{\|\boldsymbol{q}\|_p \|\boldsymbol{v}\|_{\boldsymbol{u}}} \geq \gamma \frac{\|\boldsymbol{q}\|_0}{\|\boldsymbol{q}\|_p} \frac{\|\boldsymbol{v}\|_1}{\|\boldsymbol{v}\|_{\boldsymbol{u}}}, \tag{3.78}$$

which once again leaves us with the task of proving equivalence of norms. For the velocity norms, we exploit Poincaré's inequality:

$$\begin{aligned} \|\boldsymbol{v}\|_1^2 &\geq \frac{1}{2c_p^2}\|\boldsymbol{v}\|_0^2 + \frac{1}{2}\|\boldsymbol{v}\|_1^2 = \frac{\Delta t}{2c_p^2} \frac{\langle \mathcal{M}_{\boldsymbol{u}}\boldsymbol{v}, \boldsymbol{v} \rangle}{\Delta t} + \frac{1}{2\mu}\mu \langle \mathcal{K}_{\boldsymbol{u}}\boldsymbol{v}, \boldsymbol{v} \rangle \\ &\geq \frac{1}{2} \min\left\{ \frac{\Delta t}{c_p^2}, \frac{1}{\mu} \right\} \|\boldsymbol{v}\|_{\boldsymbol{u}}^2, \end{aligned} \tag{3.79}$$

where c_p is Poincaré's constant, such that $\|\boldsymbol{v}\|_0 \leq c_p \|\boldsymbol{v}\|_1$. For the pressure norms we follow a similar derivation as for the upper bound (3.75), and recover

$$\frac{\|\boldsymbol{q}\|_0^2}{\|\boldsymbol{q}\|_p^2} \geq \underline{c}_{\mathcal{M}}^p \Delta x^2 \left(\frac{1}{\bar{c}_{\mathcal{K}}^p \Delta t} + \frac{\mu}{\bar{c}_{\mathcal{M}}^p \Delta x^2} \right) = \frac{\Delta x^2}{\Delta t} \frac{\underline{c}_{\mathcal{M}}^p}{\bar{c}_{\mathcal{K}}^p} + \mu \frac{\underline{c}_{\mathcal{M}}^p}{\bar{c}_{\mathcal{M}}^p}. \tag{3.80}$$

Combining the two bounds eqs. (3.79) and (3.80), and substituting them in (3.78) provides a lower bound for (3.64):

$$\frac{\left\langle \mathcal{B}\mathcal{F}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\boldsymbol{q}, \boldsymbol{q} \right\rangle}{\left\langle \mathcal{K}_p\mathcal{F}_p^{-1}\mathcal{M}_p\boldsymbol{q}, \boldsymbol{q} \right\rangle} \geq \frac{\gamma^2}{2} \min\left\{ 1, \frac{\mu \Delta t}{c_p^2} \right\} \left(\frac{\Delta x^2}{\mu \Delta t} \frac{\underline{c}_{\mathcal{M}}^p}{\bar{c}_{\mathcal{K}}^p} + \frac{\underline{c}_{\mathcal{M}}^p}{\bar{c}_{\mathcal{M}}^p} \right) = \underline{\sigma}^2. \tag{3.81}$$

Collecting the results from eqs. (3.76) and (3.81), and assuming small μ and Δt, we get

$$\frac{\gamma^2}{2c_p^2}\left(\Delta x^2 \frac{\underline{c}_{\mathcal{M}}^p}{\bar{c}_{\mathcal{K}}^p} + \mu\Delta t \frac{\underline{c}_{\mathcal{M}}^p}{\bar{c}_{\mathcal{M}}^p}\right) \leq \frac{\left\langle \mathcal{B}\mathcal{F}_{\boldsymbol{u}}^{-1}\mathcal{B}^T\boldsymbol{q},\boldsymbol{q}\right\rangle}{\left\langle \mathcal{K}_p\mathcal{F}_p^{-1}\mathcal{M}_p\boldsymbol{q},\boldsymbol{q}\right\rangle} \leq \frac{1}{\mu\Delta t}\frac{\bar{c}_{\mathcal{M}}^p}{\underline{c}_{\mathcal{K}}^p} + \frac{\bar{c}_{\mathcal{M}}^p}{\underline{c}_{\mathcal{M}}^p}. \tag{3.82}$$

Unfortunately, formula (3.82) fails to provide a useful bound as $\Delta x, \Delta t \to 0$, since the lower side approaches 0 and the upper bound diverges to ∞. Even so, we can still provide a numerical indication of the clustering of the eigenvalues of the preconditioned system. Plots of the eigenvalues of (3.58) in the complex plane are shown in Fig. 3.1, for varying values of Δt, Δx, and intensity of the advection field $\boldsymbol{w}(\boldsymbol{x},t)$ in (3.1) (the latter is regulated by the Péclet number, Pe: see (3.86) for its definition). We see that in all cases the eigenvalues are nicely bounded, they remain away from the origin, and are largely independent of variations in the discretisation parameters, providing semi-rigorous theoretical support for the proposed preconditioner.

It is worth pointing out that increasing the Péclet number has the effect of pushing some eigenvalues towards 2, and spreading more of them away from the real axis. It also makes the distributions more sensitive with respect to the level of refinement in the spatial discretisation, as can be seen comparing top and bottom plots in Fig. 3.1: the dark blue tokens distribute themselves more neatly in an arc as Δx decreases. These considerations make us wary of a possible degradation in the performance of the preconditioner as advection becomes predominant, an issue that is observed and discussed in Sec. 3.4.2.2. In general, however, most of the eigenvalues remain clustered inside $[0.1, 2] \times [-0.6, 0.6]$ in the complex plane.

3.4 Results

To demonstrate the performance of the preconditioner introduced in Sec. 3.2, we consider its application for the solution of a variety of flow configurations. For simplicity, we focus on problems defined on $\Omega \subset \mathbb{R}^2$, although we point out that the applicability of (3.23) is by no means limited to 2D simulations. The test-cases hereby proposed have been adapted from problems used extensively in the literature (see for example [40, Chap. 3.1 and Chap. 6.1]), and are briefly described in Sec. 3.4.1. The experiments conducted provide evidence for the optimal scalability of (3.23), as well as its potential for speedup via time parallelisation. Additionally, we investigate simple extensions to more general frameworks, such as the solution of nonlinear incompressible flow, and its applicability in advection-dominated regimes.

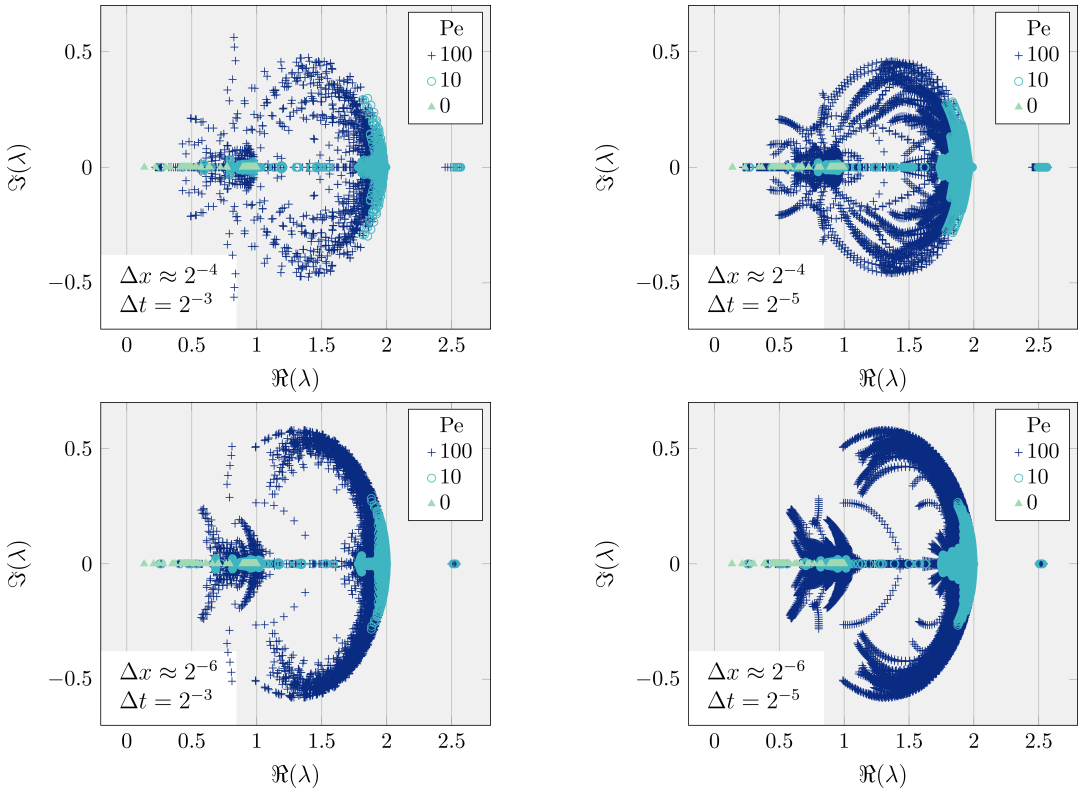


Figure 3.1: Eigenvalue distribution of the operators (3.58), corresponding to problem 4, for varying values of Δt, Δx, and Péclet number, Pe, which represents the intensity of the advection field. Here, Pe denotes the continuous Péclet number (i.e., not scaled by mesh spacing); see Sec. 3.4.1 for its definition (notice choosing $\mathrm{Pe} = 0$ corresponds to problem 1). The eigenvalues are computed numerically using the MATLAB function `eigs`.

3.4.1 Model problems

The first three problems described in this section are test-cases for the time-dependent Stokes equations (3.2), while the last one is designed for Oseen (3.1). They are defined as follows.

Problem 1 *Driven cavity flow.* This represents a fully enclosed flow, defined on a square spatial domain $\boldsymbol{x} = (x, y)^T \in [0, 1]^2 =: \Omega_\square$. No forcing term is considered, and homogeneous Dirichlet BC are imposed on the bottom, left and right sides of the domain. To accelerate the flow, we prescribe the velocity on the top side

$$\boldsymbol{u}_C(\boldsymbol{x}|_{y=1}, t) = 8tx(1-x)(2x^2 - 2x + 1). \tag{3.83}$$

This profile is regularised so as to smoothly match the BC on the left and right sides. Notice that, in order to introduce time-dependency, the velocity is ramped up linearly

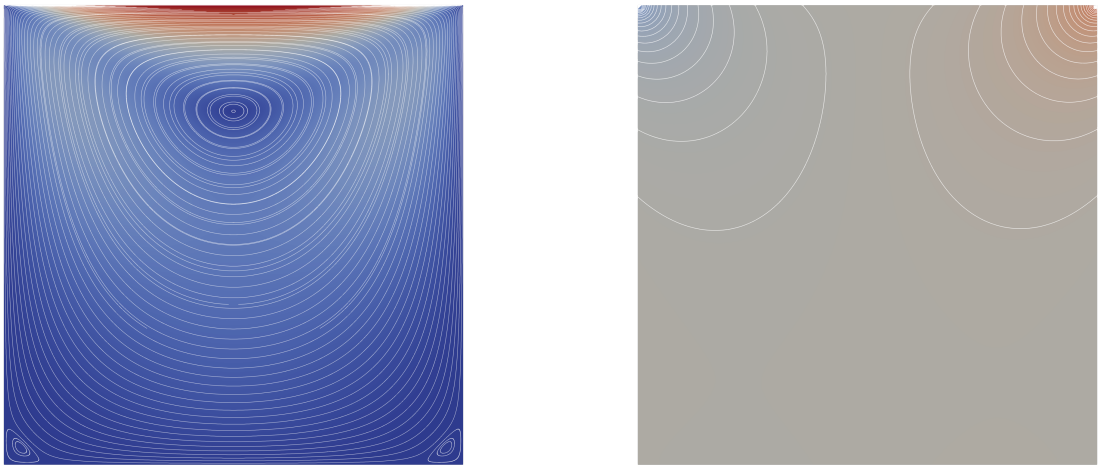


Figure 3.2: Velocity magnitude and streamlines (left), and pressure contour plot (right) for an example solution of the driven cavity flow problem 1.

from a quiet state: this is done in the following test-cases as well. An example solution for this problem is shown in Fig. 3.2.

Problem 2 *Poiseuille flow.* Also defined on $\Omega_{\square}$, this set-up models laminar flow along a channel, for which the following analytical solution can be recovered:

$$\boldsymbol{u}_P(\boldsymbol{x},t) = 4t \begin{bmatrix} y(1-y) \\ 0 \end{bmatrix}, \quad p_P(\boldsymbol{x},t) = 8(1-x). \tag{3.84}$$

The forcing term is defined in such a way to counterbalance the time-derivative: $\boldsymbol{f}_P = 4[y(1-y),0]^T$. No-slip conditions are included in the top and bottom sides of the channel, and a parabolic inflow profile is prescribed at $x = 0$. At the outflow

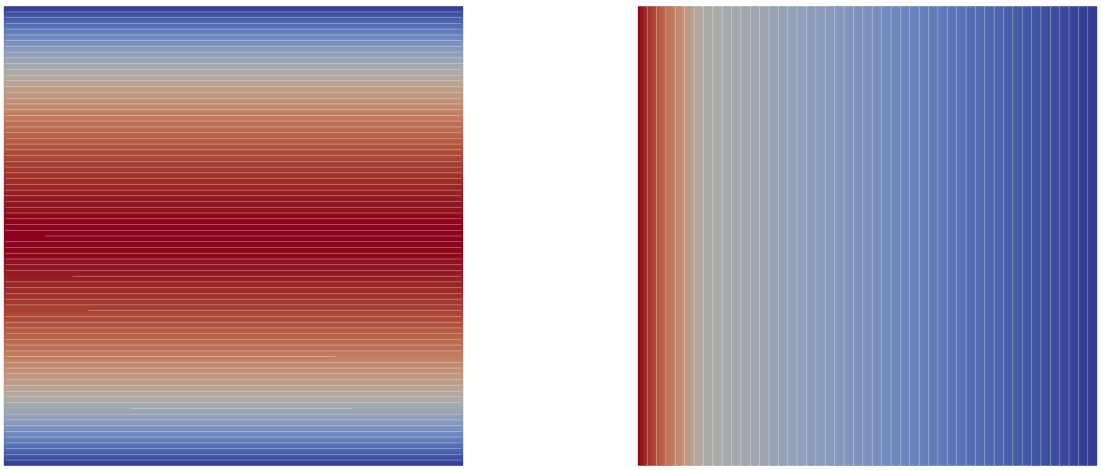


Figure 3.3: Velocity magnitude and streamlines (left), and pressure contour plot (right) for the analytical solution of the Poiseuille flow problem 2.

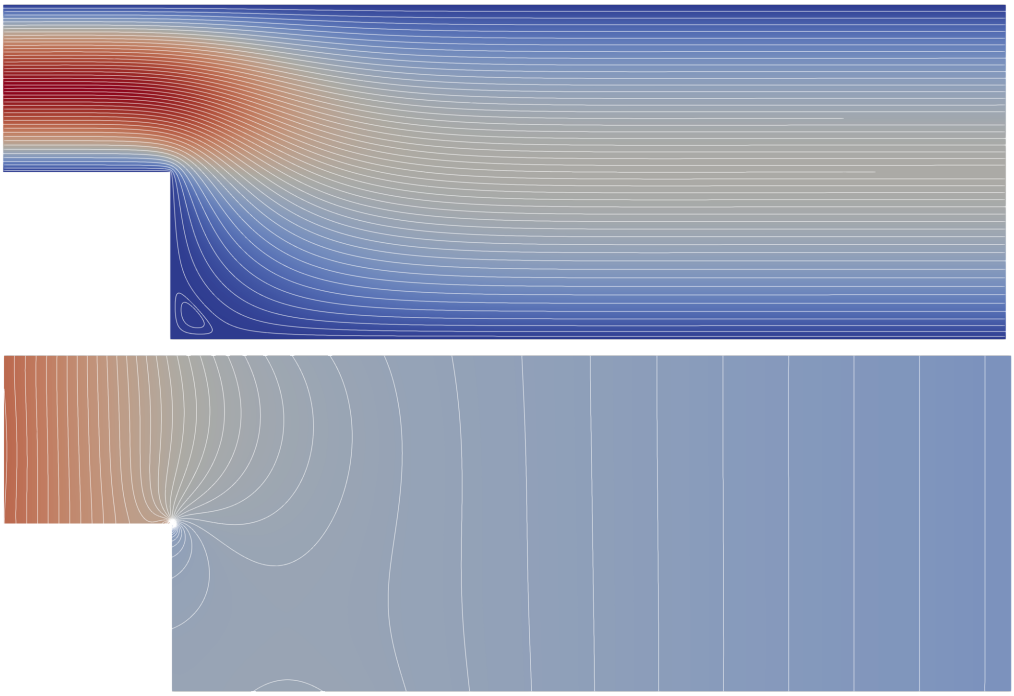


Figure 3.4: Velocity magnitude and streamlines (top), and pressure contour plot (bottom) for an example solution of the flow over backward-facing step problem 3.

$x = 1$, homogeneous Neumann BC are imposed. The analytical solution is plotted in Fig. 3.3.

Problem 3 *Flow over backward-facing step.* This is another example of flow through a channel, except the fluid undergoes a sudden expansion. The domain is a stretched L-shape, $\Omega_L = [0, 8] \times [0, 1] \cup [1, 8] \times [0, -1]$, and the BC imposed are similar to those for Poiseuille (the inflow, where the parabolic velocity profile is prescribed, is on the side $x = 0$). As for problem 1, no forcing term is included; an example solution is provided in Fig. 3.4.

Problem 4 *Double-glazing.* The set-up for this problem is analogous to that of the driven cavity flow, except now the effect of a recirculating wind is included in the system:

$$\boldsymbol{w}_G(\boldsymbol{x}, t) = 2t \begin{bmatrix} -(2y-1)(4x^2 - 4x + 1) \\ (2x-1)(4y^2 - 4y + 1) \end{bmatrix} \mu\mathrm{Pe}, \tag{3.85}$$

whose intensity can be adjusted by tweaking the value of the Péclet number Pe [118, Chap. 13.2]. This parameter describes the relative intensity of advective transport with respect to diffusive transport, and for a generic advection field $\boldsymbol{w}(\boldsymbol{x}, t)$ it can be defined as

$$\mathrm{Pe} := \frac{LU}{\mu} = \frac{L}{\mu} \left(\frac{1}{T - T_0} \int_{T_0}^{T} \max_{\boldsymbol{x} \in \Omega} \boldsymbol{w}(\boldsymbol{x}, t) \, dt \right), \tag{3.86}$$

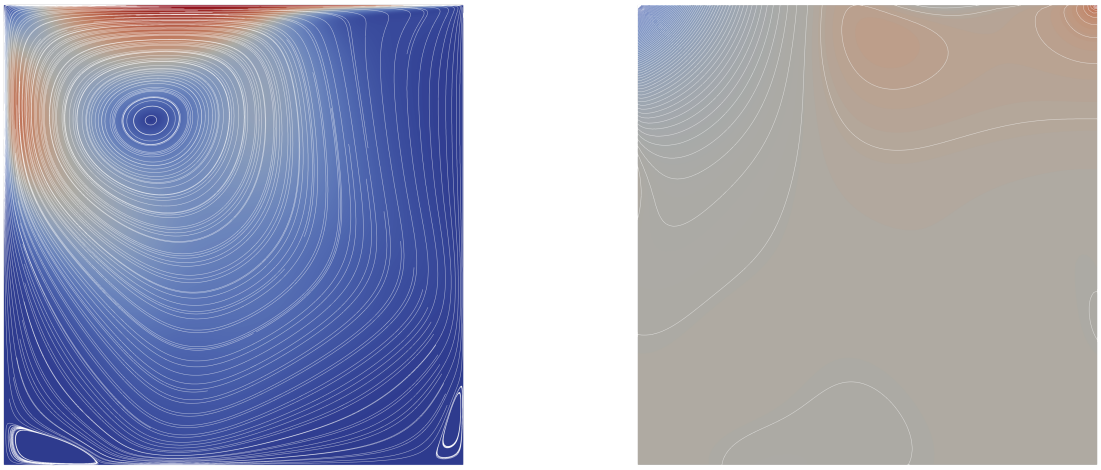


Figure 3.5: Velocity magnitude and streamlines (left), and pressure contour plot (right) for an example solution of the double-glazing flow problem 4 with $\mathrm{Pe} = 10$.

where L is a characteristic length of the spatial domain ($L = 1$ for this problem), and U a characteristic speed (considered averaged over the temporal domain). The impact that the added advection field has on the solution is illustrated in Fig. 3.5.

For each test-case the temporal domain is given by $t \in [0, 1]$, and the viscosity parameter is fixed to $\mu = 1$. The discrete functional spaces are approximated using Taylor-Hood finite elements: P_2 for velocity and P_1 for pressure [118, Chap. 17.4], both defined on a triangular mesh. The code developed for our experiments is publicly available at the repository [29]. It makes use of the MFEM finite element package [102] for the assembly of the finite element matrices, and we rely on the PETSc [7] and *hypre* [81] implementations of the various solvers used. Unless otherwise specified, we prescribe a null space-time initial guess for the GMRES iterations.

3.4.2 Performance of preconditioner

In this section, we conduct a series of experiments aimed at measuring the effectiveness of the proposed preconditioner (3.23) with GMRES. We chose the total number of iterations to convergence as the main measure of performance. The reason behind this preference (rather than, for example, computational time), lies in the fact that we believe it provides a more indicative metric for assessing the efficacy of our preconditioning strategy. This is in fact less dependent on the details of the actual implementation of the underlying solvers, and particularly of the one for the space-time velocity block (3.22), the optimal design of which is beyond the purpose of this

work: indeed a key point of our approach is the flexibility it provides in choosing such a solver.

3.4.2.1 Exact *versus* approximate solvers

As a first experiment, we directly test the application of our preconditioner to the solution of the model problems introduced in Sec. 3.4.1. We provide two different set-ups for the solvers involved in the application of (3.23): an *ideal* one, in which we make use of exact solvers for the relevant blocks, in order to provide a best-case scenario for the performance of such preconditioner; and an *approximate* case, which instead gives us a measure of the performance degradation we can expect by applying iterative solvers instead of exact solvers to approximate the solution of the relevant systems, which is more realistic in practice.

In more detail, the application of the inverse of (3.23) involves inverting three different operators: two of these are associated with the application of the approximate space-time Schur complement (3.37), and require inverting the pressure mass matrix $\mathcal{M}_p$ (3.29a), and the pressure "Laplacian" $\tilde{\mathcal{K}}_p$ (3.35); the last one must deal with the monolithic space-time matrix for the velocity variable $F_{\boldsymbol{u}}$ (3.22). In the ideal case, we exploit LU solvers for both (3.29a) and (3.35), and a sequential time-stepping procedure for (3.22) (the latter further requires inverting the spatial velocity operator $\mathcal{F}_{\boldsymbol{u},k}$ (3.17) at each time-step k, for which we also use LU).

However, resorting to direct methods becomes infeasible when solving systems of a large size, both in terms of memory and of computations required. Moreover, our final goal is to actually parallelise in time the solution of the monolithic system. In order to do so, we must substitute the time-stepping procedure for the space-time velocity block $F_{\boldsymbol{u}}$ with a parallel alternative. Here, we use GMRES preconditioned by *hypre*'s implementation of nonsymmetric AMG based on approximate ideal restriction (AIR) [99, 100], which we introduced in Sec. 2.2.4. Although likely not as effective as space-time geometric multigrid techniques (e.g., [77]) for diffusive problems, AIR is likely more robust for advective regimes and also less intrusive to apply, as we simply need to form the space-time velocity matrix and pass it to *hypre*. Due to the use of an inner Krylov method, we swap the outer GMRES solver with a *flexible*-GMRES (FGMRES) solver [125]. The inversions of the pressure mass and stiffness matrices are also approximated: for $\mathcal{M}_p$, we apply a fixed number of Chebyshev iterations [154], making use of the bounds on its eigenvalues provided by [155]; for $\tilde{\mathcal{K}}_p$, we apply *hypre*'s BoomerAMG [121].

Table 3.1: Number of GMRES iterations to 10^{-10} relative residual tolerance, when applied to the monolithic space-time system (3.21) and right-preconditioned with P_T^{-1} (3.23). Different levels of refinement are considered both for the spatial (rows) and temporal meshes (columns). The four sets of results refer to the four test problems introduced in Sec. 3.4.1 (problem 4 has Péclet number Pe = 10). Results on the left of each column are recovered using exact solvers, and results on the right of each column (in brackets) are recovered using approximate iterative solvers (inversion of $\mathcal{M}_p$ is approximated with 8 Jacobi-preconditioned Chebyshev iterations, inversion of $\mathcal{K}_p$ is approximated using 15 iterations of AMG, and the solution of the space-time velocity block $F_{\boldsymbol{u}}$ is recovered applying 15 iterations of GMRES preconditioned by AIR). The outer solver in the latter case is FGMRES. Crosses identify simulations that could not be completed due to memory requirements becoming too severe.

Pb	$\Delta t\rightarrow$ $\Delta x\downarrow$	2^{-1}		2^{-2}		2^{-3}		2^{-4}		2^{-5}		2^{-6}		2^{-7}	
1	2^{-2}	23	(23)	24	(24)	26	(26)	27	(27)	29	(29)	30	(30)	31	(31)
	2^{-3}	22	(22)	23	(23)	25	(25)	26	(29)	27	(29)	29	(29)	30	(30)
	2^{-4}	22	(22)	23	(23)	24	(24)	25	(25)	26	(26)	28	(38)	29	(36)
	2^{-5}	20	(20)	21	(21)	22	(22)	23	(23)	25	(25)	26	(36)	28	(30)
	2^{-6}	19	(19)	20	(20)	21	(21)	22	(21)	22	(22)	23	(24)	25	(26)
	2^{-7}	18	(18)	19	(19)	20	(20)	21	(20)	21	(21)	22	(24)	24	(25)
	2^{-8}	17	(17)	18	(18)	19	(19)	19	(19)	20	(20)	22	(23)	23	(25)
2	2^{-2}	28	(28)	32	(32)	34	(34)	36	(36)	40	(40)	43	(43)	49	(50)
	2^{-3}	31	(31)	34	(34)	35	(35)	36	(41)	38	(42)	39	(38)	38	(39)
	2^{-4}	30	(30)	32	(32)	33	(33)	34	(34)	34	(36)	35	(49)	35	(44)
	2^{-5}	29	(29)	31	(30)	32	(32)	33	(33)	34	(35)	34	(49)	34	(37)
	2^{-6}	28	(28)	30	(30)	31	(31)	32	(32)	32	(32)	33	(34)	31	(32)
	2^{-7}	27	(27)	29	(28)	30	(30)	30	(30)	30	(30)	29	(30)	29	(30)
	2^{-8}	25	(25)	26	(26)	27	(26)	28	(27)	26	(26)	26	(27)	26	(27)
3	2^{-2}	33	(34)	35	(40)	37	(42)	38	(38)	42	(42)	46	(46)	53	(55)
	2^{-3}	33	(33)	35	(35)	36	(37)	37	(49)	39	(47)	39	(39)	40	(40)
	2^{-4}	31	(31)	33	(33)	34	(34)	35	(47)	35	(38)	36	(53)	37	(48)
	2^{-5}	30	(30)	32	(32)	33	(33)	34	(34)	34	(35)	36	(53)	35	(39)
	2^{-6}	28	(28)	31	(31)	32	(32)	32	(32)	33	(33)	33	(34)	33	(35)
	2^{-7}	27	(27)	29	(29)	30	(30)	31	(31)	31	(31)	30	(30)	30	(30)
	2^{-8}	25	(26)	27	(27)	28	(28)	28	(28)	28	(28)	26	(28)	×	(×)
4	2^{-2}	25	(25)	28	(28)	30	(30)	31	(31)	32	(32)	31	(31)	32	(32)
	2^{-3}	24	(24)	26	(26)	27	(27)	29	(32)	30	(32)	30	(30)	31	(31)
	2^{-4}	24	(24)	25	(25)	26	(26)	27	(29)	28	(29)	29	(38)	31	(38)
	2^{-5}	23	(24)	24	(26)	25	(27)	26	(28)	27	(30)	28	(40)	29	(32)
	2^{-6}	21	(23)	22	(25)	23	(27)	24	(29)	25	(31)	26	(33)	26	(36)
	2^{-7}	20	(24)	21	(27)	22	(29)	23	(32)	24	(35)	25	(38)	25	(42)
	2^{-8}	19	(24)	20	(27)	21	(30)	22	(33)	23	(36)	23	(39)	24	(44)

Results from using direct or approximate solvers are reported side-by-side in Tab. 3.1, for ease of comparison. The two cases show a remarkably similar convergence behaviour, with the total number of iterations to convergence remaining substantially unchanged. The only notable disparities are observed for the most com-

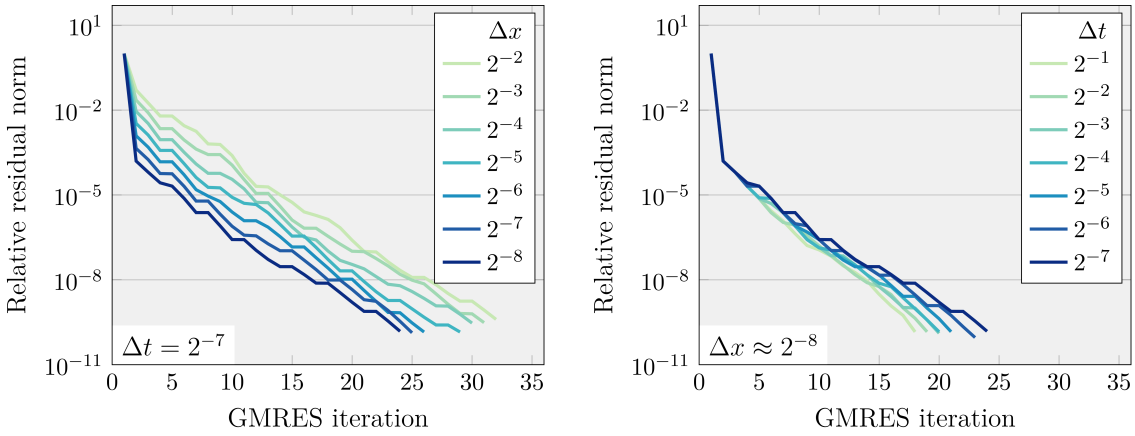


Figure 3.6: Details on residual evolution of GMRES applied to problem 1, using P_T^{-1} with ideal components as a right-preconditioner. Results on the left refer to simulations with fixed $\Delta t = 2^{-7}$ and varying Δx; results on the right are obtained by fixing $\Delta x \approx 2^{-8}$ and varying Δt. The set-up for the solvers is similar to the one used for Tab. 3.1.

plex problem 4, likely because the 15 AIR iterations used as an approximate solver for the space-time velocity block do not provide as accurate of an approximation as for problems 1 to 3. Even here, the increase in iterations compared with exact inner solves is nicely bounded, remaining below a factor of 2 with respect to the ideal case. This shows that the approximate solvers employed can be as effective as their direct counterparts, and gives an example of an effective time-parallel procedure which can work well in tandem with the preconditioner proposed.

We point out that, for the largest problems considered, the relevant systems have sizes $N_u = 526,338$, $N_p = 66,049$, and $N_t = 128$, for a total number of unknowns of $(N_u + N_p) \cdot N_t \approx 76 \cdot 10^6$; this further increases to $N_u = 5,775,362$, $N_p = 722,945$, and $N_t = 128$ (totalling $\approx 832 \cdot 10^6$) for problem 3. Compared to the noticeable size of the system, the total number of iterations to convergence reported in Tab. 3.1 remains small, regardless of mesh size and problem type, providing evidence of the optimal scalability of the preconditioners proposed. This is further confirmed by tracking in more detail the evolution of the residual as the GMRES iterations progress: an example of this is provided in Fig. 3.6, which shows reasonably uniform convergence profiles, independently of the choice of both Δx and Δt.

3.4.2.2 Dependence on Péclet number

When applied to the solution of Oseen equations, we observe that the performance of the preconditioner degrades as we increase the intensity of the advection field

Table 3.2: Number of iterations to convergence for problem 4 with different values of Pe. Dashes represent simulations that did not converge in the maximum prescribed number of iterations (100). Same set-up as for the ideal case in Tab. 3.1.

$\Delta t\rightarrow$ $\Delta x\downarrow$	2^{-4}					2^{-5}					2^{-6}					2^{-7}				
Pe$\rightarrow$	2^4	2^5	2^6	2^7	2^8	2^4	2^5	2^6	2^7	2^8	2^4	2^5	2^6	2^7	2^8	2^4	2^5	2^6	2^7	2^8
2^{-2}	32	60	//	//	//	32	57	//	//	//	31	55	//	//	//	31	54	//	//	//
2^{-3}	28	35	54	//	//	28	35	53	//	//	28	35	53	//	//	27	35	52	//	//
2^{-4}	27	32	40	55	//	27	32	39	55	//	25	31	38	53	//	25	31	38	52	//
2^{-5}	25	30	35	45	66	25	28	35	44	66	24	28	34	44	64	23	27	34	43	63
2^{-6}	24	27	32	39	49	23	27	32	38	49	23	27	32	37	48	22	26	30	37	47
2^{-7}	23	26	31	35	40	22	26	30	35	41	22	26	29	33	41	21	25	29	33	40
2^{-8}	22	25	29	32	37	21	25	29	32	37	21	25	28	32	37	20	24	27	32	37

imposed, as foreshadowed by the analysis in Sec. 3.3. This is shown in Tab. 3.2, where we progressively ramp up the value of Pe in problem 4, solving for various Δx: convergence fails for coarser spatial meshes and largest Péclet numbers. This hints at the necessity of either employing appropriately refined meshes, or relying on stabilisation techniques [118, Chap. 13.8] (which we did not apply in our analysis), whenever advection-dominated flows need to be resolved. The preconditioner for the single time-step case (3.38) was shown to suffer from a similar lack of robustness [40, Table 9.3], so it is somewhat expected that this weakness carries over to the whole space-time case. However, we point out that the performance of the preconditioner does not vary significantly as long as the *grid* Péclet number, $\tilde{\text{Pe}} := \Delta x \text{Pe}/L$ [118, Chap. 13.2], is kept constant. Unlike its *global* counterpart (3.86), this parameter provides a *local* measure of the dominance of advection over diffusion, and is useful in identifying conditions for which instabilities and oscillations might arise in the numerical solution. This parameter is kept fixed down each diagonal of Tab. 3.2, which shows only slow growth in the number of iterations to convergence as $\tilde{\text{Pe}}$ increases.

3.4.3 The nonlinear setting: incompressible Navier-Stokes

As stated in Sec. 3.1, Oseen equations (3.1) can be interpreted as a linearisation of the Navier-Stokes system (3.3). If we then introduce an *outer* solver to tackle the nonlinear advection term in Navier-Stokes, we can use our preconditioner to accelerate the solution of the corresponding *inner* (linearised) problems, much like what is done for the single time-step case in [40, Chap. 10]. We experiment on the performance of our preconditioner P_T (3.23) in this situation, and resolve the nonlinearity in (3.3)

via *Picard iterations* [115]. That is, we iteratively solve the linearised system

$$\left[\begin{array}{c|c} F_{\boldsymbol{u}}((\boldsymbol{u})^j) & B^T \\ \hline B & \mathbf{0} \end{array}\right] \left[\begin{array}{c} (\boldsymbol{u})^{j+1} \\ \hline (\boldsymbol{p})^{j+1} \end{array}\right] = \left[\begin{array}{c} \boldsymbol{f} \\ \hline \mathbf{0} \end{array}\right], \tag{3.87}$$

until convergence is reached. Here, $(\boldsymbol{u})^{j+1}$ and $(\boldsymbol{p})^{j+1}$ denote the whole space-time velocity and pressure solutions at the $j+1$-th Picard iteration, while we have made explicit the dependence of the space-time velocity block $F_{\boldsymbol{u}} = F_{\boldsymbol{u}}(\boldsymbol{w})$ on the advection field, which we substitute with the approximate solution at the previous iteration $(\boldsymbol{u})^j$.

Linear and nonlinear convergence obtained by applying a Picard linearisation (3.87) to the problems in Sec. 3.4.1 are reported in Tab. 3.3. Especially for problem 1, both the inner and outer solve iterations remain $\mathcal{O}(1)$, independently on the spatial and temporal mesh spacing. Similar results hold for the inflow-outflow problem problem 3, but here the average number of inner GMRES iterations is larger. In the specific case of a very large time-step relative to the spatial mesh, the number of inner iterations can be a fair amount larger. However, this is not necessarily surprising, as problem 3 offers more complex dynamical behaviour and we are resolving

Table 3.3: Convergence results for Navier-Stokes with the set-up of problems 1 and 3 (Reynolds number = 1 in both cases). Picard iterations were used as the outer nonlinear solver, which achieves convergence with a tolerance of 10^{-9}. Inside each column, the number of outer iterations to convergence is reported on the left, while the average number of inner GMRES iterations per outer Picard iteration appears on the right (in brackets). The inner GMRES solver uses the same set-up used for recovering the results in the left column of Tab. 3.1. Notice problem 2 is missing: this is because the nonlinearity does not affect the analytical solution for Poiseuille, and hence results are not particularly informative in this case (convergence occurs at the very first Picard iteration in almost all of the experiments). Also problem 4 is missing, since it already represents a linearisation of Navier-Stokes applied to problem 1.

Pb	$\Delta t\rightarrow$ $\Delta x\downarrow$	2^{-1}		2^{-2}		2^{-3}		2^{-4}		2^{-5}		2^{-6}		2^{-7}	
1	2^{-2}	5	(11.6)	4	(14.75)	4	(14.25)	4	(14.50)	4	(13.50)	4	(13.25)	4	(13.0)
	2^{-3}	4	(13.5)	4	(13.25)	4	(13.25)	4	(13.00)	4	(12.50)	4	(12.25)	5	(9.40)
	2^{-4}	4	(12.5)	4	(12.50)	4	(12.25)	4	(12.00)	4	(11.50)	5	(8.800)	5	(8.60)
	2^{-5}	4	(11.5)	4	(11.00)	4	(11.50)	4	(10.75)	5	(8.000)	5	(8.200)	4	(9.25)
	2^{-6}	4	(10.5)	4	(10.00)	4	(10.00)	4	(9.500)	4	(9.250)	4	(8.500)	4	(8.75)
	2^{-7}	4	(9.75)	4	(9.000)	4	(9.250)	5	(6.800)	4	(8.250)	4	(8.000)	4	(7.75)
	2^{-8}	3	(11.0)	3	(11.00)	3	(10.67)	3	(10.33)	3	(10.33)	4	(7.250)	4	(6.75)
3	2^{-2}	5	(19.80)	5	(18.20)	5	(18.00)	5	(18.00)	5	(18.40)	5	(19.00)	5	(21.40)
	2^{-3}	5	(21.80)	5	(19.60)	5	(17.80)	5	(17.00)	5	(16.80)	5	(16.40)	5	(16.20)
	2^{-4}	5	(22.60)	5	(20.60)	5	(18.80)	5	(16.80)	5	(15.60)	5	(14.80)	5	(14.20)
	2^{-5}	5	(26.60)	5	(22.80)	5	(20.60)	5	(17.60)	5	(16.20)	5	(15.80)	5	(14.20)
	2^{-6}	4	(36.00)	4	(31.75)	4	(28.25)	5	(18.00)	5	(17.20)	5	(16.00)	5	(14.80)
	2^{-7}	4	(43.00)	4	(37.00)	4	(31.75)	4	(24.75)	5	(18.40)	5	(18.60)	5	(15.00)
	2^{-8}	4	(52.50)	4	(46.25)	4	(33.50)	4	(28.25)	5	(22.40)	×	(×)	×	(×)

the nonlinearity over a very large time-step. This likely results in a very stiff and ill-conditioned linear system to solve for each nonlinear iteration. Nevertheless, the outer space-time Picard linearisation remains perfectly scalable.

3.4.4 Comparison with sequential time-stepping

In Sec. 3.2.1.1 we explain how the two main factors in analysing the effectiveness of our preconditioning strategy compared with classical time-stepping are given by the ratios $C_{F_{\boldsymbol{u}}}/(C_{\mathcal{F}_{\boldsymbol{u}}} N_t)$ and N_{it}^{ST}/N_{it}^0. The first ratio directly refers to the efficiency of the method chosen for the parallel-in-time integration of the velocity block $F_{\boldsymbol{u}}$: in this section we do not investigate the effectiveness of the various PinT methods available in the literature, but we refer to Chap. 2 for some examples on their typical performance, which is generally satisfactory for a parabolic problem such as the one considered here. Instead, we focus on the second ratio, since it is intrinsic to the preconditioning strategy proposed: in fact, it describes the overhead produced by tackling the whole space-time system at once, rather than via sequential time-stepping.

To give an indication of how N_{it}^0 and N_{it}^{ST} compare with each other when ideal solvers are employed, we collect convergence results from the application of (3.38) within the time-stepping procedure, for problems equivalent to those analysed in

Table 3.4: Ratio N_{it}^{ST}/N_{it}^0, where N_{it}^{ST} is taken from the ideal case in Tab. 3.1, while N_{it}^0 represents the average number of iterations to convergence per time-step, solving for (3.18) via forward substitution. For the latter, GMRES is used at each time-step, right-preconditioned with the single time-step counterpart of P_T, (3.38) with (3.39). Convergence is reached with a tolerance of $10^{-10}/\sqrt{N_t}$. The remaining options for the solvers involved are the same ones used to fill the left column in Tab. 3.1.

Pb	$\Delta t\rightarrow$ $\Delta x\downarrow$	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	Pb
1	2^{-2}	1.18	1.37	1.39	1.61	1.82	2.13	1.39	1.54	1.75	2.13	2.44	2.94	3
	2^{-3}	1.08	1.16	1.33	1.64	1.92	2.08	1.28	1.43	1.64	1.82	2.00	2.27	
	2^{-4}	1.08	1.22	1.41	1.56	1.79	2.00	1.20	1.32	1.43	1.59	1.79	2.08	
	2^{-5}	1.06	1.23	1.32	1.52	1.64	1.89	1.18	1.28	1.43	1.56	1.82	2.08	
	2^{-6}	1.08	1.18	1.25	1.32	1.47	1.67	1.18	1.27	1.37	1.56	1.85	2.27	
	2^{-7}	1.04	1.14	1.20	1.28	1.41	1.67	1.15	1.27	1.45	1.69	1.96	2.33	
	2^{-8}	1.03	1.11	1.15	1.28	1.56	1.82	1.16	1.30	1.52	1.75	1.92	×	
2	2^{-2}	1.43	1.61	1.82	2.13	2.50	3.13	1.28	1.43	1.54	1.69	1.82	2.13	4
	2^{-3}	1.35	1.43	1.61	1.82	2.08	2.27	1.14	1.20	1.39	1.69	1.85	2.08	
	2^{-4}	1.19	1.25	1.35	1.49	1.72	1.96	1.12	1.20	1.41	1.56	1.72	1.89	
	2^{-5}	1.15	1.19	1.28	1.47	1.67	2.00	1.19	1.27	1.37	1.49	1.59	1.69	
	2^{-6}	1.10	1.18	1.28	1.45	1.82	2.17	1.10	1.15	1.27	1.39	1.45	1.59	
	2^{-7}	1.10	1.20	1.30	1.56	1.89	2.44	1.06	1.10	1.22	1.35	1.49	1.59	
	2^{-8}	1.08	1.19	1.43	1.59	1.96	2.56	1.05	1.10	1.22	1.37	1.47	1.72	

Tab. 3.1. To make the comparison more fair, for the single time-step we ask for a stricter tolerance on convergence, scaled by a factor $\sqrt{N_t}$ with respect to the whole space-time case: in this way, the Euclidean norm of the final space-time residual is similar in the two cases. We report the resulting ratio N_{it}^{ST}/N_{it}^{0} in Tab. 3.4. We see that this sits between roughly 1 and 3, in most cases closer to one, but becoming larger as $\Delta t \to 0$, particularly $\Delta t \ll \Delta x$. This observation can be explained by pointing out that a large advantage of time-stepping is given by the fact that, at each iteration, a good initial guess is provided by the value of the solution at the previous time-step, which we exploit in our experiments: with a reduced step-size, it is safe to expect that also the solution varies less between one time-step and the next, improving the quality of such initial guess. Nevertheless, results demonstrate little-to-no degradation in convergence speed between the two approaches and, thus, minimal overhead in terms of iteration counts when applying block preconditioning techniques in an all-at-once fashion, when compared with sequential time-stepping.

3.5 Remarks on STBP for incompressible flow

The space-time preconditioning approach pursued in this chapter exploits the block structure of the target system *at the space-time level* in order to simplify the type of problem we need to (parallel-)integrate in time. The most demanding task of the resulting procedure reduces in fact to the solution of a single-variable, time-dependent advection-diffusion equation. This approach provides a different take on time parallelisation for incompressible flows, compared with what has typically been done in the literature [23,50,103,148,149]. To our knowledge, most of the latter relies on algorithms such as Parareal, which leave the underlying time-stepping procedure for solving (3.18) largely unchanged, and rather focus on building an overarching framework which breaks its innate sequentiality.

A number of arguments can be made for supporting the latter approach: most noticeably, time parallelisation is a relatively new concept, largely seen as an extra feature to include when one desires to gain some additional concurrency. In this sense, relying on a non-intrusive PinT algorithm has the appealing advantage of allowing to reuse already-existing time-integration schemes, out of the box. This is especially true in the case of complex systems of PDEs, for which time-stepping in itself can be a challenging task, often requiring the design of solvers *ad hoc*. Having to re-invent such solvers from scratch in order to achieve time parallelisation would then be detrimental.

Fortunately, the analysis conducted in this chapter provides evidence for stating that this needs not be the case. In fact, in Sec. 3.2.1 and 3.2.2 we have shown that, at least for the model problem considered, the very same concepts that guided the design of a block preconditioner for the single time-step case can be seamlessly extended to the space-time framework, achieving similar advantages and with a marginal overhead, as detailed in Sec. 3.4 and more specifically in Sec. 3.4.4. Indeed an interesting analogy can be drawn between the role of the preconditioner for the single time-step case (3.38) and our space-time preconditioner (3.23): the first provides an effective method for solving Oseen via sequential time stepping, so long as a fast solver for an elliptic operator is available; the second provides an effective method for solving Oseen all-at-once, so long as a fast (time-parallel) solver for a *parabolic* operator is available (and research on PinT schemes for this class of problems is already relatively mature: see for example [48, 55, 77, 138], as well as Chap. 2).

In the following chapter, we investigate how the concept of space-time block preconditioning can be extended to a more complicated problem, representing the target application of our project: magnetohydrodynamics.

Chapter 4

Applications to incompressible magnetohydrodynamics

Lying at the interface between the fields of electromagnetism and fluid dynamics, *Magnetohydrodynamics* (MHD) is responsible of modelling the behaviour of electrically charged fluids when subject to electromagnetic interactions. MHD models are used to describe a variety of fascinating phenomena, such as galaxy formation, the generation of solar flares, as well as their interaction with the magnetosphere of the Earth; our interest lies in more technical applications, linked to the development of effective magnetic confinement for the plasma within a fusion reactor. More specifically, our target model considers *resistive* plasmas: in this framework, it is possible for the topology of the magnetic field lines to change, and phenomena in which these break or coalescence can occur [66]. This represents a problem in the development of fusion reactors, as during these events some matter might escape the magnetic confinement, and end up colliding with the interior walls of the vessel, thus damaging them, losing heat in the process (see also Fig. 1.2), and ultimately undermining the possibility of sustaining a reaction for long periods of time. Due to the complexity and cost associated with setting up large-scale experiments, fusion researchers at our industrial partner institution, CCFE, rely extensively on the results from numerical simulations in guiding their design choices for properly controlling the reaction.

The corresponding system of PDEs is tightly coupled, non-symmetric, and presents nonlinearities which give rise to turbulent behaviour and spawn a range of phenomena interacting at various scales, both temporal and spatial: these are all characteristics which make its approximate solution particularly challenging. Nonetheless, there is great interest in developing strategies for tackling these problems effectively, as testified by the amount of research dedicated to the design of stable and efficient schemes and preconditioners for accelerating the solution of the discretised MHD system: see

for example [1,15,21,88,114,134,135], and the work this chapter mostly relies on, [25]. Given the difficulties already posed by having to solve this problem in a time-stepping framework, perhaps it is not surprising to see that attempts to include time parallelisation into this task are rare and far-between. To our knowledge, only [8,119,129] and more recently [128], have tested the application of a parallel-in-time algorithm (specifically, Parareal from Sec. 2.1.1) to MHD problems, showing modest speedups of 8-10× using hundreds of processors. It is precisely the complexity of the equations involved, however, that makes the development of effective parallelisation techniques to reduce time-to-solution particularly appealing.

In this chapter we describe how the concept of space-time block preconditioning developed in Chap. 3 can be extended to the framework of resistive, incompressible MHD, thus building evidence supporting the flexibility and broad spectrum of applicability of this technology, and showcasing an alternative approach to time parallelisation for plasma simulation. In designing our space-time block preconditioner for MHD, we follow similar steps as in the previous chapter, and start by identifying a suitable single time-step counterpart. The choice falls on the *Continuous Schur Complement preconditioner* introduced in [25, (3.32)]: its particularity is that it relies on the very same PCD operator (3.38) which is the foundation underpinning our STBP preconditioner for incompressible flow derived in Sec. 3.2.1. As such, when considering its extension to the whole space-time framework, we can directly reuse most of the technology developed in the previous chapter. We measure the performance of the preconditioner when applied to the space-time solution of some MHD model problems. The convergence results are in large part still preliminary, and additional work is required in order to perfect the final definition of the space-time preconditioner, but they already hint at its scalability and show that it compares well with its single time-step counterpart.

As the work described here represents a direct extension of the technique developed in Chap. 3, the structure of this chapter closely resembles that of the previous one. In Sec. 4.1 we introduce the resistive incompressible magnetohydrodynamic system used for our experiments, together with a brief derivation of the model and a justification for the choice of the specific formulation employed, and cover details regarding its space-time discretisation. Our preconditioning strategy is illustrated in Sec. 4.2, where we show how to extend the work done in [25] from the single time-step to the whole space-time setting. Finally, some results on the effectiveness of the preconditioner, both from a theoretical and an experimental point of view, are presented and discussed in Sec. 4.3 and Sec. 4.4.

4.1 Problem definition

The system we are interested in solving can be interpreted as an extension of the Navier-Stokes equations introduced in (3.3), which takes into account the interaction between flow and a magnetic field [66, Chap. 6]. The model is provided by

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t}(\boldsymbol{x},t) + (\boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla)\boldsymbol{u}(\boldsymbol{x},t) - \mu\nabla^2\boldsymbol{u}(\boldsymbol{x},t) \\ \qquad\qquad +\nabla p(\boldsymbol{x},t) - \nabla\cdot\mathcal{T}_M(\boldsymbol{B}(\boldsymbol{x},t)) &= \boldsymbol{f}(\boldsymbol{x},t) \\ \qquad\qquad -\nabla\cdot\boldsymbol{u}(\boldsymbol{x},t) &= 0 \qquad\qquad \text{in } \Omega\times(T_0,T], \\ \dfrac{\partial \boldsymbol{B}(\boldsymbol{x},t)}{\partial t} - \nabla\times(\boldsymbol{u}(\boldsymbol{x},t)\times\boldsymbol{B}(\boldsymbol{x},t)) \\ \qquad\qquad +\dfrac{\eta}{\mu_0}\nabla\times(\nabla\times\boldsymbol{B}(\boldsymbol{x},t)) &= \boldsymbol{0} \end{cases} \tag{4.1}$$

where η is the electric resistivity, and μ_0 is the magnetic permeability in the void, both considered constant in $\Omega\subset\mathbb{R}^2$. Compared to (3.3), the momentum equation here contains an extra term,

$$\mathcal{T}_M(\boldsymbol{B}) = \frac{1}{\mu_0}\boldsymbol{B}\otimes\boldsymbol{B} - \frac{1}{2\mu_0}\|\boldsymbol{B}\|^2 I, \tag{4.2}$$

representing the *magnetic stress tensor*, which models the action of the force exerted by the magnetic field onto the flow, due to the charged nature of the medium (I is the 2×2 identity matrix). On top of this, the additional equation in the system models the behaviour of the magnetic field $\boldsymbol{B}(\boldsymbol{x},t)$: in particular, notice how the flow is responsible for its advection, thus rendering the system fully coupled. This last PDE is derived using some well-known formulas for electromagnetism: see for example [83, Chap. 5-6] for their description. Specifically, we start from Maxwell-Faraday equation,

$$\frac{\partial \boldsymbol{B}(\boldsymbol{x},t)}{\partial t} = -\nabla\times\boldsymbol{E}(\boldsymbol{x},t), \tag{4.3}$$

which links the magnetic field to the electric field $\boldsymbol{E}(\boldsymbol{x},t)$. We then substitute the latter by relying on Ohm's law,

$$\boldsymbol{E}(\boldsymbol{x},t) = -\boldsymbol{u}(\boldsymbol{x},t)\times\boldsymbol{B}(\boldsymbol{x},t) + \eta\,\boldsymbol{j}(\boldsymbol{x},t), \tag{4.4}$$

where we introduce the current $\boldsymbol{j}(\boldsymbol{x},t)$. This in turn is related back to the magnetic field via Ampère's law

$$\boldsymbol{j}(\boldsymbol{x},t) = \frac{1}{\mu_0}\nabla\times\boldsymbol{B}(\boldsymbol{x},t), \tag{4.5}$$

and we can see that following this chain of substitutions results directly in the last equation of (4.1). The *resistive* nature of the plasma is dictated by the presence of

the current term in (4.4); by contrast, *ideal* plasmas are perfect conductors, and for those we would have $\eta = 0$: see also [66, Chap. 6.5] and [83, Chap. 10.2].

Notice that, by taking the divergence of the magnetic field equation, we recover

$$\frac{\partial(\nabla \cdot \boldsymbol{B}(\boldsymbol{x}, t))}{\partial t} = 0, \tag{4.6}$$

as the curl terms cancel out, since $\nabla \cdot (\nabla \times (*)) = 0$. Relationship (4.6) implies that the solenoidality of the magnetic field is maintained throughout the temporal evolution, provided $\boldsymbol{B}(\boldsymbol{x}, t)$ is initially solenoidal. We can use this property to rewrite the divergence of the magnetic stress tensor in a more familiar way:

$$-\nabla \cdot \mathcal{T}_M(\boldsymbol{B}) = -\frac{1}{\mu_0} \cancel{\boldsymbol{B}(\nabla \cdot \boldsymbol{B})} - (\boldsymbol{B} \cdot \nabla)\, \boldsymbol{B} + \frac{1}{2\mu_0} \nabla(\|\boldsymbol{B}\|^2) = \frac{1}{\mu_0} \boldsymbol{B} \times (\nabla \times \boldsymbol{B})\,, \tag{4.7}$$

which represents the action of the *Lorentz force*. In practice, however, ensuring that the magnetic field remains solenoidal when solving (4.1) numerically might pose some difficulties. One possible workaround consists in enforcing the constraint $\nabla \cdot \boldsymbol{B} = 0$ weakly using a Lagrangian multiplier [133], thus augmenting (4.1) with an additional variable which covers a similar role for the magnetic field as the pressure does for the flow. An alternative way to enforce solenoidality of $\boldsymbol{B}(\boldsymbol{x}, t)$ consists in imposing

$$\boldsymbol{B}(\boldsymbol{x}, \boldsymbol{t}) = \nabla \times \boldsymbol{A}(\boldsymbol{x}, t), \tag{4.8}$$

thus introducing the vector potential $\boldsymbol{A}(\boldsymbol{x}, t)$ for the magnetic field. In our analysis we pursue this latter approach, both because this follows the work done in [25], which our analysis heavily relies upon, and because it allows for some simplifications when dealing with 2D problems, as explained next.

In order to reformulate the system (4.1) in terms of $\boldsymbol{A}$, we employ the following relationship between the vector potential and the electric field

$$\boldsymbol{E}(\boldsymbol{x}, t) = -\nabla \phi(\boldsymbol{x}, t) - \frac{\partial \boldsymbol{A}(\boldsymbol{x}, t)}{\partial t}, \tag{4.9}$$

which holds for a given scalar electric potential $\phi(\boldsymbol{x}, t)$. Substituting eqs. (4.5), (4.8) and (4.9) into (4.4), provides the following PDE:

$$\frac{\partial \boldsymbol{A}(\boldsymbol{x}, t)}{\partial t} - \boldsymbol{u}(\boldsymbol{x}, t) \times (\nabla \times \boldsymbol{A}(\boldsymbol{x}, t)) + \frac{\eta}{\mu_0} \nabla \times (\nabla \times \boldsymbol{A}(\boldsymbol{x}, t)) + \nabla \phi(\boldsymbol{x}, t) = \boldsymbol{0}. \tag{4.10}$$

Given its definition in (4.8), we notice that adding a curl-free perturbation to $\boldsymbol{A}(\boldsymbol{x}, t)$ does not modify the resulting magnetic field $\boldsymbol{B}(\boldsymbol{x}, t)$. We can exploit this fact to get rid of the additional variable $\phi(\boldsymbol{x}, t)$: we do so by re-defining

$$\tilde{\boldsymbol{A}}(\boldsymbol{x}, t) = \boldsymbol{A}(\boldsymbol{x}, t) + \int_{T_0}^{T} \nabla \phi(\boldsymbol{x}, t)\, dt; \tag{4.11}$$

notice $\boldsymbol{B}(\boldsymbol{x},t) = \nabla \times \tilde{\boldsymbol{A}}(\boldsymbol{x},t) \equiv \nabla \times \boldsymbol{A}(\boldsymbol{x},t)$, since $\nabla \times (\nabla(*)) = \boldsymbol{0}$. From here on, with a slight abuse of notation, we drop the tilde on $\tilde{\boldsymbol{A}}(\boldsymbol{x},t)$ and refer to this modified vector potential instead. Substituting the new definition (4.11) in (4.10) simplifies the latter equation to

$$\frac{\partial \boldsymbol{A}(\boldsymbol{x},t)}{\partial t} - \boldsymbol{u}(\boldsymbol{x},t) \times (\nabla \times \boldsymbol{A}(\boldsymbol{x},t)) + \frac{\eta}{\mu_0} \nabla \times (\nabla \times \boldsymbol{A}(\boldsymbol{x},t)) = \boldsymbol{0}. \tag{4.12}$$

Still, $\boldsymbol{A}(\boldsymbol{x},t)$ remains defined up to a curl-free function: to fix it once and for all[1], we impose the extra condition $\nabla \cdot \boldsymbol{A} = 0$. Using the identity $\nabla \times (\nabla \times (*)) = \nabla(\nabla \cdot (*)) - \nabla^2(*)$, this allows us to rewrite

$$\frac{\partial \boldsymbol{A}(\boldsymbol{x},t)}{\partial t} - \boldsymbol{u}(\boldsymbol{x},t) \times (\nabla \times \boldsymbol{A}(\boldsymbol{x},t)) - \frac{\eta}{\mu_0} \nabla^2 \boldsymbol{A}(\boldsymbol{x},t) = \boldsymbol{0}. \tag{4.13}$$

When limiting ourselves to 2D geometries, it suffices to track the z-component of the vector potential,

$$\boldsymbol{A}(\boldsymbol{x},t) \underset{(\text{if } \boldsymbol{x} \in \mathbb{R}^2)}{=} \begin{bmatrix} 0 \\ 0 \\ A(\boldsymbol{x},t) \end{bmatrix} = A(\boldsymbol{x},t)\hat{\boldsymbol{k}} \quad \Longrightarrow \quad \boldsymbol{B}(\boldsymbol{x},t) = \begin{bmatrix} \frac{\partial A}{\partial y}(\boldsymbol{x},t) \\ -\frac{\partial A}{\partial x}(\boldsymbol{x},t) \\ 0 \end{bmatrix}, \tag{4.14}$$

($\hat{k}$ represents the unit vector in the z-direction), which allows us to effectively turn equation (4.13) into a scalar one:

$$\frac{\partial A(\boldsymbol{x},t)}{\partial t} + \boldsymbol{u}(\boldsymbol{x},t) \cdot \nabla A(\boldsymbol{x},t) - \frac{\eta}{\mu_0} \nabla^2 A(\boldsymbol{x},t) = 0. \tag{4.15}$$

To fully express system (4.1) in terms of $A(\boldsymbol{x},t)$, we also need to rewrite the Lorentz force term (4.7). This is done as follows:

$$\begin{aligned} \frac{1}{\mu_0} \boldsymbol{B}(\boldsymbol{x},t) \times (\nabla \times \boldsymbol{B}(\boldsymbol{x},t)) &= \frac{1}{\mu_0} (\nabla \times (A(\boldsymbol{x},t)\hat{\boldsymbol{k}})) \times (\nabla \times \nabla \times (A(\boldsymbol{x},t)\hat{\boldsymbol{k}})) \\ &= \frac{1}{\mu_0} (\nabla A(\boldsymbol{x},t) \times \hat{\boldsymbol{k}}) \times (-\hat{\boldsymbol{k}} \nabla^2 A(\boldsymbol{x},t)) \\ &= \frac{1}{\mu_0} \nabla A(\boldsymbol{x},t) \nabla^2 A(\boldsymbol{x},t). \end{aligned} \tag{4.16}$$

Changing (4.7) to (4.16), and substituting the magnetic field equation with (4.15),

[1] The procedure for fixing the choice of the vector potential goes under the name of *gauge fixing*. There is no unique way to achieve this: our approach, where we request $\boldsymbol{A}(\boldsymbol{x},t)$ to have null divergence, corresponds to imposing the *Coulomb gauge fixing condition* [83, Chap. 6.5].

we modify (4.1) to

$$\begin{cases} \dfrac{\partial \boldsymbol{u}(\boldsymbol{x},t)}{\partial t} + (\boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla)\boldsymbol{u}(\boldsymbol{x},t) - \mu\nabla^2\boldsymbol{u}(\boldsymbol{x},t) & & \\ \qquad +\nabla p(\boldsymbol{x},t) + \dfrac{1}{\mu_0}\nabla A(\boldsymbol{x},t)\nabla^2 A(\boldsymbol{x},t) &=& \boldsymbol{f}(\boldsymbol{x},t) \\ \qquad -\nabla\cdot\boldsymbol{u}(\boldsymbol{x},t) &=& 0 \\ \dfrac{\partial A(\boldsymbol{x},t)}{\partial t} + \boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla A(\boldsymbol{x},t) - \dfrac{\eta}{\mu_0}\nabla^2 A(\boldsymbol{x},t) &=& 0 \end{cases} \quad \text{in } \Omega\times(T_0,T], \tag{4.17}$$

Under this formulation, the Lorentz term presents some high-order derivatives (on top of the nonlinearity), which might make its treatment difficult depending on the chosen discretisation. We tackle this issue by introducing the current as an auxiliary variable: in the 2D case, this reduces to

$$\begin{aligned} \boldsymbol{j}(\boldsymbol{x},t) &= \frac{1}{\mu_0}\nabla\times(\nabla\times\boldsymbol{A}(\boldsymbol{x},t)) \underset{\left(\begin{smallmatrix}\text{if}\quad \boldsymbol{x}\in\mathbb{R}^2\\ \nabla\cdot A\equiv 0\end{smallmatrix}\right)}{=} j(\boldsymbol{x},t)\,\hat{\boldsymbol{k}}, \qquad \text{with} \\ j(\boldsymbol{x},t) &= \frac{1}{\mu_0}\nabla^2 A(\boldsymbol{x},t); \end{aligned} \tag{4.18}$$

(notice we used (4.5) once again). We are now ready to write our target system of equations:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}(\boldsymbol{x},t)}{\partial t} + (\boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla)\boldsymbol{u}(\boldsymbol{x},t) - \mu\nabla^2\boldsymbol{u}(\boldsymbol{x},t) & & \\ \qquad +\nabla p(\boldsymbol{x},t) + \nabla A(\boldsymbol{x},t) j(\boldsymbol{x},t) &=& \boldsymbol{f}(\boldsymbol{x},t) \\ \qquad -\nabla\cdot\boldsymbol{u}(\boldsymbol{x},t) &=& 0 \\ \qquad j(\boldsymbol{x},t) - \dfrac{1}{\mu_0}\nabla^2 A(\boldsymbol{x},t) &=& 0 \\ \dfrac{\partial A(\boldsymbol{x},t)}{\partial t} + \boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla A(\boldsymbol{x},t) - \dfrac{\eta}{\mu_0}\nabla^2 A(\boldsymbol{x},t) &=& -E(\boldsymbol{x},t) \end{cases} \quad \text{in } \Omega\times(T_0,T], \tag{4.19}$$

where we also introduced a right-hand side into the vector potential equation[2]: $E(\boldsymbol{x},t)$ represents the intensity of an applied external electric field (along the z-direction).

Next, we proceed to close the system with opportune initial and boundary conditions. For those on velocity, we reuse much of the notation introduced in Sec. 3.1, and particularly (3.5). Similarly, the initial condition prescribed for the vector potential is in the form

$$A(\boldsymbol{x},T_0) = \bar{A}^0(\boldsymbol{x}) \quad \text{on } \Omega, \tag{4.20}$$

[2]Notice we could have used (4.18) to substitute $j(\boldsymbol{x},t)$ into the equation for the vector potential. Not applying this substitution, however, has the advantage of giving the 2×2 block in the bottom-right of (4.26) an upper-triangular structure, with a standard parabolic operator in its diagonal, which makes it simpler to treat.

while its boundary conditions are denoted as

$$\begin{array}{rcll} A(\boldsymbol{x},t) & = & A_D(\boldsymbol{s},t) & \text{on } \Gamma_D^A \times (T_0,T], \quad \text{and} \\ \frac{\eta}{\mu_0}\nabla A(\boldsymbol{s},t)\cdot\boldsymbol{n}(\boldsymbol{s}) & = & h(\boldsymbol{s},t) & \text{on } \Gamma_N^A \times (T_0,T], \end{array} \tag{4.21}$$

with $A_D(\boldsymbol{s},t)$, $h(\boldsymbol{s},t)$ and $\bar{A}^0(\boldsymbol{x})$ being some known functions, while Γ_D^A and Γ_N^A identify the Dirichlet and Neumann parts of the boundary for $A(\boldsymbol{x},t)$, respectively, (which might in general differ from $\Gamma_D^{\boldsymbol{u}}$ and $\Gamma_N^{\boldsymbol{u}}$).

In the same fashion, we inherit the notation in (3.6) and (3.7) to describe the functional spaces where the velocity and pressure variables live in, and we introduce two more for the vector potential and current intensity variables, namely:

$$A(\cdot,t) \in C(\Omega) \equiv H^1_{\Gamma_D^A}(\Omega), \qquad \text{and} \qquad j(\cdot,t) \in D(\Omega) \equiv L^2(\Omega). \tag{4.22}$$

With this, we can provide the weak formulation for (4.19): we seek $[\boldsymbol{u},p,j,A]^T \in Y(\Omega) = V(\Omega)\times Q(\Omega)\times D(\Omega)\times C(\Omega)$ such that

$$\left\{\begin{array}{rcl} \dfrac{\partial}{\partial t}\langle \boldsymbol{u},\boldsymbol{v}\rangle + \langle(\boldsymbol{u}\cdot\nabla)\boldsymbol{u},\boldsymbol{v}\rangle + \mu\langle\nabla\boldsymbol{u},\nabla\boldsymbol{v}\rangle & & \\ \qquad -\langle p,\nabla\cdot\boldsymbol{v}\rangle + \langle j\nabla A,\boldsymbol{v}\rangle & = & \langle \boldsymbol{f},\boldsymbol{v}\rangle + \displaystyle\int_{\Gamma_N^{\boldsymbol{u}}} \boldsymbol{g}\cdot\boldsymbol{v} \\ \qquad -\langle q,\nabla\cdot\boldsymbol{u}\rangle & = & 0 \\ \langle j,l\rangle + \langle\nabla A,\nabla l\rangle - \displaystyle\int_{\Gamma_D^A}\nabla A\cdot\boldsymbol{n}l & = & \displaystyle\int_{\Gamma_N^A} hl \\ \dfrac{\partial}{\partial t}\langle A,c\rangle + \langle\boldsymbol{u}\cdot\nabla A,c\rangle + \dfrac{\eta}{\mu_0}\langle\nabla A,\nabla c\rangle & = & -\langle E,c\rangle + \displaystyle\int_{\Gamma_N^A} hc \end{array}\right. , \tag{4.23}$$

for each $[\boldsymbol{v}(\boldsymbol{x}),q(\boldsymbol{x}),l(\boldsymbol{x}),c(\boldsymbol{x})]^T \in Y(\Omega)$, and for each instant $t \in (T_0,T]$. (In order to reduce notation, in (4.23) we don't report explicitly the dependence of the variables on $\boldsymbol{x}$ and t.) We express the weak formulation more compactly by introducing the nonlinear operator $\mathcal{N}(\boldsymbol{z})$, which maps $\boldsymbol{z} \in Y(\Omega)$ to the residual of (4.19), so that (4.23) can be rewritten as: find $\boldsymbol{z} \in Y(\Omega)$ such that

$$\langle \mathcal{N}(\boldsymbol{z}),\boldsymbol{y}\rangle = 0 \qquad \forall \boldsymbol{y} \in Y(\Omega). \tag{4.24}$$

To resolve the nonlinearity in (4.23), we use Newton's method: that is, we iteratively solve the linearised variational problem

$$\begin{array}{ll} \left\langle \mathcal{J}_{\mathcal{N}}(\boldsymbol{z}^k)\delta\boldsymbol{z}^k,\boldsymbol{y}\right\rangle = -\left\langle \mathcal{N}(\boldsymbol{z}^k),\boldsymbol{y}\right\rangle, & \forall \boldsymbol{y} \in Y(\Omega) \\ \boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \delta\boldsymbol{z}^k & \end{array} \tag{4.25}$$

for the perturbation $\delta \boldsymbol{z}^k$, at each iteration k. Here, $\mathcal{J}_{\mathcal{N}}(\boldsymbol{z})$ denotes the Jacobian of the nonlinear form $\mathcal{N}$, evaluated at $\boldsymbol{z}$. This is a linear tensor operator in the form

$$\mathcal{J}_{\mathcal{N}}\left(\begin{bmatrix} \boldsymbol{u} \\ p \\ j \\ A \end{bmatrix}\right) := \left[\begin{array}{c|c|c|c} \dfrac{\partial}{\partial t} + \boldsymbol{u}\cdot\nabla + \nabla\boldsymbol{u} - \mu\nabla^2 & \nabla & \nabla A & j\nabla \\ \hline -\nabla\cdot & & & \\ \hline & & \mathcal{I} & -\dfrac{1}{\mu_0}\nabla^2 \\ \hline \nabla A\cdot & & & \dfrac{\partial}{\partial t} + \boldsymbol{u}\cdot\nabla - \dfrac{\eta}{\mu_0}\nabla^2 \end{array}\right], \tag{4.26}$$

whose discretisation is described in the following section.

We point out that resorting to Newton's method as a nonlinear solver is by no means necessary, and that our choice mostly stems from the desire of testing the efficacy of our space-time block preconditioners with solvers alternative to Picard (already used in Sec. 3.4.3), as well as the interest in following the setup presented in [25]. Although not investigated in our work, an appealing alternative in this sense could be provided by the *minimal decoupling Picard* iterations outlined in [88]: by dropping the $\nabla A\cdot$ term in the bottom-left block of (4.26), this scheme renders the target operator block upper-triangular, thus making it more straightforward to invert, and in particular simplifying most of the discussion in Sec. 4.2.

4.1.1 Space-time discretisation

For discretising the system (4.19), we follow the same approach used for (3.1): temporal derivatives are approximated using implicit Euler on a uniform grid, while for the spatial discretisation we use a Finite Element approach. In addition to the discrete velocity and pressure function spaces introduced in (3.10), we also consider

$$C_h(\Omega) := \operatorname{span}\{\zeta_0(\boldsymbol{x}), \dots, \zeta_{N_A-1}(\boldsymbol{x})\}, \quad \text{and} \tag{4.27a}$$

$$D_h(\Omega) := \operatorname{span}\{\xi_0(\boldsymbol{x}), \dots, \xi_{N_j-1}(\boldsymbol{x})\}, \tag{4.27b}$$

approximations of $C(\Omega)$ and $D(\Omega)$, with N_A and N_j degrees of freedom, respectively, so that functions in (4.27) can be represented by vectors $\boldsymbol{A} \in \mathbb{R}^{N_A}$ and $\boldsymbol{j} \in \mathbb{R}^{N_j}$ according to

$$A(\boldsymbol{x}) = \sum_{i=0}^{N_A-1} [\boldsymbol{A}]_i \, \zeta_i(\boldsymbol{x}) \qquad \text{and} \qquad j(\boldsymbol{x}) = \sum_{i=0}^{N_j-1} [\boldsymbol{j}]_i \, \xi_i(\boldsymbol{x}). \tag{4.28}$$

With these, we are ready to describe the relevant Galerkin matrices composing the discretised version of the linearised system (4.25). We begin with the definition of the spatial operators of interest, proceeding then to opportunely combine them in order to create a discretisation of the space-time operators appearing in (4.26).

Starting from the momentum equation, we have that the linearisation of the advection term $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$ produces the operator

$$\left[\tilde{\mathcal{W}}_{\boldsymbol{u}}(\boldsymbol{u}(\boldsymbol{x}))\right]_{m,n=0}^{N_{\boldsymbol{u}}-1} \coloneqq \int_\Omega \boldsymbol{\phi}_m(\boldsymbol{x})\cdot\left[(\boldsymbol{u}(\boldsymbol{x})\cdot\nabla)\,\boldsymbol{\phi}_n(\boldsymbol{x}) + (\boldsymbol{\phi}_n(\boldsymbol{x})\cdot\nabla)\,\boldsymbol{u}(\boldsymbol{x})\right]\,d\boldsymbol{x}; \quad (4.29)$$

while linearising the Lorentz force term with respect to the vector potential and the current gives instead

$$[\mathcal{Z}_j(A(\boldsymbol{x}))]_{m,n=0}^{N_{\boldsymbol{u}}-1,N_j-1} \coloneqq \int_\Omega \nabla A(\boldsymbol{x})\cdot\boldsymbol{\phi}_m(\boldsymbol{x})\xi_n(\boldsymbol{x})\,d\boldsymbol{x}, \qquad \text{and} \quad (4.30a)$$

$$[\mathcal{Z}_A(j(\boldsymbol{x}))]_{m,n=0}^{N_{\boldsymbol{u}}-1,N_A-1} \coloneqq \int_\Omega j(\boldsymbol{x})\boldsymbol{\phi}_m(\boldsymbol{x})\cdot\nabla\zeta_n(\boldsymbol{x})\,d\boldsymbol{x}. \quad (4.30b)$$

For the definition of $j(\boldsymbol{x},t)$, the relevant operators appearing are the current mass matrix,

$$[\mathcal{M}_j]_{m,n=0}^{N_j-1} \coloneqq \int_\Omega \xi_m(\boldsymbol{x})\xi_n(\boldsymbol{x})\,d\boldsymbol{x}, \quad (4.31)$$

and the mixed current-vector potential stiffness matrix (scaled by μ_0),

$$[\mathcal{K}_{jA}]_{m,n=0}^{N_j-1,N_A-1} \coloneqq \frac{1}{\mu_0}\int_\Omega \nabla\xi_m(\boldsymbol{x})\cdot\nabla\zeta_n(\boldsymbol{x})\,d\boldsymbol{x} - \frac{1}{\mu_0}\int_{\Gamma_D^A} \xi_m(\boldsymbol{s})\,\nabla\zeta_n(\boldsymbol{s})\cdot\boldsymbol{n}(\boldsymbol{s})\,d\boldsymbol{s}. \quad (4.32)$$

Lastly, for the vector potential equation, we introduce its mass, stiffness, and advection matrices

$$[\mathcal{M}_A]_{m,n=0}^{N_A-1} \coloneqq \int_\Omega \zeta_m(\boldsymbol{x})\cdot\zeta_n(\boldsymbol{x})\,d\boldsymbol{x}, \quad (4.33a)$$

$$[\mathcal{K}_A]_{m,n=0}^{N_A-1} \coloneqq \int_\Omega \nabla\zeta_m(\boldsymbol{x})\cdot\nabla\zeta_n(\boldsymbol{x})\,d\boldsymbol{x}, \quad \text{and} \quad (4.33b)$$

$$[\mathcal{W}_A(\boldsymbol{u}(\boldsymbol{x}))]_{m,n=0}^{N_A-1} \coloneqq \int_\Omega \zeta_m(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x})\cdot\nabla\zeta_n(\boldsymbol{x})\,d\boldsymbol{x}. \quad (4.33c)$$

The latter is one of the two terms stemming from the linearisation of $\boldsymbol{u}\cdot\nabla A$; the other is given by

$$[\mathcal{Y}(A(\boldsymbol{x}))]_{m,n=0}^{N_A-1,N_{\boldsymbol{u}}-1} \coloneqq \int_\Omega \zeta_m(\boldsymbol{x})\nabla A(\boldsymbol{x})\cdot\boldsymbol{\phi}_n(\boldsymbol{x})\,d\boldsymbol{x}. \quad (4.34)$$

The relevant discrete space-time operators are then assembled using the matrices defined above as building blocks. In particular, combining the linearised advection operator (4.29) with the velocity mass and stiffness matrices (3.12), we produce

$$\mathcal{F}_{\boldsymbol{u}}(\boldsymbol{u}(\boldsymbol{x})) \coloneqq \frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t} + \tilde{\mathcal{W}}_{\boldsymbol{u}}(\boldsymbol{u}(\boldsymbol{x})) + \mu\mathcal{K}_{\boldsymbol{u}}. \quad (4.35)$$

This appears in the discrete space-time velocity matrix

$$F_{\boldsymbol{u}}(\boldsymbol{u}(\boldsymbol{x},t)) \coloneqq \begin{bmatrix} \mathcal{F}_{\boldsymbol{u}}(\boldsymbol{u}(\boldsymbol{x},t_1)) & & \\ -\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t} & \ddots & \\ & \ddots & \mathcal{F}_{\boldsymbol{u}}(\boldsymbol{u}(\boldsymbol{x},t_{N_t})) \end{bmatrix}, \tag{4.36}$$

which shares the same block bi-diagonal structure as (3.22). Similarly, we combine (4.33) to create

$$\mathcal{F}_A(\boldsymbol{u}(\boldsymbol{x})) \coloneqq \frac{\mathcal{M}_A}{\Delta t} + \mathcal{W}_A(\boldsymbol{u}(\boldsymbol{x})) + \frac{\eta}{\mu_0}\mathcal{K}_A, \tag{4.37}$$

which we use in the definition of the discrete space-time vector potential operator

$$F_A(\boldsymbol{u}(\boldsymbol{x},t)) \coloneqq \begin{bmatrix} \mathcal{F}_A(\boldsymbol{u}(\boldsymbol{x},t_1)) & & \\ -\frac{\mathcal{M}_A}{\Delta t} & \ddots & \\ & \ddots & \mathcal{F}_A(\boldsymbol{u}(\boldsymbol{x},t_{N_t})) \end{bmatrix}. \tag{4.38}$$

The remaining operators do not involve any temporal derivative, and as a consequence their space-time counterparts preserve a block diagonal structure, much like operator B in (3.22). These are given by:

$$\begin{aligned} Z_j(A(\boldsymbol{x},t)) &\coloneqq \operatorname*{diag}_{k=1,\dots,N_t}(\mathcal{Z}_A(A(\boldsymbol{x},t_k))), \\ Z_A(j(\boldsymbol{x},t)) &\coloneqq \operatorname*{diag}_{k=1,\dots,N_t}(\mathcal{Z}_j(j(\boldsymbol{x},t_k))), \\ M_j &\coloneqq \operatorname*{diag}_{N_t}(\mathcal{M}_j), \\ K_{jA} &\coloneqq \operatorname*{diag}_{N_t}(\mathcal{K}_{jA}), \qquad \text{and} \\ Y(A(\boldsymbol{x},t)) &\coloneqq \operatorname*{diag}_{k=1,\dots,N_t}(\mathcal{Y}(A(\boldsymbol{x},t_k))), \end{aligned} \tag{4.39}$$

where $\operatorname{diag}_{k=1,\dots,N}((*)_k)$ denotes a $N \times N$ block diagonal matrix, whose diagonal blocks are taken in order from the indexed operators $\{(*)_k\}_{k=1}^N$.

In turn, the space-time operators eqs. (4.36), (4.38) and (4.39) compose the discrete version of the Jacobian (4.26),

$$J_{\mathcal{N}}\left(\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \\ \boldsymbol{j} \\ \boldsymbol{A} \end{bmatrix}\right) \coloneqq \begin{bmatrix} F_{\boldsymbol{u}}(\boldsymbol{u}) & B^T & Z_j(\boldsymbol{A}) & Z_A(\boldsymbol{j}) \\ B & & & \\ & & M_j & K_{jA} \\ Y(\boldsymbol{A}) & & & F_A(\boldsymbol{u}) \end{bmatrix}, \tag{4.40}$$

where $\boldsymbol{u}$, $\boldsymbol{p}$, $\boldsymbol{j}$, and $\boldsymbol{A}$ represent the whole space-time approximations to the unknowns of (4.19), so that, for example, $\boldsymbol{A} = \left[(\boldsymbol{A}^1)^T, (\boldsymbol{A}^2)^T, \dots, (\boldsymbol{A}^{N_t})^T\right]^T$, with $\boldsymbol{A}^i \approx A(\boldsymbol{x}, t_i)$ according to (4.28).

To complete the description of the discretisation of the linearised system (4.25), we also need to define the discrete counterpart of the residual $\mathcal{N}(\boldsymbol{z}(\boldsymbol{x},t))$, which we denote as $\boldsymbol{N}(\boldsymbol{z})$, with $\boldsymbol{z} = [\boldsymbol{u}^T, \boldsymbol{p}^T, \boldsymbol{j}^T, \boldsymbol{A}^T]^T$. This is also expressed in a space-time block fashion:

$$\boldsymbol{N}(\boldsymbol{z}) = \left[\boldsymbol{N}_{\boldsymbol{u}}(\boldsymbol{z})^T, \boldsymbol{N}_p(\boldsymbol{z})^T, \boldsymbol{N}_j(\boldsymbol{z})^T, \boldsymbol{N}_A(\boldsymbol{z})^T\right]^T, \tag{4.41}$$

where for each variable $(*)$ in the system, $\boldsymbol{N}_{(*)}(\boldsymbol{z})$ is in itself a space-time vector, created by collating the spatial vectors $\boldsymbol{N}^k_{(*)}(\boldsymbol{z})$ for each instant $k = 1, \dots, N_t$. These are given by

$$\begin{aligned}
\boldsymbol{N}^k_{\boldsymbol{u}}(\boldsymbol{z}) :=& \frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\left(\boldsymbol{u}^k - \boldsymbol{u}^{k-1}\right) + \mu\mathcal{K}_{\boldsymbol{u}}\boldsymbol{u}^k + \mathcal{B}\boldsymbol{p}^k - \boldsymbol{f}^k \\
&+ \left[\int_\Omega \left((\boldsymbol{u}^k \cdot \nabla)\boldsymbol{u}^k + \boldsymbol{j}^k \nabla \boldsymbol{A}^k\right) \cdot \boldsymbol{\phi}_i\right]_{i=0}^{N_{\boldsymbol{u}}-1}, && \text{(4.42a)} \\
\boldsymbol{N}^k_p(\boldsymbol{z}) :=& \mathcal{B}\boldsymbol{u}^k, && \text{(4.42b)} \\
\boldsymbol{N}^k_j(\boldsymbol{z}) :=& \mathcal{M}_j\boldsymbol{j}^k + \mathcal{K}_{jA}\boldsymbol{A}^k - \boldsymbol{h}^k, \qquad \text{and} && \text{(4.42c)} \\
\boldsymbol{N}^k_A(\boldsymbol{z}) :=& \frac{\mathcal{M}_A}{\Delta t}\left(\boldsymbol{A}^k - \boldsymbol{A}^{k-1}\right) + \frac{\eta}{\mu_0}\mathcal{K}_A\boldsymbol{A}^k + \boldsymbol{E}^k + \left[\int_\Omega \boldsymbol{u}^k \cdot \nabla \boldsymbol{A}^k \zeta_i\right]_{i=0}^{N_A-1}, && \text{(4.42d)}
\end{aligned}$$

with $\boldsymbol{f}^k$ in (4.42a) defined as in (3.16), while for (4.42c) and (4.42d) we take

$$\begin{aligned}
[\boldsymbol{h}^k]_{i=0}^{N_j-1} &:= \int_{\Gamma_N^A} h(\boldsymbol{s}, t_k)\xi_i(\boldsymbol{s})\, d\boldsymbol{s}, \qquad \text{and} \\
[\boldsymbol{E}^k]_{i=0}^{N_A-1} &:= \int_\Omega E(\boldsymbol{x}, t_k)\zeta_i(\boldsymbol{x})\, d\boldsymbol{x} - \int_{\Gamma_N^A} h(\boldsymbol{s}, t_k)\zeta_i(\boldsymbol{s})\, d\boldsymbol{s}.
\end{aligned} \tag{4.43}$$

Notice that, similarly to (3.19), for $k = 1$ we modify $\boldsymbol{N}^1_{\boldsymbol{u}}(\boldsymbol{z})$ and $\boldsymbol{N}^1_A(\boldsymbol{z})$ by considering the discretised initial conditions $\bar{\boldsymbol{u}}^0$ and $\bar{\boldsymbol{A}}^0$ instead of $\boldsymbol{u}^0$ and $\boldsymbol{A}^0$, respectively. With the relevant quantities defined, we can finally write the target discrete system for the k-th Newton iteration:

$$J_{\mathcal{N}}(\boldsymbol{z}^k)\delta\boldsymbol{z}^k = -\boldsymbol{N}(\boldsymbol{z}^k). \tag{4.44}$$

Solving systems in the form of (4.44) is the main target of the study in this chapter. To do so effectively, and in a time-parallel fashion, we follow a strategy similar to the one pursued in Chap. 3. That is, we identify a preconditioner which has shown its efficacy when used within a time-stepping procedure, and extend it to cover the whole space-time case. The preconditioner of choice was initially developed by Cyr *et al.* in [25] for the single time-step system arising from the discretisation of (4.17). The main motivation behind this particular choice stems from the specific design of this

preconditioner: it acts by splitting the solution of the fully coupled problem into two separate sub-problems, one dealing with the coupling between velocity and pressure, the other with velocity and magnetism. The former is largely equivalent to the one we already encountered in Chap. 3, which conveniently allows us to reuse most of the techniques developed within that framework. More details are provided in the following section, where we describe the preconditioner introduced in [25], adapt it to our formulation of the IMHD equations (4.19) (which includes the extra variable $j(\boldsymbol{x}, t)$), and consider its application to the space-time setting.

4.2 Space-time block preconditioning for IMHD

In designing a right preconditioner for (4.40), we could proceed as in (3.24) and attempt to provide an approximation for the block upper-triangular factor in its block LU factorisation. This factorisation is given by

$$\begin{aligned} J_{\mathcal{N}} = &\begin{bmatrix} I & & & \\ BF_{\boldsymbol{u}}^{-1} & I & & \\ & & I & \\ YF_{\boldsymbol{u}}^{-1} & -YF_{\boldsymbol{u}}^{-1}B^T S_p^{-1} & -YF_{\boldsymbol{u}}^{-1}(I + B^T S_p^{-1} BF_{\boldsymbol{u}}^{-1}) Z_j M_j^{-1} & I \end{bmatrix} \\ &\cdot \begin{bmatrix} F_{\boldsymbol{u}} & B^T & Z_j & Z_A \\ & S_p & -BF_{\boldsymbol{u}}^{-1} Z_j & -BF_{\boldsymbol{u}}^{-1} Z_A \\ & & M_j & K_{jA} \\ & & & \bar{S}_A \end{bmatrix}, \end{aligned} \tag{4.45}$$

where from (4.40) we dropped the dependency of the operators on the unknowns, in order to further contract notation. Operator S_p is the space-time pressure Schur complement, which we already encountered in (3.25):

$$S_p \coloneqq -BF_{\boldsymbol{u}}^{-1} B^T, \tag{4.46}$$

whereas $\bar{S}_A$ is the space-time magnetic Schur complement

$$\bar{S}_A \coloneqq F_A - YF_{\boldsymbol{u}}^{-1}(I + B^T S_p^{-1} BF_{\boldsymbol{u}}^{-1})(Z_A - Z_j M_j^{-1} K_{jA}). \tag{4.47}$$

Herein lies the problem with this approach: approximating the inverse of the latter operator represents a particularly challenging task, not least because of the presence of S_p^{-1}, *nested* inside the magnetic Schur complement $\bar{S}_A$. Reordering the variables in the system results in different block LU factorisations, and hence different Schur complements, but this does not solve the problem of having to deal with nested inverse operators. Rather, the authors in [25] overcome this issue by starting from

an *approximate* LDU factorisation of (4.40): adapted to our formulation (4.19), and extended to the whole space-time case, this reads

$$
\begin{aligned}
J_{\mathcal{N}} &\approx \begin{bmatrix} F_{\boldsymbol{u}} & B^T & Z_j & Z_A \\ B & & & \\ & & M_j & K_{jA} \\ Y & \boxed{YF_{\boldsymbol{u}}^{-1}B^T} & & F_A \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} F_{\boldsymbol{u}} & & Z_j & Z_A \\ & I & & \\ & & M_j & K_{jA} \\ Y & & & F_A \end{bmatrix}}_{=:P_{\boldsymbol{u}jA}} \underbrace{\begin{bmatrix} F_{\boldsymbol{u}}^{-1} & & & \\ & I & & \\ & & I & \\ & & & I \end{bmatrix}}_{=:F_{\boldsymbol{u}I}^{-1}} \underbrace{\begin{bmatrix} F_{\boldsymbol{u}} & B^T & & \\ B & & & \\ & & I & \\ & & & I \end{bmatrix}}_{=:P_{\boldsymbol{u}p}}.
\end{aligned} \tag{4.48}
$$

Such approximation consists in including an additional term in the vector potential equation, identified by the highlighted block in (4.48). Although a valid physical intuition about the effect of this perturbation is somewhat hard to grasp, the analyses conducted in [25] and Sec. 4.3 hint at the fact that the perturbed matrix has a spectrum similar to that of the original Jacobian, making this a valid approximation for our purposes. Moreover, the resulting factorisation has the remarkable advantage of neatly splitting the original fully-coupled system into two simpler subsystems, denoted as $P_{\boldsymbol{u}jA}$ and $P_{\boldsymbol{u}p}$, thus removing the complexity of dealing with nested Schur complements. We can convince ourselves of this by noticing how the factors in (4.48) present some block rows/columns which contain only an identity on the main diagonal: this effectively isolates the influence of some variables in the system. In the following sections we analyse more in detail the single factors composing this factorisation, providing approximations to their respective Schur complements, with the goal of simplifying the application of their inverse. It is these simplifications that our space-time preconditioner ultimately depends upon.

4.2.1 Velocity-pressure coupling

The rightmost matrix $P_{\boldsymbol{u}p}$ in (4.48) considers only the coupling between the velocity and pressure variables: applying the inverse of this factor is equivalent to solving an incompressible flow problem, without considering the effect of the magnetic field. This system corresponds to a similar problem to the one we tackled in Chap. 3, the sole difference lying in the velocity advection operator $\mathcal{W}_{\boldsymbol{u}}$. In fact, comparing the definition of (4.29) with that of (3.14), we notice the former presents an extra term,

$$
[\Delta\mathcal{W}_{\boldsymbol{u}}(\boldsymbol{u}(\boldsymbol{x}))]_{m,n=0}^{N_{\boldsymbol{u}}-1} \coloneqq [(\tilde{\mathcal{W}}_{\boldsymbol{u}} - \mathcal{W}_{\boldsymbol{u}})(\boldsymbol{u}(\boldsymbol{x}))]_{m,n=0}^{N_{\boldsymbol{u}}-1} = \int_\Omega ((\boldsymbol{\phi}_n(\boldsymbol{x}) \cdot \nabla)\, \boldsymbol{u}(\boldsymbol{x})) \cdot \boldsymbol{\phi}_m(\boldsymbol{x})\, d\boldsymbol{x}, \tag{4.49}
$$

due to the Newton linearisation. Since in Sec. 3.2.1 we derived our space-time PCD preconditioner from a commutation argument between $\mathcal{F}_{\boldsymbol{u}}$ (which contains $\mathcal{W}_{\boldsymbol{u}}$) and an equivalent operator $\mathcal{F}_p$ acting on the pressure field, one would expect that modifications to $\mathcal{F}_{\boldsymbol{u}}$ should push us to change $\mathcal{F}_p$ as well, thus modifying the final form of the preconditioner. In practice, the extra term in (4.49) includes some interplay between different components of the velocity variable, and it is not straightforward to construct its equivalent on the pressure variable, as the latter is only a scalar field. For this reason, to our knowledge the common approach in the single time-step case [24, 39] is to just ignore this extra contribution on the definition of $\mathcal{F}_p$.

We pursue the same approach here, limiting ourselves to reusing the same preconditioner introduced in Chap. 3. Specifically, to simplify the solution of systems involving $P_{\boldsymbol{u}p}$, we consider its block LU factorisation,

$$P_{\boldsymbol{u}p} = \underbrace{\begin{bmatrix} I & & & \\ BF_{\boldsymbol{u}}^{-1} & I & & \\ & & I & \\ & & & I \end{bmatrix}}_{=:L_{\boldsymbol{u}p}} \underbrace{\begin{bmatrix} F_{\boldsymbol{u}} & B^T & & \\ & S_p & & \\ & & I & \\ & & & I \end{bmatrix}}_{=:U_{\boldsymbol{u}p}}, \tag{4.50}$$

and substitute its block upper-triangular term $U_{\boldsymbol{u}p}$ with the approximation

$$\tilde{U}_{\boldsymbol{u}p} \coloneqq \begin{bmatrix} F_{\boldsymbol{u}} & B^T & & \\ & \tilde{S}_p & & \\ & & I & \\ & & & I \end{bmatrix}, \tag{4.51}$$

where the space-time pressure Schur complement S_p is swapped for the operator $\tilde{S}_p \coloneqq -K_p F_p^{-1} M_p$ introduced in (3.37).

4.2.2 Velocity-magnetic field coupling

On the other hand, the leftmost matrix $P_{\boldsymbol{u}jA}$ in (4.48) considers only the coupling between velocity, current, and vector potential, ignoring the incompressibility constraint associated with the pressure variable. To find an effective way to approximately solve systems involving $P_{\boldsymbol{u}jA}$, we once again consider its block LU decomposition:

$$P_{\boldsymbol{u}jA} = \underbrace{\begin{bmatrix} I & & & \\ & I & & \\ & & I & \\ YF_{\boldsymbol{u}}^{-1} & & -YF_{\boldsymbol{u}}^{-1}Z_j M_j^{-1} & I \end{bmatrix}}_{=:L_{\boldsymbol{u}jA}} \underbrace{\begin{bmatrix} F_{\boldsymbol{u}} & & Z_j & Z_A \\ & I & & \\ & & M_j & K_{jA} \\ & & & S_A \end{bmatrix}}_{=:U_{\boldsymbol{u}jA}}. \tag{4.52}$$

Here, S_A represents another space-time magnetic field Schur complement[3], different from (4.47), and specific to this sub-problem. This is defined as:

$$S_A \coloneqq F_A - Y F_{\boldsymbol{u}}^{-1} \underbrace{(Z_A - Z_j M_j^{-1} K_{jA})}_{=:Z}, \tag{4.53}$$

where we compact the whole linearisation of the Lorentz force term[4] into

$$\begin{aligned} Z(j(\boldsymbol{x},t), A(\boldsymbol{x},t)) \coloneqq & Z_A(j(\boldsymbol{x},t)) - Z_j(A(\boldsymbol{x},t)) M_j^{-1} K_{jA} \\ = & \operatorname*{diag}_{k=1,\dots,N_t} \left(\mathcal{Z}(j(\boldsymbol{x},t_k), A(\boldsymbol{x},t_k)) \right), \end{aligned} \tag{4.55}$$

whose individual blocks are defined as

$$\mathcal{Z}(j(\boldsymbol{x}), A(\boldsymbol{x})) \coloneqq \mathcal{Z}_A(j(\boldsymbol{x})) - \mathcal{Z}_j(A(\boldsymbol{x})) \mathcal{M}_j^{-1} \mathcal{K}_{jA}. \tag{4.56}$$

The main goal of this section is to provide an adequate approximation to the space-time operator (4.53), and in particular its term $Y F_{\boldsymbol{u}}^{-1} Z$. Analogously to (3.27), this term is composed of blocks in the form

$$\mathcal{Y}_k \mathcal{F}_{\boldsymbol{u},k}^{-1} \left(\prod_{l=i}^{k-1} \left(\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t} \mathcal{F}_{\boldsymbol{u},l}^{-1} \right) \right) \mathcal{Z}_i^T, \quad \text{with} \quad \begin{cases} k = 1, \dots, N_t \\ i = 1, \dots, k \end{cases}, \tag{4.57}$$

where $\mathcal{Z}_i$, $\mathcal{Y}_i$ and $\mathcal{F}_{\boldsymbol{u},i}$ is shorthand notation for $\mathcal{Z}(j(\boldsymbol{x},t_i), A(\boldsymbol{x},t_i))$, $\mathcal{Y}(A(\boldsymbol{x},t_i))$ and $\mathcal{F}(\boldsymbol{u}(\boldsymbol{x},t_i))$, for some fixed fields $\boldsymbol{u}(\boldsymbol{x},t)$, $j(\boldsymbol{x},t)$, and $A(\boldsymbol{x},t)$. In [25] the authors make use of a small-commutator argument akin to the one exploited in Sec. 3.2.1, and assume that

$$\mathcal{M}_A^{-1} \mathcal{F}_{A,i} \mathcal{M}_A^{-1} \mathcal{Y}_k \approx \mathcal{M}_A^{-1} \mathcal{Y}_k \mathcal{M}_{\boldsymbol{u}}^{-1} \mathcal{F}_{\boldsymbol{u},i}, \qquad \forall i,k = 1, \dots, N_t. \tag{4.58}$$

Notice that the operator $\mathcal{F}_{\boldsymbol{u},i}$ contains the extra term (4.49) which does not find an equivalent in $\mathcal{F}_{A,i}$, but we accept this discrepancy under similar considerations as for

[3] More specifically, S_A simplifies $\bar{S}_A$, by dropping the contribution from the whole term $B F_{\boldsymbol{u}}^{-1} Z$. Considering the role covered by the single operators appearing in that term, one can interpret this simplification as assuming that the divergence (B) of the velocity field resulting from ($F_{\boldsymbol{u}}^{-1}$) the Lorentz force (Z) is null.

[4] Notice that in [25, (3.7)], the single time-step magnetic Schur complement assumes the form

$$\mathcal{S}_A = \mathcal{F}_A - \mathcal{Y} \mathcal{F}_{\boldsymbol{u}}^{-1} \mathcal{Z}, \tag{4.54}$$

where $\mathcal{Z}$ comes from the linearisation of (4.16). In virtue of our formulation (4.19), which explicitly considers the current as an additional variable, this term appears in a different form in (4.53), *i.e.*, it is further split into its two components $\mathcal{Z}_A$ and $-\mathcal{Z}_j \mathcal{M}_j^{-1} \mathcal{K}_{jA}$. Nonetheless, as it represents the same operator, we can still apply also to our context similar considerations used in [25] for approximating (4.54).

Sec. 4.2.1. Even ignoring such a discrepancy, the formula would not hold in general; nonetheless, similar assumptions are not uncommon when applying this argument. Following a procedure similar to the one described in Sec. 3.2.1, we recursively use assumption (4.58) to recover reasonable approximations to the blocks (4.57), and thus to the whole space-time operator (4.53). Left-multiplying (4.58) by $\mathcal{M}_A\mathcal{F}_{A,i}^{-1}\mathcal{M}_A$, and right-multiplying it by $\mathcal{F}_{\boldsymbol{u},i}^{-1}$, we recover

$$\mathcal{Y}_k\mathcal{F}_{\boldsymbol{u},i}^{-1} \approx \mathcal{M}_A\mathcal{F}_{A,i}^{-1}\mathcal{Y}_k\mathcal{M}_{\boldsymbol{u}}^{-1}, \qquad \forall i,k = 1,\dots,N_t. \tag{4.59}$$

Starting from $i = k$, continuing to right-multiply by $\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\mathcal{F}_{\boldsymbol{u},k-j}^{-1}$, increasing the index j from 0 until we reach $k - j = l$, repeatedly using (4.59) for each $i = k - j$, and finally right-multiplying by $\mathcal{Z}_l$, gives us the following approximation

$$\begin{aligned} &\mathcal{Y}_k\mathcal{F}_{\boldsymbol{u},k}^{-1}\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\mathcal{F}_{\boldsymbol{u},k-1}^{-1} \approx \mathcal{M}_A\mathcal{F}_{A,k}^{-1}\underbrace{\frac{\mathcal{Y}_k}{\Delta t}\mathcal{F}_{\boldsymbol{u},k-1}^{-1}}_{\approx\frac{\mathcal{M}_A}{\Delta t}\mathcal{F}_{A,k-1}^{-1}\mathcal{Y}_k\mathcal{M}_{\boldsymbol{u}}^{-1}} \\ \implies\quad &\mathcal{Y}_k\mathcal{F}_{\boldsymbol{u},k}^{-1}\left(\prod_{j=1}^{k-l}\left(\frac{\mathcal{M}_{\boldsymbol{u}}}{\Delta t}\mathcal{F}_{\boldsymbol{u},k-j}^{-1}\right)\right)\mathcal{Z}_l \approx \mathcal{M}_A\mathcal{F}_{A,k}^{-1}\left(\prod_{j=1}^{k-l}\left(\frac{\mathcal{M}_A}{\Delta t}\mathcal{F}_{A,k-j}^{-1}\right)\right)\mathcal{Y}_k\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{Z}_l, \end{aligned} \tag{4.60}$$

which we can apply to each of the blocks in (4.57). In turn, this identifies a first approximation for the space-time magnetic Schur complement (4.53):

$$\begin{aligned} &YF_{\boldsymbol{u}}^{-1}Z \approx M_AF_A^{-1}YM_{\boldsymbol{u}}^{-1}Z \\ \implies\quad &S_A \approx F_A - M_AF_A^{-1}YM_{\boldsymbol{u}}^{-1}Z = M_AF_A^{-1}\left(F_AM_A^{-1}F_A - YM_{\boldsymbol{u}}^{-1}Z\right), \end{aligned} \tag{4.61}$$

which is a direct space-time equivalent of the one proposed in [25, (3.24)] for the single time-step magnetic Schur complement at instant k:

$$\mathcal{S}_{A,k} \approx \mathcal{M}_A\mathcal{F}_{A,k}^{-1}\left(\mathcal{F}_{A,k}\mathcal{M}_A^{-1}\mathcal{F}_{A,k} - \mathcal{Y}_k\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{Z}_k\right), \qquad \forall k = 1,\dots,N_t. \tag{4.62}$$

An additional approximation is performed in [25] in order to further simplify the term $\mathcal{Y}_k\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{Z}_k$ in (4.62): this term has proven difficult to treat there, as it heavily disrupts the sparsity pattern of the first term $\mathcal{F}_{A,k}\mathcal{M}_A^{-1}\mathcal{F}_{A,k}$, rendering its solution via multigrid more challenging[5]. The justification behind the simplification proposed in [25] comes from analysing the continuous problem associated with inverting $P_{\boldsymbol{u}jA}$, and can be seamlessly extended to the whole space-time framework. In particular,

[5]In our formulation, an additional complication stems from the presence in $\mathcal{Z}_k$ of the inverse of the mass matrix $\mathcal{M}_j$, which would also need to be properly approximated (together with $\mathcal{M}_{\boldsymbol{u}}$).

solving for the velocity-magnetic field coupling (4.52) corresponds to finding a solution to the PDE

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \mu\nabla^2\boldsymbol{u} + \dfrac{1}{\mu_0}\boldsymbol{B}\times(\nabla\times\boldsymbol{B}) &= \boldsymbol{0} \\ \dfrac{\partial \boldsymbol{B}}{\partial t} - \nabla\times(\boldsymbol{u}\times\boldsymbol{B}) + \dfrac{\eta}{\mu_0}\nabla\times(\nabla\times\boldsymbol{B}) &= \boldsymbol{0} \end{cases} \quad \text{in} \quad \Omega\times(T_0,T], \tag{4.63}$$

obtained by removing the pressure variable from (4.1). Solving for the magnetic Schur complement is the discrete equivalent of using the first equation to express $\boldsymbol{u}$ in terms of $\boldsymbol{B}$, substituting it in the second equation, and recovering its solution $\boldsymbol{B}$. To find an approximation to (4.62), the authors in [25] perform a similar procedure, but at the continuous level, and consider a linearised version of (4.63). Using perturbation theory, they expand the solution

$$\boldsymbol{u}(\boldsymbol{x},t) = \boldsymbol{u}_0(\boldsymbol{x},t) + \boldsymbol{u}_1(\boldsymbol{x},t) \qquad \text{and} \qquad \boldsymbol{B}(\boldsymbol{x},t) = \boldsymbol{B}_0(\boldsymbol{x},t) + \boldsymbol{B}_1(\boldsymbol{x},t) \tag{4.64}$$

around a quiet state $\boldsymbol{u}_0(\boldsymbol{x},t)\equiv\boldsymbol{0}$ with a constant background magnetic field $\boldsymbol{B}_0(\boldsymbol{x},t)\equiv\boldsymbol{B}_0$. Positioning themselves in the hyperbolic limit, for which the second-order dissipative terms in (4.63) are neglected, and dropping high-order perturbations, they recover

$$\begin{cases} \dfrac{\partial \boldsymbol{u}_1}{\partial t} + \dfrac{1}{\mu_0}\boldsymbol{B}_0\times(\nabla\times\boldsymbol{B}_1) &= \boldsymbol{0} \\ \dfrac{\partial \boldsymbol{B}_1}{\partial t} - \nabla\times(\boldsymbol{u}_1\times\boldsymbol{B}_0) &= \boldsymbol{0} \end{cases} \quad \text{in} \quad \Omega\times(T_0,T]. \tag{4.65}$$

To find the solution $\boldsymbol{B}_1$, they proceed to take an extra time derivative on the second equation,

$$\frac{\partial^2\boldsymbol{B}_1}{\partial t^2} - \nabla\times\left(\frac{\partial\boldsymbol{u}_1}{\partial t}\times\boldsymbol{B}_0\right) - \nabla\times\left(\boldsymbol{u}_1\times\frac{\partial\boldsymbol{B}_0}{\partial t}\right) = 0, \tag{4.66}$$

and substitute back in the first equation, to recover

$$\begin{aligned} &\frac{\partial^2\boldsymbol{B}_1}{\partial t^2} - \nabla\times\left(\frac{1}{\mu_0}(\nabla\times\boldsymbol{B}_1)\times\boldsymbol{B}_0\times\boldsymbol{B}_0\right) = 0 \\ \implies\quad &\frac{\partial^2\boldsymbol{B}_1}{\partial t^2} - \frac{\|\boldsymbol{B}_0\|^2}{\mu_0}\nabla^2\boldsymbol{B}_1 = 0. \end{aligned} \tag{4.67}$$

This equation can be analogously expressed in terms of the vector potential,

$$\frac{\partial^2 A}{\partial t^2} - \frac{\|\boldsymbol{B}_0\|^2}{\mu_0}\nabla^2 A = 0, \tag{4.68}$$

since the discussion in Sec. 4.1 ensures that there is unique correspondence between A and its curl. The magnetic Schur complement (4.62) should then approximate at least to some extent the action of the operator in (4.68), that is, it should represent a

wave equation with propagation speed[6] $\sqrt{\|\boldsymbol{B}_0\|^2/\mu_0}$. In [25], this fact is exploited to justify substituting the term $\mathcal{Y}\mathcal{M}_{\boldsymbol{u}}^{-1}\mathcal{Z}$ in (4.62) with a discrete Laplacian operator, opportunely scaled by the wave speed. In the space-time case, this translates to the following approximation of the magnetic Schur complement:

$$S_A \approx M_A F_A^{-1} \underbrace{\left(F_A M_A^{-1} F_A + K_A\right)}_{=:C_A}, \tag{4.69}$$

where we define

$$K_A \coloneqq \operatorname*{diag}_{k=1,\dots,N_t}\left(\frac{\|\bar{\boldsymbol{B}}_0^k\|^2}{\mu_0}\mathcal{K}_A\right), \qquad \text{with} \qquad \bar{\boldsymbol{B}}^k \coloneqq \frac{1}{|\Omega|}\int_\Omega \nabla \times (A(\boldsymbol{x}, t_k)\hat{\boldsymbol{k}})\, d\boldsymbol{x}, \tag{4.70}$$

that is, $\bar{\boldsymbol{B}}_0^k$ is taken as the space average of the magnetic field at instant t_k. The task of explicitly assembling C_A in (4.69) is complicated by the presence of $\mathcal{M}_A^{-1}$. As a last simplification, we consider instead its approximation $\tilde{C}_A$, where we pick only the diagonal of $\mathcal{M}_A$, denoted as $\mathcal{D}_{\mathcal{M}_A}$, as a surrogate for the whole mass matrix. Overall, the space-time operator $\tilde{C}_A$ has the following block tri-diagonal structure:

$$\tilde{C}_A \coloneqq \operatorname*{diag}_{k=1,\dots,N_t}\left(\tilde{\mathcal{C}}_{A,k}\right) + \frac{1}{\Delta t}\operatorname*{diag}_{k=1,\dots,N_t-1}\left(\tilde{\mathcal{C}}_{A,-1,k}, -1\right) + \frac{1}{\Delta t^2}\operatorname*{diag}_{N_t-2}\left(\tilde{\mathcal{C}}_{A,-2}, -2\right), \tag{4.71}$$

where the notation $\operatorname{diag}_{k=1,\dots,N-i}((*)_k, -i)$ indicates a $N \times N$ block diagonal matrix whose i-th block sub-diagonal is filled by taking in order the indexed operators $\{(*)_k\}_{k=1}^{N-i}$. For each block diagonal, these are given by

$$\tilde{\mathcal{C}}_{A,k} \coloneqq \mathcal{F}_{A,k}\mathcal{D}_{\mathcal{M}_A}^{-1}\mathcal{F}_{A,k} + \frac{\|\bar{\boldsymbol{B}}_0^k\|^2}{\mu_0}\mathcal{K}_A \qquad k = 1, \dots, N_t \tag{4.72a}$$

$$\tilde{\mathcal{C}}_{A,-1,k} \coloneqq \mathcal{M}_A\mathcal{D}_{\mathcal{M}_A}^{-1}\mathcal{F}_{A,k} + \mathcal{F}_{A,k+1}\mathcal{D}_{\mathcal{M}_A}^{-1}\mathcal{M}_A \qquad k = 1, \dots, N_t - 1 \tag{4.72b}$$

$$\tilde{\mathcal{C}}_{A,-2} \coloneqq \mathcal{M}_A\mathcal{D}_{\mathcal{M}_A}^{-1}\mathcal{M}_A. \tag{4.72c}$$

This allows us to define the final form for our approximate space-time magnetic Schur complement:

$$S_A \approx \tilde{S}_A \coloneqq M_A F_A^{-1}\tilde{C}_A \qquad \Longleftrightarrow \qquad \tilde{S}_A^{-1} \coloneqq \tilde{C}_A^{-1}F_A M_A^{-1}. \tag{4.73}$$

[6] This corresponds to the speed at which *Alfvén waves* travel: this is a type of magnetohydrodynamic wave generated in response to a perturbation to the magnetic field induced by the presence of currents in the plasma [2].

We thus proceed to simplify the solution of systems involving (4.52), in complete analogy with Sec. 4.2.1, and substitute its block upper-triangular factor $U_{\boldsymbol{u}jA}$ with its approximation

$$\tilde{U}_{\boldsymbol{u}jA} \coloneqq \begin{bmatrix} F_{\boldsymbol{u}} & & Z_j & Z_A \\ & I & & \\ & & M_j & K_{jA} \\ & & & \tilde{S}_A \end{bmatrix}. \tag{4.74}$$

4.2.3 Definition of the space-time block preconditioner

Incorporating the approximate Schur complements introduced in Sec. 4.2.1 and 4.2.2 into the approximate factorisation (4.48), we are ready to define our space-time block preconditioner for (4.40). This appears in the form

$$P \coloneqq L_{\boldsymbol{u}jA}\tilde{U}_{\boldsymbol{u}jA}F_{\boldsymbol{u}I}^{-1}L_{\boldsymbol{u}p}\tilde{U}_{\boldsymbol{u}p} \quad \Longleftrightarrow \quad P^{-1} \coloneqq \tilde{U}_{\boldsymbol{u}p}^{-1}L_{\boldsymbol{u}p}^{-1}F_{\boldsymbol{u}I}\tilde{U}_{\boldsymbol{u}jA}^{-1}L_{\boldsymbol{u}jA}^{-1}, \tag{4.75}$$

so that applying its inverse involves the following sequence of operations:

Step 1 *Invert* $L_{\boldsymbol{u}jA}$. From its definition in (4.52), after some modifications we can show that applying its inverse to a generic block vector $[\boldsymbol{x}_{\boldsymbol{u}}^T, \boldsymbol{x}_p^T, \boldsymbol{x}_j^T, \boldsymbol{x}_A^T]^T$ is equivalent to computing

$$L_{\boldsymbol{u}jA}^{-1}\begin{bmatrix} \boldsymbol{x}_{\boldsymbol{u}} \\ \boldsymbol{x}_p \\ \boldsymbol{x}_j \\ \boldsymbol{x}_A \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_{\boldsymbol{u}} \\ \boldsymbol{x}_p \\ \boldsymbol{x}_j \\ \boldsymbol{x}_A + Y F_{\boldsymbol{u}}^{-1}(Z_j M_j^{-1}\boldsymbol{x}_j - \boldsymbol{x}_{\boldsymbol{u}}) \end{bmatrix}. \tag{4.76}$$

This operation requires solving N_t independent systems involving the current intensity mass matrix $\mathcal{M}_j$, (which can be trivially parallelised over the various time-steps), and more importantly inverting the velocity space-time matrix $F_{\boldsymbol{u}}$ (4.36). We faced a similar[7] problem in Chap. 3, so we refer to Sec. 3.4 for strategies on how to effectively tackle its parallel-in-time solution.

Step 2 *Apply* $F_{\boldsymbol{u}I}\tilde{U}_{\boldsymbol{u}jA}^{-1}$. The two operators can be combined, rendering the task in this step equivalent to inverting the system

$$\bar{U}_{\boldsymbol{u}jA} \coloneqq \tilde{U}_{\boldsymbol{u}jA}F_{\boldsymbol{u}I}^{-1} = \begin{bmatrix} I & & Z_j & Z_A \\ & I & & \\ & & M_j & K_{jA} \\ & & & \tilde{S}_A \end{bmatrix}. \tag{4.77}$$

[7] The sole difference being the definition of the diagonal block $\mathcal{F}_{\boldsymbol{u}}$, which here contains the extra term (4.49), as mentioned in Sec. 4.2.1.

This step is possibly the most challenging in the application of the preconditioner. In fact, apart from an additional inversion of M_j, which can be dealt with in the same way as in step 1, this requires the inversion of the space-time operator $\tilde{S}_A$, and particularly its factor $\tilde{C}_A$. From its definition in (4.71), we see that this is a fundamentally different operator from $F_{\boldsymbol{u}}$: while the latter essentially represents the space-time discretisation of a standard parabolic PDE, the former rather corresponds to a space-time operator containing high-order mixed spatial and temporal derivatives. It is hence not clear what would be the best strategy to time-parallelise its solution. To keep the scope of this chapter contained, then, we limit ourselves to considering direct solvers (*i.e.*, time-stepping) for inverting (4.73), thus momentarily suspending considerations on the parallel performance of the preconditioner.

Step 3 *Invert* $L_{\boldsymbol{up}}$. Once again, the main task of this step lies in the inversion of the space-time velocity matrix $F_{\boldsymbol{u}}$, and is similar in complexity to step 1.

Step 4 *Invert* $\tilde{U}_{\boldsymbol{up}}$. Comparing the definition of (4.51) with that of (3.23), we can convince ourselves that this task is equivalent to applying the block upper-triangular space-time preconditioner P_T proposed in Sec. 3.2, which involves once again inverting the space-time velocity matrix $F_{\boldsymbol{u}}$, as well as the approximate space-time pressure Schur complement $\tilde{S}_p$ (the latter task is trivially parallelisable, as discussed in detail in Sec. 3.2.1.1).

The heaviest cost in the application of the preconditioner is associated with the solution of the four space-time systems illustrated above: three involving $F_{\boldsymbol{u}}$, and one $\tilde{C}_A$. We reduce this number by dropping some of the block lower-triangular terms in (4.75): this has the result of rendering the preconditioner closer to a block upper-triangular form. In particular, by dropping its $L_{\boldsymbol{u}jA}$ factor, we obtain

$$\tilde{P} \coloneqq \bar{U}_{\boldsymbol{u}jA} L_{\boldsymbol{up}} \tilde{U}_{\boldsymbol{up}} = \begin{bmatrix} F_{\boldsymbol{u}} & B^T & Z_j & Z_A \\ B & \Delta S_p & & \\ & & M_j & K_{jA} \\ & & & \tilde{S}_A \end{bmatrix}, \tag{4.78}$$

where $\Delta S_p \coloneqq \tilde{S}_p - S_p$. Notice that applying this simplification amounts to skipping step 1, so that one fewer space-time solve for $F_{\boldsymbol{u}}$ and one fewer diagonal solve for M_j are necessary for applying the preconditioner. A similar simplification in the single time-step framework is investigated in [25] as well. In addition, we perform one final approximation, and drop also the remaining block lower-triangular factor $L_{\boldsymbol{up}}$ (*i.e.*, skip step 3), further reducing by one the total number of space-time solve

necessary for $F_{\boldsymbol{u}}$. With this, we end up with the block upper-triangular space-time preconditioner

$$P_T \coloneqq \bar{U}_{\boldsymbol{u}jA}\tilde{U}_{\boldsymbol{u}p} = \begin{bmatrix} F_{\boldsymbol{u}} & B^T & Z_j & Z_A \\ & \tilde{S}_p & & \\ & & M_j & K_{jA} \\ & & & \tilde{S}_A \end{bmatrix}. \tag{4.79}$$

Considerations on the spectral properties of the preconditioner, together with some partial indications on how these are affected by dropping its block lower triangular terms, are provided in Sec. 4.3. Before addressing this issue, in the following section we briefly discuss on the cost associated with the application of the space-time block preconditioner introduced here, and on how it compares with its single time-step counterpart.

4.2.4 Comparison with single time-step block preconditioner

Ultimately, in order to judge whether the parallel application of the space-time block preconditioners introduced in Sec. 4.2.3 has the potential of reducing time-to-solution with respect to the sequential application of their single time-step equivalent, we need to compare the total cost (in computational time) associated with either strategy. To do so, we follow similar steps as in Sec. 3.2.1.1, and first identify the main parameters which determine the relative efficiency of one approach over the other. An estimate of these parameter is then provided in Sec. 4.4.3. To simplify our exposition, we focus our analysis on the block upper-triangular preconditioner P_T (4.79), but similar considerations hold for (4.75) and (4.78) as well.

To this purpose, let us begin by explicitly defining the single time-step equivalent of P_T. At the k-th time-step, this is given by

$$\mathcal{P}_{T,k} \coloneqq \underbrace{\begin{bmatrix} I & & \mathcal{Z}_{j,k} & \mathcal{Z}_{A,k} \\ & I & & \\ & & \mathcal{M}_j & \mathcal{K}_{jA} \\ & & & \tilde{\mathcal{S}}_{A,k} \end{bmatrix}}_{=:\mathcal{U}_{\boldsymbol{u}jA,k}} \underbrace{\begin{bmatrix} \mathcal{F}_{\boldsymbol{u},k} & \mathcal{B}^T & & \\ & \tilde{\mathcal{S}}_{p,k} & & \\ & & I & \\ & & & I \end{bmatrix}}_{=:\mathcal{U}_{\boldsymbol{u}p,k}}, \tag{4.80}$$

where $\tilde{\mathcal{S}}_{p,k}$ is the single time-step pressure Schur complement approximation introduced in (3.39), while $\tilde{\mathcal{S}}_{A,k}$ is the single time-step magnetic field Schur complement approximation from [25, (3.32)], defined as

$$\tilde{\mathcal{S}}_{A,k} \coloneqq \mathcal{M}_A \mathcal{F}_{A,k}^{-1} \tilde{\mathcal{C}}_{A,k}, \tag{4.81}$$

with $\tilde{\mathcal{C}}_{A,k}$ as in (4.72a). The two factors in (4.80) mimic the velocity-magnetic field and velocity-pressure couplings discussed in Sec. 4.2.1 and 4.2.2. In Sec. 3.2.1.1 we have already covered how the cost of inverting the rightmost factor in (4.80) in the single time-step setting compares with that of the whole space-time one, and we have reached the conclusion that, if one processor per time-step is available, the competitive advantage of the parallel method ultimately depends on two ratios: $C_{\mathcal{F}_{\boldsymbol{u}}} \cdot N_t / C_{F_{\boldsymbol{u}}}$, which provides an indication of the parallel efficiency at which we can solve the whole velocity space-time system $F_{\boldsymbol{u}}$; and N_{it}^{ST}/N_{it}^{0}, which measures the overhead in terms of number iterations to convergence associated with preconditioning in space-time. We proceed by breaking down the cost comparison for the leftmost factor in (4.80) following a similar analysis. The total cost associated with the application of (4.80) is then given by summing these two contributions.

Inverting $\tilde{\mathcal{U}}_{\boldsymbol{u}jA,k}$ requires solving for each of its diagonal blocks, as well as applying operators $\mathcal{K}_{jA}$, $\mathcal{Z}_{A,k}$ and $\mathcal{Z}_{j,k}$. When considering its space-time counterpart $\bar{U}_{\boldsymbol{u}jA}$ (4.77), a similar set of operations must be performed, except we need to deal with space-time operators. We have that M_j, K_{jA}, Z_A and Z_j are all block-diagonal, so their application (or that of their inverse, in the case of M_j) is trivially parallelisable over the time-steps: if N_t processors are available, we then expect this procedure to require a computational time comparable to that for the single time-step equivalent. The main difference rather lies in the block lower-triangular space-time operator $\tilde{S}_A$ (4.73). By looking at the single factors composing it, we have that the application of its inverse requires: (i) inverting the space-time mass matrix M_A (which is block diagonal, and hence again trivially parallelisable); (ii) applying the space-time vector potential operator F_A (which is block *bi*-diagonal, and hence requires some overhead communication between the processors, but of marginal impact in terms of total cost, following considerations similar to those discussed for F_p (3.34) in Sec. 3.2.1.1); finally, (iii) inverting the space-time operator $\tilde{C}_A$ (4.71), which represents the biggest obstacle in determining the parallel efficiency of the application of the space-time magnetic Schur complement approximation. We denote with $C_{\tilde{C}_A}$ the cost associated with operation (iii), and with $C_{\tilde{\mathcal{C}}_A}$ the average cost per time-step associated with inverting its single time-step counterpart, $\tilde{\mathcal{C}}_{A,k}$ in (4.81). Combining this with the considerations above regarding $\tilde{\mathcal{U}}_{\boldsymbol{u}p,k}$, we have that the computational times necessary to apply P_T once, and $\mathcal{P}_{T,k}$ once *per time-step* are, respectively, $C_{F_{\boldsymbol{u}}} + C_{\tilde{C}_A}$ and $(C_{\mathcal{F}_{\boldsymbol{u}}} + C_{\tilde{\mathcal{C}}_A})N_t$. Therefore, one way to measure whether the space-time approach is competitive over

the time-stepping one consists in determining if

$$\frac{N_{it,L}^{ST}}{N_{it,L}^{0}}\left(C_{F_{\boldsymbol{u}}}+C_{\tilde{C}_A}\right)<\left(C_{\mathcal{F}_{\boldsymbol{u}}}+C_{\tilde{\mathcal{C}}_A}\right)N_t, \tag{4.82}$$

$N_{it,L}^{ST}$ and $N_{it,L}^{0}$ being the number of iterations to convergence required by the solver for the linearised system in the two cases, respectively (compare this to (3.40)).

However, an additional complication with respect to Sec. 3.2.1.1, is given by the nonlinearities present in the IMHD target system of this chapter (4.23). As a consequence, another relevant metric in determining the effectiveness of the space-time approach over the time-stepping one is given by the relative cost and number of iterations associated with the Newton solver. For each outer iteration, the main task consists in re-assembling the operators composing the discrete Jacobian (4.40). Most of these are either constant (and hence can be assembled once and for all during the start-up phase of the solver), or depend on values on a single time-step (and hence their assembly can be trivially parallelised, at no overhead cost with respect to time-stepping). One exception to this, however, is given by the space-time operator $\tilde{C}_A$. In fact, the k-th block in its first block subdiagonal (defined in (4.72b)) depends on values both at instant k and $k+1$. Therefore, if the solver chosen for this space-time operator requires explicit assembly of this block, its construction might require significant additional computational time with respect to inverting its single time-step equivalent (even if it does not, additional communication between processors might be required, likely increasing the cost for inverting $\tilde{C}_A$). To keep the analysis in this chapter contained, we do not explore this issue in detail, and limit ourselves to track also the relative number of Newton iterations for the space-time and the time-stepping approaches, denoted as $N_{it,NL}^{ST}$ and $N_{it,NL}^{0}$ respectively.

Finally, also in terms of memory usage the situation is analogous to the one described in Sec. 3.2.1.1: even in this case, in fact, in our implementation we associate each processor with a different time-step, and have it assemble its corresponding spatial operators. As in the previous chapter, this results in the total memory required scaling linearly with the number of processors employed, which is in line with other PinT approaches.

4.3 Eigenvalues clustering

In this section we analyse the eigenvalue distribution for the discrete operator (4.40), right-preconditioned with the approximate block LU factorisation introduced in (4.45).

We do not consider here the effect of the approximations to the space-time Schur complements discussed in Sec. 4.2.1 and 4.2.2, and rather stick to the ideal case where the exact S_p and S_A operators ((4.46) and (4.53)) are used, which gives us a best-case scenario of the performance of the preconditioner. The goal is once again to provide some evidence for the effectiveness of this choice of preconditioner, although similar caveats as those made for Sec. 3.3 hold here as well: eigenvalues do not tell the whole picture when it comes to convergence of GMRES [68], especially when dealing with nonsymmetric matrices the likes of which compose our target system (4.44). Similarly to Sec. 3.3, we show that the eigenvalues distribution of the space-time preconditioned system is tightly connected to that of its single time-step counterpart.

Using the definitions of (4.40) and (4.45), we see that the preconditioned system is given by

$$J_{\mathcal{N}}P^{-1} = \left[\begin{array}{cc|cc} F_{\boldsymbol{u}} & B^T & Z_j & Z_A \\ B & & & \\ \hline & & M_j & K_{jA} \\ Y & & & F_A \end{array}\right] \left[\begin{array}{cc|cc} F_{\boldsymbol{u}} & B^T & Z_j & Z_A \\ B & & & \\ \hline & & M_j & K_{jA} \\ Y & YF_{\boldsymbol{u}}^{-1}B^T & & F_A \end{array}\right]^{-1}. \tag{4.83}$$

Notice we highlighted the 2×2 block structure of the two factors: they can in fact be more compactly expressed as

$$\begin{bmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} \end{bmatrix} \begin{bmatrix} \mathrm{A} & \mathrm{B} \\ \tilde{\mathrm{C}} & \mathrm{D} \end{bmatrix}^{-1}, \tag{4.84}$$

whose common composing blocks are given by

$$\mathrm{A} = \begin{bmatrix} F_{\boldsymbol{u}} & B^T \\ B & \end{bmatrix}, \quad \mathrm{B} = \begin{bmatrix} Z_j & Z_A \end{bmatrix}, \quad \text{and} \quad \mathrm{D} = \begin{bmatrix} M_j & K_{jA} \\ & F_A \end{bmatrix}, \tag{4.85}$$

while the bottom-left corner differs, being

$$\mathrm{C} = \begin{bmatrix} Y & \end{bmatrix}, \quad \text{and} \quad \tilde{\mathrm{C}} = \begin{bmatrix} Y & YF_{\boldsymbol{u}}^{-1}B^T \end{bmatrix}, \tag{4.86}$$

for the matrix on the left and on the right, respectively. After some manipulations, we see that explicitly multiplying the factors in (4.83) results in the following block lower-triangular matrix:

$$\begin{bmatrix} I_{N_t(N_{\boldsymbol{u}}+N_p)} & \\ \mathrm{CA}^{-1} - \mathrm{S_D}\mathrm{S}_{\tilde{\mathrm{D}}}^{-1}\tilde{\mathrm{C}}\mathrm{A}^{-1} & \mathrm{S_D}\mathrm{S}_{\tilde{\mathrm{D}}}^{-1} \end{bmatrix}. \tag{4.87}$$

This already identifies a number $N_t(N_{\boldsymbol{u}} + N_p)$ of unitary eigenvalues: due to the triangular nature of the matrix, the remaining eigenvalues are given by those of the

bottom-right block. The two factors composing it are provided, respectively, by

$$\begin{aligned}\mathrm{S_D} = \mathrm{D} - \mathrm{CA^{-1}B} &= \begin{bmatrix} M_j & K_{jA} \\ & F_A \end{bmatrix} - \begin{bmatrix} \\ Y \end{bmatrix} \begin{bmatrix} F_{\boldsymbol{u}} & B^T \\ B & \end{bmatrix}^{-1} \begin{bmatrix} Z_j & Z_A \\ & \end{bmatrix} \\ &= \begin{bmatrix} M_j & K_{jA} \\ \underbrace{-YF_{\boldsymbol{u}}^{-1}\left(I + B^T S_p^{-1} B F_{\boldsymbol{u}}^{-1}\right) Z_j}_{=:\mathrm{E}} & \underbrace{F_A - YF_{\boldsymbol{u}}^{-1}\left(I + B^T S_p^{-1} B F_{\boldsymbol{u}}^{-1}\right) Z_A}_{=:\mathrm{F}} \end{bmatrix},\end{aligned} \tag{4.88}$$

where S_p is the familiar space-time pressure Schur complement (4.46), and

$$\begin{aligned}\mathrm{S_{\tilde{D}}} = \mathrm{D} - \tilde{\mathrm{C}}\mathrm{A^{-1}B} &= \begin{bmatrix} M_j & K_{jA} \\ & F_A \end{bmatrix} - \begin{bmatrix} Y & YF_{\boldsymbol{u}}^{-1}B^T \end{bmatrix} \begin{bmatrix} F_{\boldsymbol{u}} & B^T \\ B & \end{bmatrix}^{-1} \begin{bmatrix} Z_j & Z_A \\ & \end{bmatrix} \\ &= \begin{bmatrix} M_j & K_{jA} \\ \underbrace{\mathrm{E} + YF_{\boldsymbol{u}}^{-1}B^T S_p^{-1} B F_{\boldsymbol{u}}^{-1} Z_j}_{\tilde{\mathrm{E}}} & \underbrace{\mathrm{F} + YF_{\boldsymbol{u}}^{-1}B^T S_p^{-1} B F_{\boldsymbol{u}}^{-1} Z_A}_{\tilde{\mathrm{F}}} \end{bmatrix}.\end{aligned} \tag{4.89}$$

These too have a 2×2 block structure, and their product is in the form

$$\begin{bmatrix} M_j & K_{jA} \\ \mathrm{E} & \mathrm{F} \end{bmatrix} \begin{bmatrix} M_j & K_{jA} \\ \tilde{\mathrm{E}} & \tilde{\mathrm{F}} \end{bmatrix}^{-1}. \tag{4.90}$$

After some more calculations, their multiplication produces yet another block lower-triangular matrix,

$$\begin{bmatrix} I_{N_t N_j} & \\ \mathrm{E}M_j^{-1} - \mathrm{S_F S_{\tilde{F}}^{-1}} \tilde{\mathrm{E}} M_j^{-1} & \mathrm{S_F S_{\tilde{F}}^{-1}} \end{bmatrix}, \tag{4.91}$$

which allows us to rule out some additional $N_t N_j$ unitary eigenvalues. Just like for (4.87), even in this case the remaining eigenvalues coincide with those of the bottom-right block of the matrix. Another round of algebraic manipulations gives us an explicit formula for this operator:

$$\begin{aligned}\mathrm{S_F S_{\tilde{F}}^{-1}} &= (\mathrm{F} - \mathrm{E}M_j^{-1}K_{jA})(\tilde{\mathrm{F}} - \tilde{\mathrm{E}}M_j^{-1}K_{jA})^{-1} \\ &= I_{N_t N_A} - YF_{\boldsymbol{u}}^{-1}B^T S_p^{-1} B F_{\boldsymbol{u}}^{-1}\left(Z_A - Z_j M_j^{-1} K_{jA}\right) S_A^{-1},\end{aligned} \tag{4.92}$$

where S_A is the space-time magnetic Schur complement introduced in (4.53). Each of the space-time operators composing (4.92) has either a block diagonal or a block lower-triangular structure, implying that in order to recover its eigenvalues it suffices to identify those of the operators appearing in its diagonal blocks. These in turn are given by

$$I_{N_A} - \mathcal{Y}_k \mathcal{F}_{\boldsymbol{u},k}^{-1} \mathcal{B}^T \mathcal{S}_{p,k}^{-1} \mathcal{B} \mathcal{F}_{\boldsymbol{u},k}^{-1}\left(\mathcal{Z}_{A,k} - \mathcal{Z}_{j,k}, \mathcal{M}_j^{-1}\mathcal{K}_{jA}\right) \mathcal{S}_{A,k}^{-1}, \qquad \forall k = 1, \dots, N_t, \tag{4.93}$$

where $\mathcal{Z}_{j,k}$ and $\mathcal{Z}_{A,k}$ are the Lorentz force linearisation terms, following (4.30). By considering how we compact these two terms into (4.56), and adjusting for the notation used in their article, we can convince ourselves that operator (4.93) corresponds to the one described in [25, (3.10)], which ultimately dictates the clustering properties of their preconditioner. As the spectrum of the single time-step preconditioned system is analysed in detail in [25, Fig. 3.1], we do not repeat their analysis here, and rather limit ourselves to pointing out how even in the application to IMHD considered in this chapter, the eigenvalues distribution of the whole space-time preconditioned system ends up depending directly on that of its single time-step counterpart, similarly to what described in Sec. 3.3.

This remains valid if we decide to simplify (4.75) by dropping its first block lower-triangular term $L_{\boldsymbol{u}jA}$, as discussed at the end of Sec. 4.2.3. In fact, if we consider $\tilde{P}$ from (4.78) as a simplification to P, the preconditioned system is modified to

$$\begin{aligned} J_{\mathcal{N}}\tilde{P}^{-1} &= J_{\mathcal{N}}\tilde{U}_{\boldsymbol{up}}^{-1}L_{\boldsymbol{up}}^{-1}F_{\boldsymbol{u}I}\tilde{U}_{\boldsymbol{u}jA}^{-1} = J_{\mathcal{N}}(\tilde{U}_{\boldsymbol{up}}^{-1}L_{\boldsymbol{up}}^{-1}F_{\boldsymbol{u}I}\tilde{U}_{\boldsymbol{u}jA}^{-1}L_{\boldsymbol{u}jA}^{-1})L_{\boldsymbol{u}jA} \\ &= J_{\mathcal{N}}P^{-1}\begin{bmatrix} I & & & \\ & I & & \\ & & I & \\ YF_{\boldsymbol{u}}^{-1} & & -YF_{\boldsymbol{u}}^{-1}Z_jM_j^{-1} & I \end{bmatrix}. \end{aligned} \tag{4.94}$$

We have already established how, in the ideal case where we pick the exact pressure and magnetic Schur complements, $J_{\mathcal{N}}P^{-1}$ gives rise to the block lower-triangular operator (4.91). The matrix we are now further multiplying by, $L_{\boldsymbol{u}jA}$, is also block lower-triangular, and presents identities along its whole diagonal: from this, we safely conclude that excluding this factor has no effect on the eigenvalue distribution of the original preconditioned system.

Unfortunately, dropping the second block lower-triangular term $L_{\boldsymbol{u},p}$ as well has a less clear effect on the spectral properties of the resulting preconditioned system, and the corresponding analysis becomes more challenging. In the scope of this thesis, we do not investigate this further, and rather showcase the effectiveness of this additional simplification with some numerical experiments, which are presented in the following section.

4.4 Results

In this section, we experiment with the effectiveness of the preconditioner introduced in Sec. 4.2 by employing it for the acceleration of the solution of some model problems describing the arising and evolution of instabilities in resistive plasma. In the

framework of resistive plasmas, the instabilities described here result in modifications to the topology of the magnetic field lines, giving rise to separation and reconnection phenomena, usually associated with the generation of current sheets [66, Chap. 9.6]. These phenomena have been a fascinating subject of study over the last decades, and research on this area has undergone considerable effort: providing a complete overview of the topic is beyond the scope of this thesis, and we rather restrict ourselves to giving a general description of the dynamics triggered within the problems considered in Sec. 4.4.1. Nonetheless, we refer the interested reader to [66, Chap. 9 and Chap. 20], as well as to the excellent review on the topic by Dieter Biskamp, particularly [9, Chap. 3-4].

We point out that most of the results in this section are still preliminary: additional effort needs to be put into the design of the space-time preconditioner before it can reach the maturity necessary for the application to real-world scenarios. Particularly, we do not discuss its performance for small values of μ and η, which require *ad-hoc* stabilisation, and will be a matter for future research. Nonetheless, the experiments conducted here serve as proof of concept to validate the effectiveness of the preconditioner developed in Sec. 4.2, and more generally to sustain the claim for the possibility to extend the framework of space-time block preconditioning to more challenging systems of PDEs.

4.4.1 Model problems

The problems introduced in this section have been often used in the literature in order to test algorithms for MHD: see for example [15,114] for problem 1, and [25,114,134] for problem 2, whose examples we follow in our set-up. We proceed to describe these in detail next.

Problem 1 *Tearing mode.* This is one of the classic instabilities in resistive plasmas. It arises from the interaction of magnetic fields with opposite orientations: in the presence of resistivity, the opposite fields are allowed to "diffuse" into each other, thus progressively weakening until annihilation occurs. At this stage, the magnetic field lines running in one direction reconnect with those in the opposite direction, and produce a so-called *x-point*, where the lines intersect (as opposed to *o-points*, around which the lines rotate). This evolution is dictated by nonlinear interactions, and it occurs on a relatively large time-scale; nonetheless, the solution eventually stabilises, at which point it is said to have reached *saturation*. An example solution

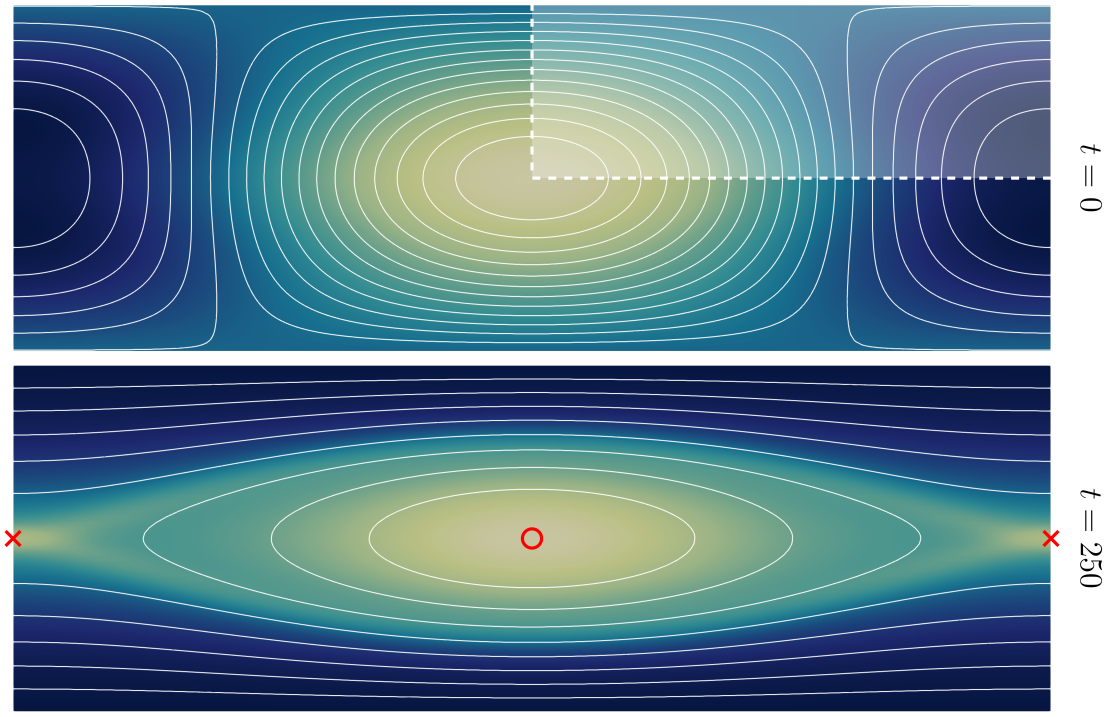


Figure 4.1: Evolution of contour plots of current intensity with superimposed isolines of the magnetic vector potential A (*i.e.*, streamlines of the magnetic field), for an example solution of problem problem 1, with $\mu = \eta = 10^{-3}$ and $\mu_0 = 1$. The top plot shows the initial conditions, with the current flowing alternately outside the domain (light area) and inside the domain (dark areas) as we move along the x-axis, and the resulting magnetic field lines rotating counter- and clock-wise, respectively; the shaded area highlights the actual domain of the simulation. As time proceeds, the magnetic field lines at the sides of the domain get progressively squashed and eroded, and eventually annihilated. Notice that this causes the lines to intersect at the middle of the left and right sides of the domain (x-points, highlighted in red together with the o-points). The solution at saturation is presented in the bottom plot.

of this problem, illustrating the dynamics described above, is plotted in Fig. 4.1; the corresponding velocity and pressure fields are instead reported in Fig. 4.2.

The system configuration for this first test-case is as follows. The flow is initially at quiet state,

$$\bar{\boldsymbol{u}}^0_{TM}(\boldsymbol{x}) = 0, \tag{4.95}$$

and the vector potential is at the *Harris sheet equilibrium* [9, Chap. 4.1], that is

$$A^{eq}_{TM}(\boldsymbol{x}) = \frac{1}{\lambda} \ln\left(\cosh(\lambda y)\right), \tag{4.96}$$

for a given λ. This must be sustained by the external electric field

$$E^{eq}_{TM}(\boldsymbol{x}) = \frac{\eta}{\mu_0} \nabla^2 A^{eq}_{TM}(\boldsymbol{x}) = \frac{\lambda\eta}{\mu_0} \cosh(\lambda y)^{-2}, \tag{4.97}$$

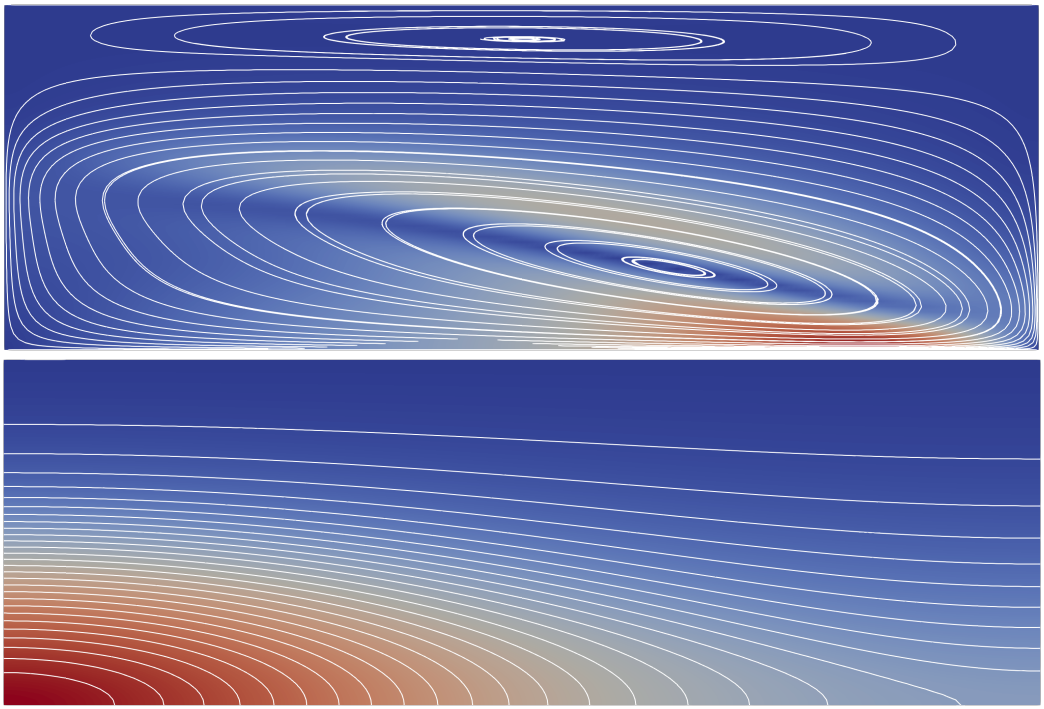


Figure 4.2: Velocity magnitude and streamlines (top), and pressure contour plot (bottom) for an example solution of the flow in the tearing mode problem 1 at saturation $t = 250$, with $\mu = \eta = 10^{-3}$ and $\mu_0 = 1$. The flow is accelerated in a vortex by Lorentz force: the vorticity of the field has largest intensity at the *separatrix* (the border of the central "eye" in Fig. 4.1, where the current gradient is largest). Only the simulation domain (top-right corner in Fig. 4.1) is shown.

which is imposed as a right-hand side for the vector potential equation in (4.19). The action of $A^{eq}_{TM}(\boldsymbol{x})$ on the flow (via Lorentz force) is counterbalanced by the pressure term

$$p^{eq}_{TM}(\boldsymbol{x}) = \frac{1}{2\mu_0}\cosh(\lambda y)^{-2} - \bar{p}^{eq}_{TM}, \tag{4.98}$$

where the constant $\bar{p}^{eq}_{TM}$ is selected to ensure that the pressure has zero mean at equilibrium. In order to excite the tearing mode, we perturb the equilibrium by considering the following initial conditions for the vector potential:

$$\bar{A}^0_{TM}(\boldsymbol{x}) = A^{eq}_{TM}(\boldsymbol{x}) - \varepsilon\cos(\pi y)\cos\left(\frac{2\pi x}{L}\right), \tag{4.99}$$

with L being the x-length of the domain. The resulting solution is periodic in $x \in [-L, L]$, and shows alternating x- and o-points at saturation; the "eye" around the o-point is referred to as a *magnetic island*. The solution also presents axial symmetries, so instead of considering the whole spatial domain $[-L, L] \times [-1/2, 1/2]$, we can simulate only its top-right corner $\Omega = [0, L] \times [0, 1/2]$, and flip its solution around

both the x and y axes. As such, we prescribe symmetric boundary conditions on sides $x = 0$, $x = L$, and $y = 0$; on the top side $y = 1/2$, instead, we prescribe Dirichlet BC for A (fixed at the equilibrium value (4.96)), and slip conditions for $\boldsymbol{u}$. Notice the resulting flow is fully enclosed, so that the considerations detailed in Sec. 3.2.1.2 on the assembly of the approximate space-time pressure Schur complement $\tilde{S}_p$ hold in this case as well. The parameters used in our problem are $\lambda = 5$, $\varepsilon = 10^{-3}$, and $L = 3$.

Problem 2 *Island coalescence.* This problem describes the evolution of two magnetic islands in a plasma, as they could stem from the tearing mode perturbation introduced in problem 1: notice the final condition in Fig. 4.1 roughly matches the initial condition in half of the domain in Fig. 4.3. The two o-points are pulled together by Lorentz force, until eventually the x-point in between them separates and the magnetic islands merge into one (that is, they *coalesce*), generating an electric discharge; the specific dynamics depends on the value of the magnetic resistivity, but generally reconnection occurs in a time-scale much smaller than that for problem 1; an example of the evolution of this system is provided in Fig. 4.3 and 4.4.

As in problem 1, we start from a quiet state,

$$\bar{\boldsymbol{u}}^0_{IC}(\boldsymbol{x}) = 0. \tag{4.100}$$

with the vector potential given by a modification to the Harris sheet equilibrium in (4.96). This is at times referred to as *Fadeev* or *Schmid-Burgk equilibrium*, since they pioneered its study in [45, 131], or also as *corrugated sheet pinch equilibrium* [9, Chap. 4.3], and is given by

$$A^{eq}_{IC}(\boldsymbol{x}) = \frac{1}{2\pi} \ln\left[\cosh(2\pi y) + \beta\cos(2\pi x)\right], \tag{4.101}$$

for a given parameter β. This too must be sustained by an external electric field, in this case given by

$$E^{eq}_{IC}(\boldsymbol{x}) = \frac{\eta}{\mu_0}\nabla^2 A^{eq}_{IC}(\boldsymbol{x}) = \frac{\eta}{\mu_0}\frac{2\pi(1-\beta^2)}{\left(\cosh(2\pi y) + \beta\cos(2\pi x)\right)^2}. \tag{4.102}$$

The flow equilibrium is maintained by a counterbalancing pressure,

$$p^{eq}_{IC}(\boldsymbol{x}) = \frac{1}{2\mu_0}\frac{1-\beta^2}{\left(\cosh(2\pi y) + \beta\cos(2\pi x)\right)^2} - \bar{p}^{eq}_{IC}, \tag{4.103}$$

where again the quantity $\bar{p}^{eq}_{IC}$ is chosen to ensure the equilibrium pressure is zero-mean valued. This equilibrium is perturbed by considering as initial conditions for

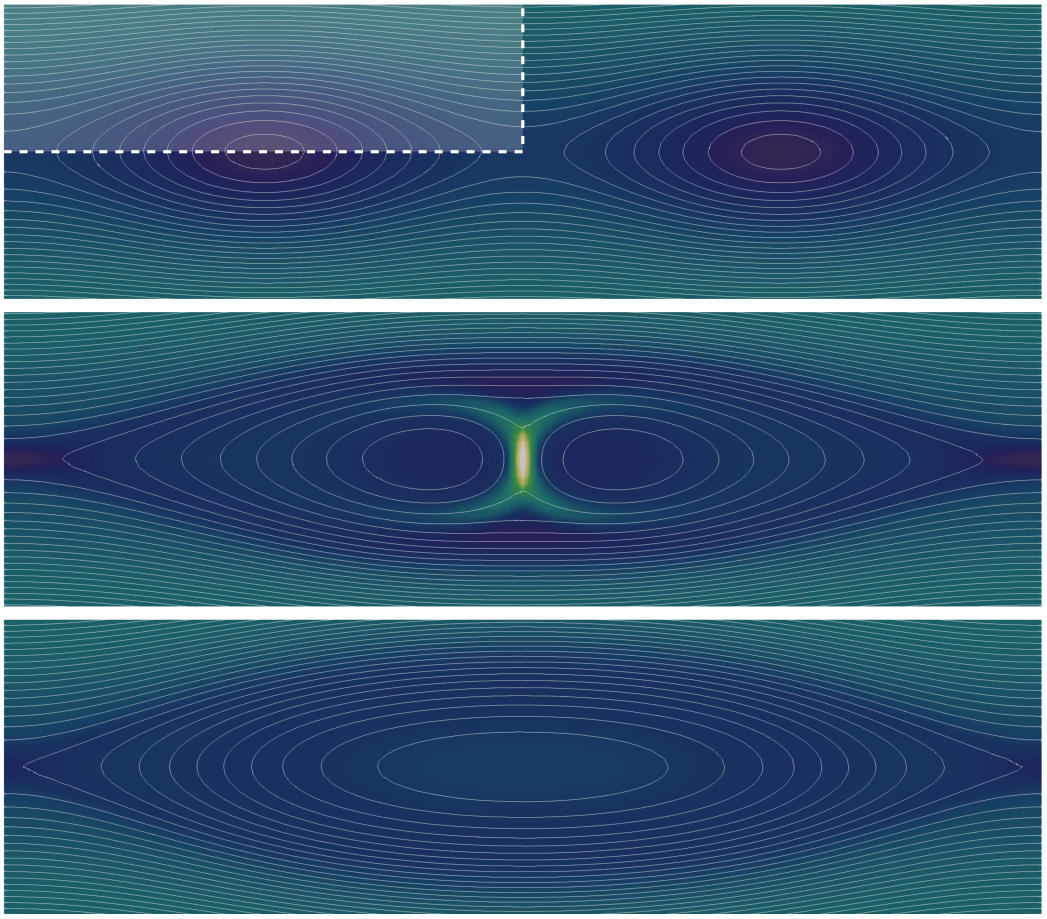


Figure 4.3: Evolution of contour plots of current intensity with superimposed isolines of the magnetic vector potential A (*i.e.*, streamlines of the magnetic field), for an example solution of problem 2, with $\mu = \eta = 10^{-3}$ and $\mu_0 = 1$. The top plot shows the initial conditions, with the magnetic islands induced by the perturbation; the shaded area highlights the actual domain of the simulation (notice the y-axis has been cropped to better zoom on the islands). As time proceeds, the islands start attracting each other until reconnection occurs in the middle plot, where we see the associated electric discharge. In the last plot the reconnection has completed, and the two islands have coalesced into one.

the vector potential

$$\bar{A}^0_{IC}(\boldsymbol{x}) = A^{eq}_{IC}(\boldsymbol{x}) + \varepsilon \cos\left(\frac{\pi}{2} y\right) \cos(\pi x). \tag{4.104}$$

Similarly to problem 1, in this case too the resulting solution is periodic in $x \in [0, 2]$ and presents a number of symmetries, which we exploit to simplify our simulation and reduce the domain from $[0, 2] \times [-1, 1]$ to its top-left corner $\Omega = \Omega_\square = [0, 1]^2$. We impose symmetry on the $x = 0$, $x = 1$ and $y = 0$ axes, while on $y = 1$, we prescribe Dirichlet BC for A (again fixed to the equilibrium (4.101)), and slip conditions for $\boldsymbol{u}$,

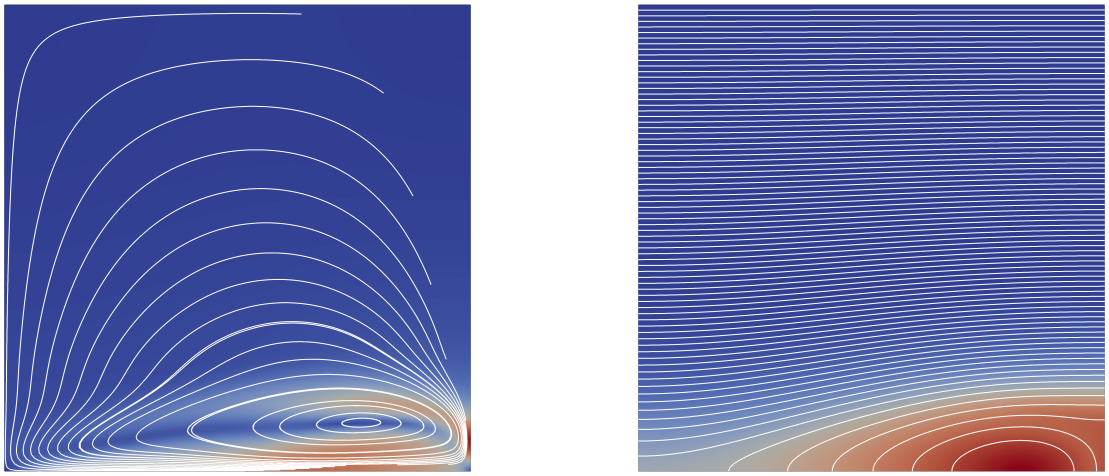


Figure 4.4: Velocity magnitude and streamlines (left), and pressure contour plot (right) for an example solution of the flow in the island coalescence problem 2 at reconnection $t = 7$, with $\mu = \eta = 10^{-3}$ and $\mu_0 = 1$. Only the simulation domain (top-left corner in Fig. 4.3) is shown.

rendering the flow fully enclosed also in this case. The parameters considered for this problem are $\beta = 0.2$ and $\varepsilon = 10^{-3}$.

In our experiments, the choice of finite-element approximation for the discretisation of the functional spaces (3.10) and (4.27) falls on the Taylor-Hood $P_3 - P_2$ pair for velocity and pressure [118, Chap. 17.4], while we consider P_1 for both the vector potential and the current intensity variables. This stems from considerations on *Finite Element Exterior Calculus* [4, 79], and from the necessity of ensuring that the two forcing terms in the momentum equation, ∇p and $j\nabla A$, can balance each other out at equilibrium; this choice is otherwise rather empirical, and an additional theoretical justification is needed to show whether these FE spaces indeed satisfy the required stability conditions.

Regarding the parameters for the problems, we put ourselves in a somewhat favourable position, by considering only $\mu = \eta = \mu_0 = 1$. While this choice has the effect of rendering some of the dynamics described in problems 1 and 2 less evident, it allows us to neglect matters regarding stability under advection-dominated regimes, which is beyond the purpose of this analysis, and rather focus on providing a general indication of what kind of performance one might expect from the space-time preconditioner.

The actual code used for the numerical simulations is an expansion of the one developed for Sec. 3.4: as such, it still relies on MFEM and PETSc [7, 102] for the

linear solvers necessary to tackle the linearised system (4.44) at each Newton iteration. As an initial guess for the discrete space-time velocity, pressure, and vector potential, we pick the projection of the equilibrium conditions, (4.95), (4.98), (4.96) and (4.100), (4.103), (4.101), onto their respective discrete finite element spaces (3.10) and (4.27); the additional variable j is instead initialised as the solution to its associated discrete system (4.42c):

$$\mathcal{M}_j \boldsymbol{j}^k = -\mathcal{K}_{jA} \boldsymbol{A}^k + \boldsymbol{h}^k, \qquad \forall k = 1, \dots, N_t, \tag{4.105}$$

where $\boldsymbol{A}^k$ is the initial guess on the vector potential at instant k, so that the initial residual associated with this variable is effectively 0.

In the following, we report some preliminary results from the application of the space-time preconditioner.

4.4.2 Performance of preconditioner

This section is dedicated to measuring the effectiveness of the application of the space-time preconditioner introduced in Sec. 4.2.3 for accelerating the iterative solution of the linearised IMHD system (4.44). In analogy with the previous chapter, in order to evaluate the preconditioner performance, we focus on tracking the number of iterations necessary to achieve convergence, for both the outer Newton solver and the inner GMRES solver. This preference is motivated by similar considerations as those made for Sec. 3.4: for the purpose of this study, this parameter represents a more indicative measure of the behaviour of the preconditioner than, *e.g.*, time-to-solution, as the latter strongly depends on implementation choices. This becomes even more significant given the possible challenges represented by step 2, and particularly by the inversion of $\tilde{C}_A$ in (4.71), whose effective time parallelisation we do not investigate further in this chapter.

For similar reasons, in this preliminary study we consider only the use of *exact* solvers for inverting the relevant block operators appearing in P_T. That is to say, we employ sequential time-stepping for the solution of the space-time operators $F_{\boldsymbol{u}}$ and $\tilde{C}_A$, and use LU factorisation for inverting the single blocks in their block diagonals $\mathcal{F}_{\boldsymbol{u},k}$ and $\tilde{\mathcal{C}}_{A,k}$ (see their definitions in (4.35) and (4.71)), as well as the vector potential mass matrix $\mathcal{M}_A$ (appearing within M_A in (4.73)), the current intensity mass matrix $\mathcal{M}_j$ (present in M_j in (4.74)), and the pressure mass and stiffness operators $\mathcal{M}_p$, $\tilde{\mathcal{K}}_p$ (included in (3.37) and discussed in Sec. 3.2).

Notice moreover that the results included in this section only consider the application of the *simplified* block upper-triangular preconditioner P_T in (4.79), since

Table 4.1: Iterations to convergence for Newton method applied for the solution of problems 1 and 2, for various refinement levels of the spatial and temporal grids (the temporal domain is fixed at $T = 1$). The outer solver achieves convergence with a tolerance of 10^{-10}. The inner GMRES solver is right-preconditioned with P_T (4.79), and converges with relative tolerance 10^{-2} or absolute tolerance of 10^{-14}. To invert the relevant operators appearing in P_T, we use exact solvers (time-stepping and LU). Inside each column, the number of Newton iterations to convergence is reported on the left, the average number of inner GMRES iterations per outer Newton iteration appears on the right (in brackets).

Pb	$\Delta t\rightarrow$ $\Delta x\downarrow$	2^{-2}		2^{-3}		2^{-4}		2^{-5}		2^{-6}		2^{-7}	
1	2^{-2}	5	(7.80)	5	(9.40)	5	(11.60)	5	(13.20)	5	(13.60)	5	(16.40)
	2^{-3}	5	(8.40)	5	(10.20)	5	(13.00)	5	(14.80)	5	(15.00)	5	(16.60)
	2^{-4}	5	(8.00)	5	(10.40)	5	(12.20)	5	(13.80)	5	(15.40)	5	(17.20)
	2^{-5}	5	(7.40)	5	(9.40)	5	(12.00)	5	(13.40)	5	(14.00)	5	(18.40)
	2^{-6}	5	(7.60)	5	(9.40)	5	(12.00)	5	(13.40)	5	(13.80)	5	(17.00)
	2^{-7}	5	(7.60)	5	(9.80)	5	(11.80)	5	(13.20)	5	(14.40)	5	(16.80)
2	2^{-2}	4	(8.75)	5	(11.00)	5	(11.80)	5	(14.20)	4	(15.75)	4	(17.00)
	2^{-3}	5	(9.20)	4	(11.50)	4	(13.00)	4	(14.75)	4	(16.75)	4	(18.75)
	2^{-4}	4	(9.75)	4	(11.25)	4	(13.50)	4	(15.50)	4	(17.25)	4	(19.25)
	2^{-5}	4	(8.00)	4	(11.00)	4	(13.75)	4	(15.25)	3	(15.67)	3	(20.33)
	2^{-6}	3	(8.00)	3	(10.00)	3	(12.33)	3	(14.00)	3	(15.67)	3	(17.33)
	2^{-7}	3	(7.67)	3	(10.67)	3	(12.33)	3	(14.00)	3	(16.00)	3	(17.67)

early experiments showed negligible difference in convergence behaviour with respect to using its *full* version P from (4.75) (which makes P_T a much more appealing choice, given how its associated computational cost is significantly reduced, as per the discussion in Sec. 4.2.3).

In Tab. 4.1 we report convergence results for both inner and outer solvers, collected for various choices of spatial and temporal grid sizes Δx and Δt, on a fixed temporal domain of size $T = 1$. There, we see how the number of Newton iterations necessary for convergence remains substantially unchanged, regardless of the mesh chosen. The number of GMRES iterations seems to be increasing slightly with temporal refinement, in a somewhat more evident manner than for the fluid dynamics problems in Tab. 3.1, but the rate at which this occurs remains reasonably bounded (increasing the number of temporal nodes by a factor of 2^5 results in slightly more than ~ 2 times as many iterations). To put these numbers into perspective, notice that for these experiments the spatial unknowns for the most refined discretisation of problem 1 amount to $N_{\boldsymbol{u}} = 222,722$, $N_p = 49,665$, $N_j = N_A = 12,545$, while those for problem 2 reach $N_{\boldsymbol{u}} = 296,450$, $N_p = 66,049$, $N_j = N_A = 16,641$. Considering that the temporal grids count up to $N_t = 128$ nodes, this results in space-time sys-

Table 4.2: Convergence results for Newton iterations applied for the solution of problems 1 and 2, for various refinement levels of the spatial grid. We fix $\Delta t = 0.5$, and progressively enlarge the temporal domain to accommodate for more processors. The solvers used are as in Tab. 4.1. Inside each column, the number of Newton iterations to convergence is reported on the left, the average number of inner GMRES iterations per outer Newton iteration appears on the right (in brackets).

Pb	$T\rightarrow$ $\Delta x\downarrow$	2^2		2^3		2^4		2^5		2^6		2^7	
1	2^{-2}	5	(5.80)	5	(6.20)	5	(6.20)	5	(6.60)	5	(6.40)	5	(6.40)
	2^{-3}	5	(6.80)	5	(6.80)	5	(6.80)	5	(6.80)	5	(6.80)	5	(6.80)
	2^{-4}	5	(6.60)	5	(6.60)	5	(6.60)	5	(6.60)	5	(6.60)	5	(6.80)
	2^{-5}	4	(6.00)	5	(6.20)	4	(6.25)	4	(6.25)	5	(6.40)	5	(6.40)
	2^{-6}	4	(6.00)	4	(6.00)	4	(6.00)	4	(6.00)	4	(6.25)	4	(6.25)
	2^{-7}	4	(5.75)	4	(6.00)	4	(6.00)	4	(6.00)	4	(6.00)	4	(6.00)
2	2^{-2}	5	(6.40)	5	(6.40)	5	(6.40)	5	(6.40)	5	(6.40)	5	(6.40)
	2^{-3}	4	(6.75)	5	(7.00)	5	(7.00)	5	(7.00)	5	(7.00)	5	(7.00)
	2^{-4}	4	(6.25)	4	(6.25)	4	(6.50)	4	(6.75)	5	(6.80)	5	(6.80)
	2^{-5}	4	(6.50)	4	(6.50)	4	(6.75)	4	(6.50)	4	(6.50)	4	(6.50)
	2^{-6}	3	(6.33)	4	(6.75)	4	(6.75)	4	(6.75)	4	(6.75)	4	(6.75)
	2^{-7}	3	(6.33)	3	(6.33)	3	(6.33)	3	(6.33)	4	(6.75)	3	(6.33)

tems of total size $(N_{\boldsymbol{u}} + N_p + N_j + N_A) \cdot N_t \approx 38 \cdot 10^6$ and $\approx 52 \cdot 10^6$ for the two problems, respectively.

The temporal domain $T = 1$ considered for the experiments in Tab. 4.1 is however too small to allow for the relevant nonlinear behaviours described in Sec. 4.4.1 to manifest themselves into the solution (notice the time labels on the side of Fig. 4.1 and 4.3). To get an idea of how the preconditioner performs when these become evident, we consider another set of experiments, where we fix the time-step and progressively increase the size of the time domain. The corresponding convergence results are reported in Tab. 4.2. Overall, they draw a similar picture as those in Tab. 4.1, with the number of both outer and inner iterations kept bounded in all cases. In this test, though, we do not observe the slight increase in iterations count present in Tab. 4.1 as we increase the domain size (indeed, iterations number barely changes at all), providing evidence that the performance of the preconditioner is affected by the actual value of Δt, rather than by an increase in the number of processors available.

4.4.3 Comparison with sequential time-stepping

In this section, we expand on the discussion in Sec. 3.2.1.1 and provide an indication of how the number of inner and outer iterations to convergence for the space-time

Table 4.3: Ratios $N^{ST}_{it,NL}/N^{0}_{it,NL}$ and $N^{ST}_{it,L}/N^{0}_{it,L}$, where $N^{ST}_{it,NL}$ and $N^{ST}_{it,L}$ are the total number of outer and inner iterations to convergence for the space-time approach (resulting from the same experiments used to fill Tab. 4.1), while $N^{0}_{it,NL}$ and $N^{0}_{it,L}$ represent the average number of outer and inner iterations to convergence per time-step, using time-stepping. For the latter, the inner solver for each time-step is GMRES, right-preconditioned with the single time-step counterpart of the block upper-triangular preconditioner, (4.80). The outer solver for time-stepping reaches convergence with absolute tolerance $10^{-10}/\sqrt{N_t}$, while the inner solver has the same tolerances used for Tab. 4.1. For some of the most extreme parameters combinations, we noticed that the solution from time-stepping "freezes" from a certain (reasonably late) instant onwards: that is, the residual after the Newton update becomes smaller than the prescribed tolerance, thus preventing any other update from being carried out (notice we do not implement mesh adaptivity). To account for this, we compute the averages $N^{0}_{it,NL}$ and $N^{0}_{it,L}$ by considering the *effective* time-steps only.

Pb	$\Delta t\rightarrow$ $\Delta x\downarrow$	2^{-2}		2^{-3}		2^{-4}		2^{-5}		2^{-6}		2^{-7}	
1	2^{-2}	1.05	(1.05)	1.14	(1.18)	1.31	(1.45)	1.38	(1.54)	1.63	(1.97)	1.66	(2.41)
	2^{-3}	1.11	(1.17)	1.18	(1.29)	1.27	(1.54)	1.45	(1.80)	1.54	(2.15)	1.65	(2.40)
	2^{-4}	1.11	(1.21)	1.21	(1.40)	1.40	(1.65)	1.47	(1.82)	1.50	(2.35)	1.65	(2.73)
	2^{-5}	1.11	(1.13)	1.25	(1.38)	1.45	(1.79)	1.55	(1.93)	1.66	(2.51)	1.86	(3.50)
	2^{-6}	1.11	(1.21)	1.33	(1.50)	1.48	(1.89)	1.60	(2.06)	1.72	(2.71)	1.91	(3.52)
	2^{-7}	1.25	(1.42)	1.29	(1.56)	1.51	(1.94)	1.68	(2.21)	1.77	(3.06)	2.01	(3.91)
2	2^{-2}	1.07	(1.12)	1.54	(1.73)	1.57	(1.78)	1.72	(2.25)	1.48	(2.13)	1.59	(2.44)
	2^{-3}	1.54	(1.79)	1.39	(1.76)	1.60	(2.05)	1.60	(2.41)	1.70	(3.25)	1.79	(3.70)
	2^{-4}	1.23	(1.53)	1.45	(1.90)	1.60	(2.23)	1.78	(2.68)	1.95	(4.36)	2.18	(5.26)
	2^{-5}	1.45	(1.60)	1.45	(1.96)	1.78	(2.67)	2.03	(3.19)	1.60	(3.40)	1.90	(5.36)
	2^{-6}	1.09	(1.33)	1.26	(1.63)	1.55	(2.20)	1.57	(2.38)	1.79	(4.21)	1.98	(5.20)
	2^{-7}	1.20	(1.39)	1.41	(2.00)	1.60	(2.38)	1.75	(2.83)	1.88	(5.01)	2.04	(5.99)

preconditioner P_T, ($N^{ST}_{it,L}$ and $N^{ST}_{it,NL}$), compare with those of its single time-step counterpart $\mathcal{P}_{T,k}$, ($N^{0}_{it,L}$ and $N^{0}_{it,NL}$), for the test cases introduced in Sec. 4.4.1.

In order to conduct a balanced comparison between the single time-step and the whole space-time approaches, in Sec. 3.4.4 we adjusted the absolute tolerance of the linear solver for the former by scaling it by a factor $\sqrt{N_t}$. In this section, we do the same for the *outer* solver, but leave that of the inner solver untouched (this might slightly favour the time-stepping approach, but convergence of GMRES occured due to the *relative* tolerance being met in almost the totality of cases). Moreover, to more faithfully represent a real-case scenario, when time-stepping we reuse the value of the solution at the previous instant as initial guess for the Newton solver at the following time-step, analogously to what has been done in Sec. 3.4.4.

Given the remarks in Sec. 4.2.4, it is unclear whether a relative increase in *outer* iterations is more detrimental in raising the total computational time with respect to an increase in *inner* iterations, or if the converse holds instead. Regardless, the

fact remains that both ratios $N^{ST}_{it,L}/N^{0}_{it,L}$ and $N^{ST}_{it,NL}/N^{0}_{it,NL}$ must be kept as close as possible to 1 for the space-time preconditioning approach to represent an appealing alternative to sequential time-stepping. The results in Tab. 4.3 show that this condition is satisfied reasonably well: the ratio of Newton iterations hits a maximum of ~ 2, while that of GMRES iterations sits between 1 and 3 for most cases. As $\Delta t \to 0$, this ratio tends to increase, reaching almost 6 in certain instances. This is not surprising, as in this case the improved quality of the initial guess for Newton when using time-stepping (see the discussion in Sec. 3.4.4), and the slight increase in iterations count observed in Tab. 4.1 both play against the space-time approach.

Overall, the results in this section showcase the potential effectiveness of our STBP approach also in this more complex MHD case. At least when employing *exact* solvers, the space-time preconditioner requires limited overhead in terms of number of iterations to convergence with respect to its single time-step equivalent. This allows significant room for parallel efficiency, so long as effective PinT methods are developed to solve the single-variable space-time systems appearing within our preconditioner, namely involving operators $F_{\boldsymbol{u}}$ and $\tilde{C}_A$. Compared to having to parallelise in time the solution of the whole multi-variable system (4.44), this task is now simplified, by virtue of our space-time block preconditioner.

4.5 Limitations of STBP for compressible MHD

In Sec. 3.5, we remarked on how, rather than being a time-parallelisation method in and of itself, the space-time block preconditioning procedure outlined here and in Chap. 3 represents more of a general strategy for simplifying the application of already-existing parallel-in-time algorithms to systems of PDEs. As such, this implies that its performance is ultimately reliant on that of the aforementioned algorithms.

When dealing with the incompressible kind of flow analysed in the last two chapters, this dependence is not particularly burdensome: in fact, we mostly need to solve space-time systems stemming from discretisations of single-variable *parabolic* equations, which most traditional PinT methods have already proven to handle quite effectively, as shown in the introductory discussion in Chap. 2. However, this ceases to be the case when dealing with applications to compressible flow: within that framework, one can expect the resulting target single-variable PDEs to be *hyperbolic* in nature instead, thus posing a much greater challenge to classical time-parallelisation schemes (see Sec. 2.1, and more specifically Sec. 2.1.1.1).

Therefore, rather than considering magnetohydrodynamics problems in the compressible regime, the work described in the following chapters is mostly directed at understanding the mechanism behind the failure of PinT algorithms when applied to hyperbolic problems (Chap. 5), and in trying to determine the validity of alternative approaches (Chap. 6). This also dictates a shift from the tone used in Chap. 3 and 4, with most of the findings illustrated next revolving around the investigation of the performance of PinT methods available in the literature, rather than the description of novel techniques.

Chapter 5

Multigrid Reduction in Time for nonlinear hyperbolic equations

Recovering an accurate numerical solution to hyperbolic equations *per se* is challenging in many ways [89, Chap. 1.4], with one of the biggest difficulties lying in being able to properly capture shocks present in the solution, while retaining the desired level of accuracy and ensuring stability of the scheme used. These objectives are often at conflict with each other: better accuracy can be achieved using high-order approximations, but these approximations might generate spurious oscillations in proximity of discontinuities in the solution, undermining the stability of the algorithm. To counteract this behaviour, a variety of schemes have been proposed in the literature, some of which are described in this chapter.

When classical approaches to time parallelisation, such as Parareal and MGRIT, come into play, this problem becomes even more pronounced, as solutions from different integrators need to be combined together. Even if both fine and coarse solvers are individually stable, often their combined use triggers instabilities that, in turn, result in loss of accuracy and poor convergence, negating most of the advantages from parallelisation. Examples of successful applications, however, still remain even in this case, as outlined in the discussion in Sec. 2.1.1.1.

The aim of this chapter is to investigate in detail the performance of MGRIT when applied to some nonlinear hyperbolic problems frequently employed as test-cases. In particular, we discuss the behaviour of the algorithm when used in conjunction with high-order discretisations, which are frequently used for the solution of conservation laws. Example applications of parallel-in-time methods to high-order schemes are, to our knowledge, somewhat rare in the literature. Noticeable exceptions are: publications involving the PFASST algorithm [43], which achieves high-order accuracy

via repeated SDC corrections, but is limited to this very specific class of integrators; the works conducted in [48, 49], where k-th order multi-step BDF methods are analysed; [109], which shares a similar set-up to ours, but only shows results for a specific configuration of the integrators used; finally, [33], which is possibly the work whose goal is the closest to the one of this chapter. There, the efficacy of MGRIT applied to hyperbolic equations is studied in detail, with the aim of identifying suitable coarse integrators to achieve fast convergence, though the project only covers the linear case. The purpose of our study lies in investigating the applicability of MGRIT to the acceleration of solutions to nonlinear systems of conservation laws which are relevant in real-world scenarios. It focuses on determining the principal factors that cause degradation in the convergence speed of the algorithm, and identifies future directions for improvements.

The work in this chapter has been conducted while on a research visit at the Memorial University of Newfoundland, under the supervision of Scott MacLachlan. The corresponding findings have been submitted to the *Electronic Transaction on Numerical Analysis* journal, and are currently under review. A pre-print version can be found in [30]. The remainder of this chapter is structured as follows. In Sec. 5.1, we introduce the test problems considered here and the discretisations used in our experiments. In Sec. 5.2, we expand on the description of the MGRIT algorithm provided in Sec. 2.2.3, and discuss the important choice of the coarse integrator for a given fine integrator. In Sec. 5.3, we present and discuss numerical experiments.

5.1 Model problems

As test cases for our experiments, we pick examples of *conservation laws* that, due to their physical relevance, are widely used in the analysis and testing of numerical methods for hyperbolic equations. For simplicity, we limit ourselves to problems defined on a one-dimensional spatial domain, and with periodic boundary conditions. These equations can be written in the following general form:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \dfrac{\partial \boldsymbol{f}(\boldsymbol{u})}{\partial x} = \boldsymbol{0} & (x,t) \in [0,L] \times (0,T] \\ \boldsymbol{u}(x,0) = \bar{\boldsymbol{u}}^0(x) & x \in [0,L] \\ \boldsymbol{u}(0,t) = \boldsymbol{u}(L,t) & t \in (0,T] \end{cases}, \tag{5.1}$$

where the variable $\boldsymbol{u}(x,t) \in \mathbb{R}^D$ contains the vector of conserved variables in the system, or *state*, with associated *flux* $\boldsymbol{f}(\boldsymbol{u})$. The parameters L and T define the size

of the spatial and temporal domains, respectively, while $\bar{\boldsymbol{u}}^0(x)$ identifies the inital condition. We consider three different systems, ubiquitous in the literature.

Problem 1 The *Burgers' equation*, considered here in its inviscid formulation, which is possibly the simplest example of a scalar conservation law including nonlinear effects. It is defined by

$$\boldsymbol{u} = u, \qquad \boldsymbol{f}(\boldsymbol{u}) = \frac{u^2}{2}, \tag{5.2}$$

with u representing the flow velocity.

Problem 2 The *shallow-water equations*, also considered without viscosity, which compose a system describing a fluid flow in the regime where the vertical length scale is negligible with respect to the horizontal one. The associated state and flux are, respectively

$$\boldsymbol{u} = \begin{bmatrix} h \\ hu \end{bmatrix}, \qquad \boldsymbol{f}(\boldsymbol{u}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix}, \tag{5.3}$$

with g being the gravitational constant, u the velocity, and h the height of the fluid column.

Problem 3 The *Euler equations*, another system governing compressible fluid flow, with

$$\boldsymbol{u} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, \qquad \boldsymbol{f}(\boldsymbol{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (E + p)u \end{bmatrix} \tag{5.4}$$

where ρ is the flow density, u is once again its velocity, E its total internal energy, and p its pressure. In our case, this system is closed by considering an ideal monoatomic gas, which gives the following relationship between pressure and energy:

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho u^2\right), \quad \text{with} \quad \gamma = \frac{5}{3}. \tag{5.5}$$

These three problems are presented in detail in [89, Chap. 3.2, Chap. 5.4, and Chap. 5.1], respectively.

In the following, we provide a description of the numerical schemes employed in our experiments in order to recover their approximate solution.

5.1.1 Space discretisation

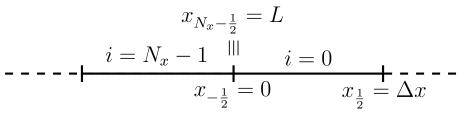
To simplify our notation, in this section we consider the state of the conservation law as being a scalar, unless otherwise specified, although the treatment described here can be seamlessly extended to the systems listed above.

Equation (5.1) is discretised via a method of lines approach, using finite volumes in space. The spatial domain $[0, L]$ is subdivided into N_x cells of uniform length $\Delta x = L/N_x$. The semi-discretised unknown is approximated by a vector $\boldsymbol{u}(t) \in \mathbb{R}^{N_x} \approx u(x,t)$, whose i-th component represents the cell-average

$$u_i(t) \approx \frac{1}{\Delta x} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} u(x,t)\, dx, \qquad i = 0, \dots, N_x - 1. \tag{5.6}$$

Here, $x_{i\pm\frac{1}{2}}$ identifies the coordinate of the right (respectively, left) interface of the i-th cell,

$$x_{i\pm\frac{1}{2}} = \left(i + \frac{1 \pm 1}{2}\right) \Delta x, \tag{5.7}$$



where the right-interface of cell $N_x - 1$, (on coordinate $x_{N_x-\frac{1}{2}} = L$), logically coincides with the left-interface of cell 0, (on coordinate $x_{-\frac{1}{2}} = 0$), due to the periodic boundary conditions being applied. Integrating equation (5.1) on cell i, we obtain

$$\frac{\partial u_i(t)}{\partial t} = -\frac{f_{i+\frac{1}{2}}(t) - f_{i-\frac{1}{2}}(t)}{\Delta x} =: \left[\mathcal{A}(\boldsymbol{u}(t))\right]_i, \tag{5.8}$$

where $f_{i+\frac{1}{2}}(t)$ defines an approximation of the flux at the cell interface at $x_{i+\frac{1}{2}}$:

$$f_{i+\frac{1}{2}}(t) \approx f\left(u\left(x_{i+\frac{1}{2}}, t\right)\right). \tag{5.9}$$

Typically, this approximation must be built starting from values of the numerical solution on the surrounding cells. There is, by far, no unique way to achieve this and, indeed, several schemes have been proposed in the literature, based on different methods for reconstructing the values in (5.9). Our experiments are based primarily on the WENO scheme, which is briefly introduced in Sec. 5.1.1.1. In Sec. 5.1.1.2 and Sec. 5.1.1.3, we illustrate the two procedures employed to recover the fluxes in (5.9) using WENO.

5.1.1.1 WENO reconstruction

The *Weighted Essentially Non-Oscillatory* scheme (or WENO) was originally developed by Liu, Chan and Osher in [94]. In this section, we aim at providing a short description of the main idea behind the method, but we refer to [75, Chap. 11.3] for a more complete treatment.

The aim of the WENO procedure is to provide a stable, high-order approximation of a certain value of interest (usually the state itself, or directly the flux) at the cell interface. We represent these generic recovered quantities as

$$q^{\pm}_{i+\frac{1}{2}} \approx q\left(x^{\pm}_{i+\frac{1}{2}}\right). \tag{5.10}$$

Since discontinuities might arise in the solution of hyperbolic equations, the approximations on the left $(-)$ and right $(+)$ of an interface can differ in general, hence the different superscripts in (5.10). These quantities are approximated via polynomial reconstruction, starting from the cell-values q_i in a stencil containing the interface. However, as pointed out in the introduction to this chapter, solutions to (5.1) can develop shocks: if the stencil used for the polynomial reconstruction happens to contain a sharp discontinuity, then the reconstructed polynomial can show oscillations due to Gibbs phenomenon, and the approximation recovered might be of poor quality. In order to counteract this, the WENO method collects approximations built using a number of different stencils, and opportunely combines them via a convex linear combination, with the weight of each approximation considered depending on an estimate of the smoothness of the particular reconstruction.

In more detail, in order to recover an approximation of the value to the left of the interface, $q^{-}_{i+\frac{1}{2}}$, using a polynomial $\tilde{q}^r_i(x)$ of degree k, one can choose between $k+1$ different stencils each containing $k+1$ cells (see Fig. 5.1). Each of these

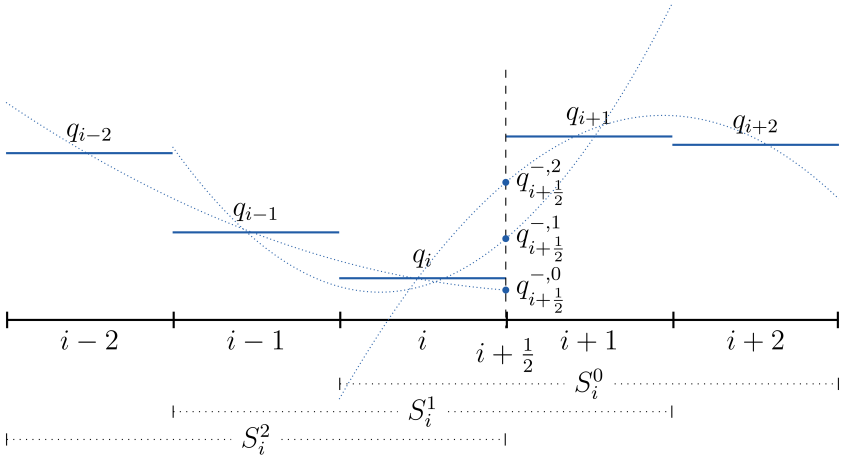


Figure 5.1: Choice of stencils S^r_i for the polynomial reconstructions of degree $k=2$ of the value left of the interface, $q^{-}_{i+\frac{1}{2}}$, starting from the cell values $q_{i-k},\dots,q_{i+k}$ (flat blue lines). Different stencils produce different approximating polynomials $\tilde{q}^r_i(x)$ (dotted blue lines) which, in turn, give different values at the interface (blue dots)

approximations can be expressed as a linear combination of the cell-averages,

$$q^{-,r}_{i+\frac{1}{2}} = \tilde{q}^r_i\left(x_{i+\frac{1}{2}}\right) = \sum_{j\in S^r_i} c_{r,j-(i+r-k)}\, q_j, \tag{5.11}$$

where here (and in the remainder of this section) r ranges from 0 to k; the stencil S^r_i contains the cells with indices $S^r_i := \{i+r-k,\dots,i+r\}$. The quality of the r-th reconstruction $q^{-,r}_{i+\frac{1}{2}}$ on cell i is measured via a *smoothness indicator*, β^r_i, based on the (scaled) L^2 norm of the derivatives of the reconstructed polynomial $\tilde{q}^r_i(x)$ on S^r_i. This can be expressed as a bilinear form,

$$\beta^r_i := \sum_{l=1}^{k} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \Delta x^{2l-1} \left(\frac{\partial^l \tilde{q}^r_i(x)}{\partial x^l}\right)^2 dx = (\boldsymbol{q}^r_i)^T B_r\, \boldsymbol{q}^r_i, \tag{5.12}$$

represented by $B_r \in \mathbb{R}^{k\times k}$, with $\boldsymbol{q}^r_i \in \mathbb{R}^k$ being the vector collecting the values of q_j on the stencil considered, $j \in S^r_i$. These smoothness indicators contribute to the definition of the weights ω^r_i as follows:

$$\omega^r_i := \frac{\alpha^r_i}{\sum_{r=0}^k \alpha^r_i}, \quad \text{and} \quad \alpha^r_i := \frac{d_r}{(\varepsilon+\beta^r_i)^2}, \tag{5.13}$$

where ε is a small parameter to prevent division by 0, (usually $\varepsilon = 10^{-6}$). The coefficients d_r are defined in such a way that, in the smooth case,

$$q_{i+\frac{1}{2}} = \sum_{r=0}^{k} d_r q^r_{i+\frac{1}{2}} = q\left(x_{i+\frac{1}{2}}\right) + \mathcal{O}(\Delta x^s) \tag{5.14}$$

holds with $s = 2k+1$. This way, the choice (5.13) of the weights ω^r_i ensures that the recovered reconstruction is a high-order approximation of the underlying quantity. Finally, the actual reconstruction is computed as a convex combination of all the different approximations in (5.11),

$$q^{-}_{i+\frac{1}{2}} = \sum_{r=0}^{k} \omega^r_i\, q^{-,r}_{i+\frac{1}{2}}. \tag{5.15}$$

The specific values of the parameters $c_{r,j}$, d_r, and B_r described above depend directly on the degree k of the polynomials employed. For example, for $k = 2$, we have:

$$[c_{r,j}] := \frac{1}{6}\begin{bmatrix} 2 & 5 & -1 \\ -1 & 5 & 2 \\ 2 & -7 & 11 \end{bmatrix}, \qquad [d_r] := \frac{1}{10}\begin{bmatrix} 3 \\ 6 \\ 1 \end{bmatrix}, \qquad \text{and}$$

$$[B_0|B_1|B_2] := \frac{1}{6}\left[\begin{array}{ccc|ccc|ccc} 20 & -31 & 11 & 8 & -13 & 5 & 8 & -19 & 11 \\ -31 & 50 & -19 & -13 & 26 & -13 & -19 & 50 & -31 \\ 11 & -19 & 8 & 5 & -13 & 8 & 11 & -31 & 20 \end{array}\right]. \tag{5.16}$$

An analogous procedure is used to recover the value to the right of the interface, $q^+_{i+\frac{1}{2}}$. We also point out that, if piecewise constant polynomials are chosen ($k = 0$), then the reconstruction becomes trivial: $q^-_{i+\frac{1}{2}} = q_i$, and $q^+_{i+\frac{1}{2}} = q_{i+1}$.

When dealing with a system of conservation laws, additional care is necessary to recover a proper reconstruction of the state. A naïve extension from the scalar case would consider reconstructing each component of the state independently, but this has been shown to cause some unphysical oscillations in the solution [116]. To limit this effect, a more stable approach rather consists of independently reconstructing the *characteristic* variables of the system, since these are more readily associated with the information carried by the characteristics [75, Chap. 11.3.4]. This comes at an additional cost, as it requires a local decomposition of the state in each of the cells considered for reconstruction on a given reference state: this procedure is described in [136, Procedure 2.8], where an averaged state at the interface is taken as a reference. Even though we broadly follow these guidelines in our implementation, we decide for simplicity to decompose the values $\boldsymbol{u}_{i-k}, \dots, \boldsymbol{u}_{i+k}$ using the central cell value $\boldsymbol{u}_i$ as a reference state, in order to recover $\boldsymbol{u}^-_{i+\frac{1}{2}}$ and $\boldsymbol{u}^+_{i-\frac{1}{2}}$.

With the WENO procedure available to reconstruct the states to the left and right of each interface, we proceed to approximate the numerical flux (5.9) in two different ways. These are described next.

5.1.1.2 Lax-Friedrichs flux

We first consider the Lax-Friedrichs definition for the numerical flux [136, Chap. 2.3.1]. For a system of conservation laws, this is given by

$$\boldsymbol{f}^{LF}_{i+\frac{1}{2}} \coloneqq \frac{1}{2}\left(\boldsymbol{f}\left(\boldsymbol{u}^+_{i+\frac{1}{2}}\right) + \boldsymbol{f}\left(\boldsymbol{u}^-_{i+\frac{1}{2}}\right)\right) + \frac{1}{2}\alpha_{LF}\left(\boldsymbol{u}^+_{i+\frac{1}{2}} - \boldsymbol{u}^-_{i+\frac{1}{2}}\right). \tag{5.17}$$

This flux is one of the simplest to prescribe; however, as a downside, it typically produces smeared out numerical solutions, since it adds artificial diffusion to the system. The amount of numerical diffusion added depends directly on the parameter α_{LF}: for our implementation, we choose the (somewhat loose) value of

$$\alpha_{LF} \coloneqq \max_{i,k}\left\{\left|\lambda_k\left(\boldsymbol{u}^{\pm}_{i+\frac{1}{2}}\right)\right|\right\}, \qquad \begin{matrix} i = & 0, \dots, N_x - 1 \\ k = & 0, \dots, D-1 \end{matrix}, \tag{5.18}$$

where the λ_k's are the D eigenvalues (if the system is composed of D equations) of the Jacobian of the flux:

$$J_{\boldsymbol{f}}(\boldsymbol{u})\,\boldsymbol{r}_k = \lambda_k \boldsymbol{r}_k, \quad \text{with} \quad \left[J_{\boldsymbol{f}}(\boldsymbol{u})\right]_{m,n} \coloneqq \frac{\partial\left[\boldsymbol{f}(\boldsymbol{u})\right]_m}{\partial\left[\boldsymbol{u}\right]_n}, \tag{5.19}$$

where the $\boldsymbol{r}_k$'s represent the corresponding eigenvectors.

5.1.1.3 Roe flux

The second scheme investigated in our experiments uses Roe's approximate Riemann solver [89, Chap. 14.2] in order to recover the numerical flux. For each interface, this solver targets a linearisation of the Riemann problem defined by (5.1) and the left and right values computed using the WENO procedure. This problem is written as:

$$\boldsymbol{u}_t + J_{\boldsymbol{f}}(\hat{\boldsymbol{u}})\, \boldsymbol{u}_x = 0, \tag{5.20}$$

where $\hat{\boldsymbol{u}}$ is an opportunely defined *Roe-averaged* state. All relevant variables in this section are to be intended as evaluated at an interface: we drop the subscripts $i+1/2$ for ease of notation. The solution to the linear Riemann problem (5.20), as well as the associated flux at the interface, are both easily expressed in terms of the eigenvalues and eigenvectors of the Jacobian. These are given by

$$\begin{matrix} \lambda_0^S = u - c_S \\ \lambda_1^S = u + c_S \end{matrix} \quad \text{and} \quad \begin{matrix} \boldsymbol{r}_0^S = \left[\ 1, & \lambda_0^S\ \right]^T \\ \boldsymbol{r}_1^S = \left[\ 1, & \lambda_1^S\ \right]^T \end{matrix} \tag{5.21}$$

for the shallow-water system, with *speed of sound* $c_S \coloneqq \sqrt{gh}$, and by

$$\begin{matrix} \lambda_0^E = u - c_E \\ \lambda_1^E = u \\ \lambda_2^E = u + c_E \end{matrix} \quad \text{and} \quad \begin{matrix} \boldsymbol{r}_0^E = \left[\ 1, & \lambda_0^E, & H - uc_E\ \right]^T \\ \boldsymbol{r}_1^E = \left[\ 1, & \lambda_1^E, & u^2/2\ \right]^T \\ \boldsymbol{r}_2^E = \left[\ 1, & \lambda_2^E, & H + uc_E\ \right]^T \end{matrix} \tag{5.22}$$

for the Euler equations, with speed of sound $c_E \coloneqq \sqrt{(\gamma-1)(H-u^2/2)}$. Here, $H \coloneqq (E+p)/\rho$ is the *specific enthalpy*. For Burgers' equation, the only eigenvalue is given by $\lambda_0^B = u$ itself. Since we are only interested in the eigenvalues and eigenvectors of the Jacobian $J_{\boldsymbol{f}}(\hat{\boldsymbol{u}})$ in (5.20), (namely $\hat{\lambda}_k$ and $\hat{\boldsymbol{r}}_k$), it suffices to define the following Roe-averaged quantities: an average velocity for the Burgers' equation, given by

$$\hat{u} \coloneqq \frac{u^+ + u^-}{2}; \tag{5.23}$$

average height and velocity for the Shallow-water equation, prescribed as

$$\hat{h} \coloneqq \frac{h^+ + h^-}{2} \qquad \text{and} \qquad \hat{u} \coloneqq \frac{u^+\sqrt{h^+} + u^-\sqrt{h^-}}{\sqrt{h^+} + \sqrt{h^-}}; \tag{5.24}$$

finally, average velocity and specific enthalpy for the Euler equations:

$$\hat{u} \coloneqq \frac{u^+\sqrt{\rho^+} + u^-\sqrt{\rho^-}}{\sqrt{\rho^+} + \sqrt{\rho^-}} \qquad \text{and} \qquad \hat{H} \coloneqq \frac{(E^+ + p^+)\sqrt{\rho^+} + (E^- + p^-)\sqrt{\rho^-}}{\sqrt{\rho^+} + \sqrt{\rho^-}}. \tag{5.25}$$

Since the target problem (5.20) is linear, this procedure is known to provide a non-entropic weak solution when transonic rarefaction waves arise [89, Chap. 14.2.2]. To

counteract this, we also apply the *entropy fix* proposed by Harten and Hyman in [72]. In the general case of a system of D conservation laws, this gives rise to the following formula for the numerical flux at the interface:

$$\boldsymbol{f}^R \coloneqq \frac{1}{2}\left(\boldsymbol{f}\left(\boldsymbol{u}^-\right) + \boldsymbol{f}\left(\boldsymbol{u}^+\right)\right) - \sum_{k=0}^{D-1} q_H(\hat{\lambda}_k)\,\alpha_k\,\hat{\boldsymbol{r}}_k. \tag{5.26}$$

Here, α_k is the *shock strength*, that is the jump in the k-th characteristic variable between the states left and right of the interface: $\alpha_k \coloneqq (\boldsymbol{u}^+ - \boldsymbol{u}^-)\cdot\hat{\boldsymbol{r}}_k$. Also, $q_H(\lambda_k)$ defines the entropy fix:

$$q_H(\hat{\lambda}_k) \coloneqq \begin{cases} \frac{1}{2}\left(\frac{\hat{\lambda}_k}{\delta_k} + \delta_k\right) & \text{if} \quad |\hat{\lambda}_k| < \delta_k \\ |\hat{\lambda}_k| & \text{else} \end{cases}, \tag{5.27}$$

where $\delta_k \coloneqq \max\left\{0, \hat{\lambda}_k - \lambda_k^-, \lambda_k^+ - \hat{\lambda}_k\right\}$, and $\lambda_k^\pm$ are the eigenvalues of $J_{\boldsymbol{f}}(\boldsymbol{u}^\pm)$.

Substituting (5.17) or (5.26) into (5.8) identifies the right-hand side $\mathcal{A}(\boldsymbol{u})$ of our semi-discretised equation. The details regarding the discretisation in time, as well as the definition of the time-steppers employed, are presented next.

5.1.2 Time discretisation

The time domain is also discretised using a uniform grid of N_t nodes, with time step $\Delta t = T/N_t$. The unknowns at each instant $t_n = n\Delta t$, $n = 0, \ldots, N_t - 1$ are approximated by a vector denoted as $\boldsymbol{u}^n \approx \boldsymbol{u}(t_n)$.

Since we aim at employing high-order spatial reconstructions, it is sensible to request that our temporal discretisation matches this accuracy. The schemes chosen belong to the family of *Strong Stability-Preserving Runge Kutta* methods (SSPRK). As the name suggests, these have the remarkable property of being able to preserve strong stability and, in particular, to be *Total Variation Diminishing*, even while achieving high-order of accuracy; for this reason, they are often employed in conjunction with high-order spatial discretisation of hyperbolic equations. We refer to [67] for a complete review of these schemes, and we only report here the definition of the methods used in our experiments, which vary in the order of accuracy d recovered. They are: a third-order scheme (SSPRK3), whose stepping procedure applied to (5.8) is given by

$$\begin{aligned} \boldsymbol{u}^{n,(1)} &= \boldsymbol{u}^n + \Delta t\,\mathcal{A}\left(\boldsymbol{u}^n\right) \\ \boldsymbol{u}^{n,(2)} &= \frac{3}{4}\boldsymbol{u}^n + \frac{1}{4}\boldsymbol{u}^{n,(1)} + \frac{1}{4}\Delta t\,\mathcal{A}\left(\boldsymbol{u}^{n,(1)}\right) \\ \boldsymbol{u}^{n+1} &= \frac{1}{3}\boldsymbol{u}^n + \frac{2}{3}\boldsymbol{u}^{n,(2)} + \frac{2}{3}\Delta t\,\mathcal{A}\left(\boldsymbol{u}^{n,(2)}\right); \end{aligned} \tag{5.28}$$

a second-order scheme (SSPRK2), defined by

$$\begin{aligned} \boldsymbol{u}^{n,(1)} &= \boldsymbol{u}^n + \Delta t\, \mathcal{A}\left(\boldsymbol{u}^n\right) \\ \boldsymbol{u}^{n+1} &= \frac{1}{2}\boldsymbol{u}^n + \frac{1}{2}\boldsymbol{u}^{n,(1)} + \frac{1}{2}\Delta t\, \mathcal{A}\left(\boldsymbol{u}^{n,(1)}\right); \end{aligned} \tag{5.29}$$

and, finally, the first-order scheme, which reduces to the well-known forward Euler (FE) method,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t\, \mathcal{A}\left(\boldsymbol{u}^n\right). \tag{5.30}$$

5.2 Time parallelisation with MGRIT

In this section we expand on the introduction to MGRIT given in Sec. 2.2.3, so to provide a more in-depth description of the various components of the algorithm used in our experiments. To aid in our exposition, we follow much of [47] and consider the space-time discretisation of a constant-coefficient, linear, homogeneous system using an expicit single-step time integrator. Similarly to (2.26), this system takes the following block Toeplitz, block bi-diagonal form

$$A\boldsymbol{u} = \boldsymbol{g} \quad \Longleftrightarrow \quad \begin{bmatrix} I & & & \\ -\mathcal{F} & I & & \\ & \ddots & \ddots & \\ & & -\mathcal{F} & I \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^0 \\ \boldsymbol{u}^1 \\ \vdots \\ \boldsymbol{u}^{N_t-1} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{u}}^0 \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{bmatrix}, \tag{5.31}$$

where $\boldsymbol{u} \in \mathbb{R}^{N_x N_t}$ is the vector containing the values of the discrete solution at each node in the space-time grid, and $\bar{\boldsymbol{u}}^0$ contains the discretisation of the initial condition; since we do not consider forcing terms, the rest of $\boldsymbol{g}$ is filled with zeros. The *fine integrator* $\mathcal{F}$ represents the action of the time-stepper, so that, for each instant $n = 1, \ldots, N_t - 1$, we have $\boldsymbol{u}^n = \mathcal{F}\boldsymbol{u}^{n-1}$. We proceed to splitting the temporal grid into fine and coarse nodes, and we subdivide the monolithic system (5.31) accordingly, as done in (2.31), and reported here for ease of reference:

$$A = \begin{bmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{bmatrix} = \begin{bmatrix} I & 0 \\ A_{CF}A_{FF}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{FF} & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & A_{FF}^{-1}A_{FC} \\ 0 & I \end{bmatrix}. \tag{5.32}$$

For simplicity, we pick the coarse nodes to be spaced every m nodes, with m being the *coarsening factor* (see Fig. 5.2), so that the coarse nodes subdivide the time domain into $(N_t - 1)/m$ different *time chunks*, each of size $m\Delta t$.

It can be shown that in our case the A_{FF} sub-matrix of (5.32) presents a block diagonal structure,

$$A_{FF} = \begin{bmatrix} B_{\mathcal{F}} & & \\ & \ddots & \\ & & B_{\mathcal{F}} \end{bmatrix}, \quad \text{with} \quad B_{\mathcal{F}} = \begin{bmatrix} I & & & \\ -\mathcal{F} & I & & \\ & \ddots & \ddots & \\ & & -\mathcal{F} & I \end{bmatrix}, \tag{5.33}$$

which implies that systems involving it can be readily solved in parallel. In (5.33), we note that A_{FF} is overall an $N_x(N_t-1)(m-1)/m \times N_x(N_t-1)(m-1)/m$ matrix, written above as an $(N_t-1)/m \times (N_t-1)/m$ block diagonal system, with blocks $B_{\mathcal{F}}$ of size $(m-1)N_x \times (m-1)N_x$. Each solve with $B_{\mathcal{F}}$ involves "stepping" forward an approximate solution on the spatial mesh through $m-2$ applications of $\mathcal{F}$. The challenge lies in finding the solution to the system associated with the Schur complement, which in this case assumes the following shape:

$$S = A_{CC} - A_{CF}A_{FF}^{-1}A_{FC} = \begin{bmatrix} I & & & \\ -\mathcal{F}^m & I & & \\ & \ddots & \ddots & \\ & & -\mathcal{F}^m & I \end{bmatrix}. \tag{5.34}$$

Attempting to solve this directly is equivalent to applying forward block substitution to (5.31), (*i.e.*, time-stepping over all temporal nodes) and would thus nullify any advantage gained from parallelisation. Instead, in the scope of the MGRIT algorithm, we resort to solving a modified system, obtained by substituting another operator $\mathcal{G} \approx \mathcal{F}^m$ in (5.34), giving

$$S_{\mathcal{G}} = \begin{bmatrix} I & & & \\ -\mathcal{G} & I & & \\ & \ddots & \ddots & \\ & & -\mathcal{G} & I \end{bmatrix} \approx S. \tag{5.35}$$

The structure of this system is equivalent to that of (5.31), so that $\mathcal{G}$ can be interpreted as yet another time-stepping routine, which acts only on the coarse nodes: we call this operator the *coarse integrator*. Notice that the splitting into fine and coarse

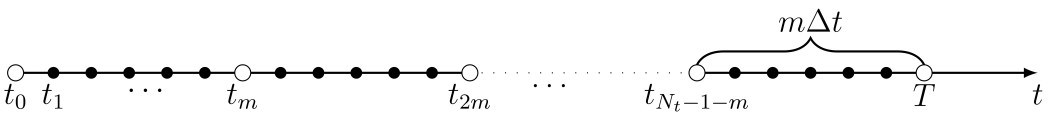


Figure 5.2: Sketch of the partitioning of the time discretisation: white dots represent coarse nodes, black dots are fine nodes. The last time chunk is also highlighted.

nodes can be applied in a recursive fashion, further extracting a hierarchy of N_l grids, together with their corresponding integrators $\mathcal{F}_{(l)}$. The hope is that, by opportunely alternating between solving for the fine nodes (inverting A_{FF}) and for the coarse nodes (inverting $S_{\mathcal{G}}$) and iterating, we can quickly converge to a suitable approximation of the solution of the original system (5.31).

In more detail, at each iteration of the MGRIT algorithm we need to apply the operations described next

Relaxation *Update the solution at the fine nodes of the current level, given a guess at the coarse nodes.* This involves solving a system in the form

$$A_{FF}^{(l)} \boldsymbol{u}_{(l)}^{F} = \boldsymbol{g}_{(l)}^{F} - A_{FC}^{(l)} \boldsymbol{u}_{(l)}^{C}, \tag{5.36}$$

where $A_{FF}^{(l)}$ has the same structure as in (5.33), but whose subdiagonal blocks contain $\mathcal{F}_{(l)}$, while $\boldsymbol{u}_{(l)}^{F}$ and $\boldsymbol{u}_{(l)}^{C}$, respectively, represent the grouping of the unknowns at the fine and coarse nodes of the current level, l. The solution is hence updated by time-stepping using $\mathcal{F}_{(l)}$, starting from the given values at the coarse nodes. Here lies the parallel part of the algorithm, since the time-stepping procedure can be carried out independently on each chunk. In our work, we consider the two types of relaxation briefly introduced in Sec. 2.2.3: *F-relaxation*, where the time-stepping covers a single time chunk, updating the fine nodes within it, and *FCF-relaxation*, where the time-stepping carries on over the following coarse node (performing a *C-relaxation*) and continues on the following time chunk (adding another F-relaxation). The latter is an *overlapping* form of relaxation that requires more work per level of the hierarchy but can be implemented with the same parallel efficiency [47].

Restriction *Transfer information from the nodes of the current level l to the nodes at the coarser level $l+1$.* Simple injection onto the coarse nodes is chosen as the restriction operator:

$$R_{(l+1)}^{(l)} \boldsymbol{u}_{(l)} = \boldsymbol{u}_{(l)}^{C}. \tag{5.37}$$

Since we are dealing with nonlinear equations, we also implement the *Full Approximation Scheme* (FAS) [150, Chap. 5.3.4]. This modifies the right-hand side of the system at the coarse level by adding a correction term:

$$S_{\mathcal{F}_{(l+1)}}\left(\boldsymbol{u}_{(l+1)}\right) = R_{(l+1)}^{(l)}\left(\boldsymbol{g}_{(l)} - S_{\mathcal{F}_{(l)}}\left(\boldsymbol{u}_{(l)}\right)\right) + \underbrace{S_{\mathcal{F}_{(l+1)}}\left(R_{(l+1)}^{(l)} \boldsymbol{u}_{(l)}\right)}_{\text{FAS correction}} = \boldsymbol{g}_{(l+1)}, \tag{5.38}$$

where $S_{\mathcal{F}_{(l+1)}}$ has the same structure as in (5.35), but with $\mathcal{F}_{l+1}$ on the subdiagonal. We point out that the same restriction operator (5.37) is applied to both the residual and the state. Given the particular structure of the operators involved, formula (5.38) simplifies, so that the right-hand side of the coarse-level system, $\boldsymbol{g}_{(l+1)}$, reduces to

$$\boldsymbol{g}^i_{(l+1)} = \boldsymbol{g}^{mi}_{(l)} + \mathcal{F}_{(l)}\left(\boldsymbol{u}^{mi-1}_{(l)}\right) - \mathcal{F}_{(l+1)}\left(\boldsymbol{u}^{m(i-1)}_{(l)}\right), \tag{5.39}$$

for each temporal node $i = 1, \dots, (N_t - 1)/m^{l+1} + 1$ of the coarser level. We also need to provide an initial guess for the solution at the coarse level $\boldsymbol{u}_{(l+1)}$: this is usually chosen to be $\boldsymbol{u}^C_{(l)}$, however we have found that a better alternative is given by

$$\boldsymbol{u}^i_{(l+1)} = \mathcal{F}_{(l)}\left(\boldsymbol{u}^{mi-1}_{(l)}\right) + \boldsymbol{g}^{mi}_{(l)}, \tag{5.40}$$

that is, by performing the last integration step for each chunk and injecting the resulting value. The rationale behind this choice is given by the fact that, if restriction follows relaxation, with (5.40) we are injecting into the coarse level information from the relaxation procedure as well, which should provide a more refined guess over $\boldsymbol{u}^C_{(l)}$. This operation comes at virtually no additional cost (since the quantity in (5.40) is already computed as part of (5.39)), and indeed in early experiments it was seen to improve convergence: in Fig. 5.5, we give an example of the effectiveness of (5.40) over simple injection. Notice that this procedure is *not* equivalent to performing an additional C-relaxation before injection, since the FAS correction (5.38) is still based on the non-updated values at the coarse nodes $\boldsymbol{u}^{m(i-1)}_{(l)}$; notice moreover that this approach *does* retain the fixed-point property of the FAS algorithm, since (5.37) and (5.40) coincide when $\boldsymbol{u}_{(l)}$ is the exact solution on level l.

Coarse grid correction *Update solution at the coarsest level.* This involves solving the system

$$S_{\mathcal{F}_{(N_l-1)}}\boldsymbol{u}_{(N_l-1)} = \boldsymbol{g}_{(N_l-1)}. \tag{5.41}$$

This procedure is sequential but, by choosing a cheap coarse integrator, the overhead to the algorithm can be limited, and parallel efficiency can still be gained.

Interpolation *Transfer information from the nodes at the coarser level $l+1$ to those at the current level l.* We choose *ideal interpolation*, which in our case consists of two steps: an injection from coarse to fine grid, followed by an F-relaxation on the

fine level, starting from the freshly updated values at the coarse nodes. Overall, we have the following definition for the interpolation operator:

$$\begin{aligned}
&\left[I_{(l)}^{(l+1)}\left(\boldsymbol{u}_{(l+1)}\right)\right]^i = \bar{I}^{(l)}_{\lfloor \frac{i}{m} \rfloor, i\%m}\left(\boldsymbol{u}^{\lfloor \frac{i}{m} \rfloor}_{(l+1)}\right), \qquad \text{with} \\
&\bar{I}^{(l)}_{k,r}(\boldsymbol{u}) = \begin{cases} \boldsymbol{u} & \text{if } r = 0, \\ \mathcal{F}_{(l)}\left(\bar{\mathcal{I}}^{(l)}_{k,r-1}(\boldsymbol{u})\right) + \boldsymbol{g}^{k+r}_{(l)} & \text{else,} \end{cases}
\end{aligned} \tag{5.42}$$

where $\lfloor * \rfloor$ and $\%$ denote the floor and modulo operators, respectively. Notice that, if the interpolation step is followed by a relaxation step, then the F-relaxation within interpolation becomes redundant, since the results would get overwritten by the relaxation step. Also notice that here we consider only *pre*-relaxation within the multigrid cycles, without *post*-relaxation. This goes against what is commonly done when applying multigrid to symmetric operators, as it would break the self-adjointness of the multigrid procedure; it is however standard setup in the MGRIT framework, where the target operator is a space-time matrix: as this is non-symmetric to begin with, this choice is not particularly concerning: see the discussion in Sec. 2.2.4 and [153].

The order in which the algorithm moves between discretisation levels defines the sequence in which the operations above are prescribed, and identifies the type of *cycle* used in MGRIT. We experiment both with a *V-cycle* and an *F-cycle* [150, Chap. 3]. In Alg. 4, the pseudo-code outlining the (recursive) definition of a V-cycle with F-relaxation is provided; a sketch of the movement along the various levels for both V- and F-cycle is given in Fig. 5.3.

Algorithm 4: MGRIT, V-cycle iteration

if $l < N_l - 1$ **then**
  `/* Parallel step */`
  Perform relaxation at current level and recover $\boldsymbol{u}^F_{(l)}$ by solving (5.36);
  Compute residual and FAS correction, and restrict to coarser level to compute $\boldsymbol{g}_{(l+1)}$ and $\boldsymbol{u}_{(l+1)}$ following formulae (5.39) and (5.40);
  `/* Recursive step */`
  Invoke this function at coarser level and find $\boldsymbol{u}_{(l+1)}$;
  `/* Parallel step */`
  Interpolate from coarser level to recover $\boldsymbol{u}_{(l)}$ applying (5.42);
else
  `/* Serial step - base step of recursion */`
  Recover $\boldsymbol{u}_{(N_l-1)}$ by solving system (5.41);

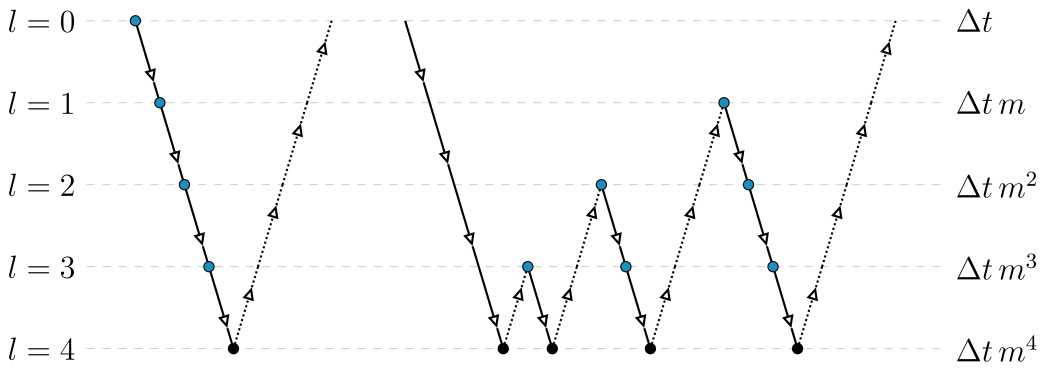


Figure 5.3: Sketch of the flow of the MGRIT algorithm if prescribing a V-cycle (left) or an F-cycle (right), on 5 levels. Restriction (solid lines) is applied when moving from a level to a coarser one (dashed lines: the size of the time-step at each level l is reported on the right). Relaxation (blue dots) is applied at each level, bar the bottom one, where the coarsest system is solved exactly (black dots). Interpolation (dotted lines) is applied when moving from a level to a finer one.

5.2.1 Choices for the coarse integrator

As hinted above, the key to the effective application of MGRIT lies in finding an adequate pair (or set) of fine and coarse integrators. In particular, the coarse integrator needs to do a "good enough" job of mimicking the action of the fine integrator, so that (5.35) represents a valid approximation to (5.34); at the same time, it should be cheap enough to compute, so that the solution to systems involving (5.35) can be promptly recovered. The necessity of overcoming this trade-off between cost and accuracy is common to other time-parallelisation algorithms (most notably, Parareal in Sec. 2.1.1), and a number of approaches have been proposed to address this. For MGRIT, the most straightforward is simple rediscretisation: the coarse integrator is none other than the fine integrator applied on a coarser time grid [46, 47]; an alternative consists in choosing integrators of varying levels of accuracy [109]. These solutions focus mostly on finding a coarse integrator which is cheap to compute, but it is unclear to what extent it matches the fine integrator. Indeed, in the hyperbolic framework, simple rediscretisation has been shown to fail in many cases (see Fig. 5.4), particularly if low-order time discretisations are employed: evidence of this is given in [33], as well as in the results of Fig. 5.6, at least for certain regimes. Therefore, when sticking to simple time-steppers, additional care is needed to ensure stability of MGRIT. To address this, we propose a novel approach which reverses the aforementioned principle: we start directly from the definition of the (explicit) fine integrator, and we progressively perform some simplifications which render it cheaper to apply,

without sacrificing "too much" accuracy. With this, we are able to "restore" MGRIT convergence for a low-order explicit time discretisation to be comparable to that of direct rediscretisation with high-order explicit time discretisations, although in neither case do we see CFL-robust convergence.

5.2.1.1 Fine integrator matching

To illustrate our approach, we consider our model equation (5.2). If a simple Lax-Friedrichs flux discretisation is applied in conjunction with a Forward Euler time discretisation, the resulting time-stepping formula reads

$$\begin{aligned} u_i^{n+1} &= \underbrace{\frac{u_{i+1}^n + u_{i-1}^n}{2}} \quad -\Delta t \quad \underbrace{\frac{f(u_{i+1}^n) - f(u_{i-1}^n)}{2\Delta x}} \\ &= \qquad Eu_i^n \qquad -\Delta t \qquad Df_i^n, \end{aligned} \tag{5.43}$$

with $f_i^n = f(u_i^n)$, and where we introduced the *central difference* operator, $D((*)_i) \coloneqq ((*)_{i+1} - (*)_{i-1})/2\Delta x$, and the *average* operator $E((*)_i) \coloneqq ((*)_{i+1} + (*)_{i-1})/2$. If we were to simply rediscretise at the coarse level and apply the same scheme to a coarser grid with time step $2\Delta t$, the resulting formula would similarly read

$$u_i^{n+1} = Eu_i^n - 2\Delta t Df_i^n. \tag{5.44}$$

However, we would like the result from the coarse integrator to be close to that of the fine integrator. The latter is given, in our case, by applying (5.43) twice (computing the nonlinear analogue of $\mathcal{F}^2$, as in (5.34) with $m = 2$), which results in

$$\begin{aligned} u_i^{n+2} &= E(Eu_i^n - \Delta t Df_i^n) - \Delta t Df(Eu_i^n - \Delta t Dfi^n) \\ &\approx E^2 u_i^n - \Delta t \left(EDf_i^n + Df(Eu_i^n)\right) \\ &+ \Delta t^2 D\left(f'(Eu_i^n) Df_i^n\right) \\ &- \Delta t^3 D\left(f''(Eu_i^n)(Df_i^n)^2\right). \end{aligned} \tag{5.45}$$

Here, we Taylor-expanded the flux in the second term around $E(u_i^n)$, exploiting the fact that Δt is a small parameter. In the case of (5.2), this formula is exact, since the flux f is a polynomial of degree 2. By directly comparing (5.45) and (5.44), we see that the formula for the coarse integrator is indeed much cheaper to compute, but it also significantly differs from that of the compounded fine integrator. A way to improve the accuracy of the coarse integrator without excessively increasing its cost consists in correcting (5.44) so that it matches (5.45) up to a certain order of Δt. This gives rise to the following integrators:

$$u_i^{n+1} = \left[\mathcal{G}^{(0)}(\boldsymbol{u}_i^n)\right]_i \coloneqq E^2 u_i^n - 2\Delta t Df_i^n, \tag{5.46}$$

if we aim for a zeroth-order match, leaving the rest untouched, or

$$u_i^{n+1} = \left[\mathcal{G}^{(1)}(\boldsymbol{u}_i^n)\right]_i := E^2 u_i^n - \Delta t \left(EDf_i^n + Df(Eu_i^n)\right), \tag{5.47}$$

for a first-order match. Formula (5.45) refers to a coarsening factor $m = 2$, but this can be easily extended to any value of $m > 1$, as follows:

$$u_i^{n+m} = E^m u_i^n - \Delta t \sum_{i=1}^{m} E^{i-1} Df(E^{m-i} u_i^n) + \mathcal{O}(\Delta t^2), \tag{5.48}$$

and the corresponding (5.46) and (5.47) can be modified accordingly.

Regrettably, this approach presents a number of downsides. First of all, it carries an increased computational cost with respect to rediscretisation, as the stencil of the operators involved grows linearly with m. For zeroth-order matching, this simply translates into a larger number of vector additions, so that the cost is still contained, provided we refrain from using aggressive coarsening strategies. For first-order matching, though, the number of flux evaluations per time-step increases as well, which unfortunately makes the method far less appealing. Secondly, this approach is limited in its application, as it requires an explicit formula for the repeated application of the fine solver (5.48): this needs to be both readily computable, and prone to simplifications without having to resort to evaluating intermediate states. All these requirements drastically limit the type of problems that can be addressed, as well as the possible schemes we can employ. In fact, we have to resort to a low-order, highly dissipative method: notice that (5.43) corresponds to the Lax-Friedrichs flux (5.17) with a choice of an even larger parameter $\alpha_{LF} = \Delta x/\Delta t$. For these reasons, we still recommend higher-order schemes, which in our tests behave reasonably well in most regimes. Nonetheless, the results in Sec. 5.3 show the validity of this method, and highlight the importance of accurately mimicking the action of the fine solver at the coarser levels to achieve fast convergence.

5.3 Numerical results

In this section, we discuss how the convergence behaviour of MGRIT is impacted by factors such as the parameters in the multigrid algorithm, and the type of integrators chosen. The code used for the simulations is publicly available at [28].

5.3.1 Fine integrator matching

As a first test, we check the effectiveness of the time-stepper proposed in Sec. 5.2.1.1 applied to Burgers' equation: the convergence results for a number of runs are reported in Fig. 5.4. We choose two different initial conditions: first, the simple sinusoidal wave $\bar{u}_S^0 = \sin(2\pi x/L)$, for which the solution develops a *stationary shock* positioned at $x_S = L/2$, at the *breaking time* $t_S = -1/\min_x(\bar{u}_S^0(x))' = L/(2\pi)$; second, its translation $\bar{u}_M^0 = (1+\sin(2\pi x/L))/2$, which instead develops a moving shock. These two choices allow us to investigate the impact of having to track a discontinuity which is not aligned with the grid.

First of all, we observe that the performance definitely improves if the coarse solver is chosen to match the fine solver up to a higher order: in fact, the blue curves in Fig. 5.4 (first-order matching, (5.47)) lie consistently below the green ones (zeroth-order matching, (5.46)). This is in line with expectations, since we are employing a more accurate solver. Still, the performance of zeroth-order matching remains competitive, achieving very effective reductions in error over the first iterations. Somewhat surprisingly, the zeroth-order matching gets slightly better initial error reductions in the moving-shock scenario than for the stationary shock. Unfortunately, though, this scheme clearly fails to effectively damp some components of the error, as we see from the fact that the corresponding error starts rising again after 5-10 iterations on finer meshes. This negative effect seems to be heightened as we refine the grid further, with the error starting to increase earlier. Nonetheless, in all the tests considered this approach behaves much better than naïve rediscretisation, for which the error blows up after the very first iteration and is hence not reported in the graph. Moreover, the convergence behaviour remains unchanged as we further refine our grids: the different lines superimpose almost perfectly. Modifying the type of multigrid cycle does not vary the nature of these observations, but as expected an F-cycle shows steeper convergence plots than a V-cycle. We have found that modifying the type of relaxation from an F- to an FCF-smoother, instead, does not improve convergence dramatically, so long as the initial guess for the solution at the coarse nodes is picked as in (5.40). An example of this is shown in Fig. 5.5, where a comparison of the error evolution of MGRIT is shown when using simple injection as a restriction operator instead of formula (5.40). The latter consistently shows better (or at worst comparable) results, particularly if MGRIT is equipped with simple multigrid components, such as V-cycle and F-relaxation as opposed to F-cycle and FCF-relaxation.

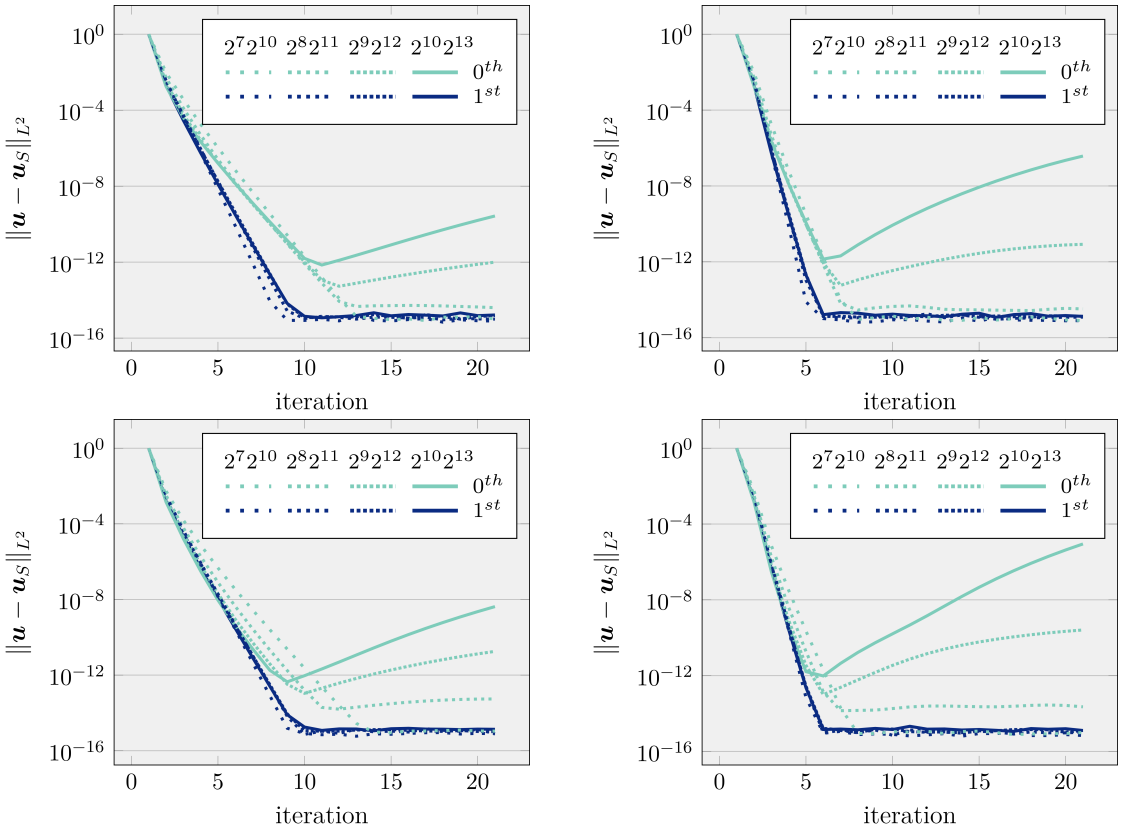


Figure 5.4: Impact of grid refinement and degree of fine-integrator matching on the evolution of the error between the MGRIT iterates and the serial solution $\boldsymbol{u}_S$. The relative L^2 norm in space-time is considered. MGRIT parameters: F-smoothing, 5 levels, coarsening factor $m = 2$, V-cycle (left), and F-cycle (right). Domain: $L = 1$, $T = 0.475$. More densely dotted graphs correspond to finer meshes: the numbers of nodes $N_x N_t$ at the finest level are reported in the legend. The time-steppers introduced in Sec. 5.2.1.1 are employed: FE with Lax-Friedrichs flux (5.43) for the finest level, and the corresponding zeroth- (5.46) and first-order matching (5.47), with varying coarsening factors, for the coarse levels (green and blue lines, respectively). Initial conditions: stationary shock (top, $u(x,0) = \bar{u}_S^0$), and moving shock (bottom, $u(x,0) = \bar{u}_M^0$). Results from pure rediscretisation are not reported, as the error diverges after the very first iteration.

The factor playing the main role in determining the performance of the algorithm, instead, lies in how closely the *Courant-Friedrichs-Lewy* CFL condition [89, Chap. 10.6] is met. In Fig. 5.4, the number of nodes N_t and N_x are chosen to guarantee that the CFL number on the *coarsest* temporal mesh is ~ 0.95 in all cases. The impact of relaxing the coarse-grid CFL condition can be clearly seen in the top-left graph of Fig. 5.6: convergence degrades noticeably, even using high-order matching. On the other hand, as we increase the number of levels in the multigrid scheme, this

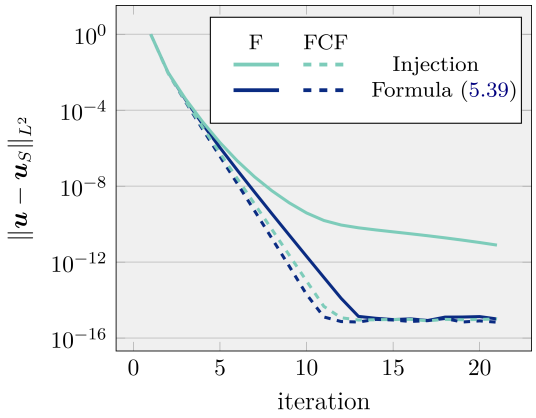


Figure 5.5: Effect of modifying the initial guess for the coarse solution on the error convergence of MGRIT: comparison of error convergence between classical injection (green) and formula (5.40) (blue). Relaxation used: F-relaxation (full lines) and FCF-relaxation (dashed lines); otherwise, same parameters as in the top-left graph in Fig. 5.4. Only results for $N_x N_t = 2^7 2^{10}$ (zeroth-order) are reported, but finer grids give comparable results.

condition becomes progressively more stringent, and translates into having to choose smaller and smaller Δt at the *finest* level, possibly over-resolving the solution.

5.3.2 High-order schemes

While the integrator proposed in Sec. 5.2.1.1 seems effective, it remains very dissipative. On the one hand, it is well-documented (see for example [35]) that smoothing effects improve convergence of MGRIT; on the other, though, artificial diffusion is very undesirable in the model problems considered, as it results in a heavy smearing of the shocks, which are the characteristic feature of the solutions of conservation laws. The choice of high-order space reconstructions is instead preferable, as they offer the possibility to preserve such discontinuities sharply, and an equally high-order time-discretisation is consequently requested. Investigating the behaviour of the MGRIT algorithm when used in conjunction with these schemes is hence of relevance, as it would more faithfully represent the setup of a real-world application. For this reason, we proceed to applying MGRIT to the test cases introduced in Sec. 5.1, for discretisations using the Lax-Friedrichs and Roe definitions of numerical fluxes, (5.17) and (5.26), and employing WENO reconstructions at various orders of accuracy s. The initial conditions for the various problems are chosen to ensure that the solutions

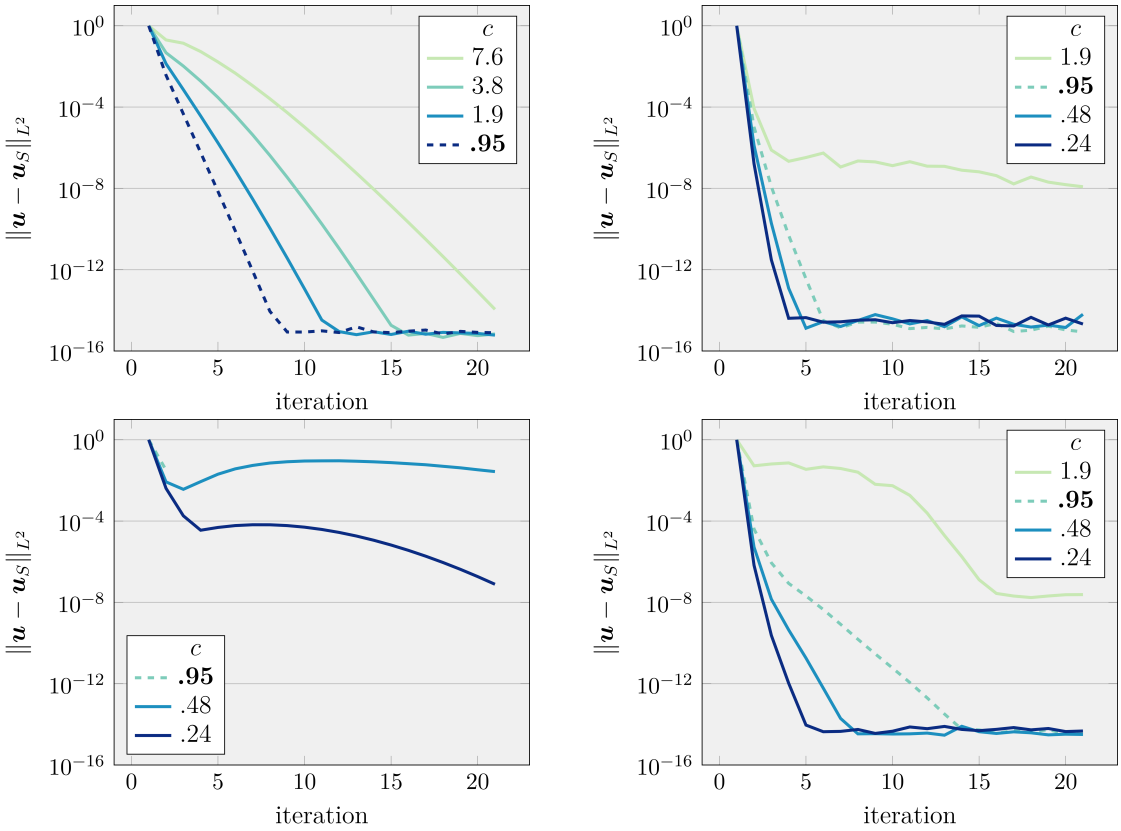


Figure 5.6: Influence of increasing the CFL number c at the coarsest level on the convergence behaviour of MGRIT applied to Burgers' equation. $N_x = 2^7$, while N_t is progressively halved to increase c. The other parameters are the same as in the top-left graph in Fig. 5.4, but different schemes are used in each plot. Top-left: first-order matching and FE; top-right: WENO reconstruction of order $s = 1$ and SSPRK3; bottom-left: WENO ($s = 5$) and FE; bottom-right: WENO ($s = 5$) and SSPRK3; the last three use a Roe flux and rediscretisation at all levels. Missing values are diverging (notice in particular the truncated $c = 0.95$ plot in the bottom-left graph).

achieve a similar maximum CFL number $c \sim 0.43$. These are:

$$\bar{u}_B^0 = \frac{4}{3}\sin\left(\frac{2\pi x}{L}\right), \quad \bar{\boldsymbol{u}}_S^0 = \frac{1}{11}\begin{bmatrix} 1 + \frac{1}{2}\sin\left(\frac{2\pi x}{L}\right) \\ 0 \end{bmatrix}, \quad \text{and} \quad \bar{\boldsymbol{u}}_E^0 = \begin{bmatrix} 1 \\ 0 \\ 1 + \frac{1}{2}\sin\left(\frac{2\pi x}{L}\right) \end{bmatrix}, \tag{5.49}$$

for Burgers', shallow-water, and Euler equations, respectively. From Fig. 5.7, we see that the error behaviour varies in quite an erratic way as we modify s. For both Lax-Friedrichs and Roe fluxes with the SSPRK3 time-stepper, the best performance is seen with an integrator which does not perform any sort of reconstruction, *i.e.*, $s = 1$. This is in line with expectations, as this choice of s corresponds to the most dissipative among the schemes considered. The tendency is for performance to

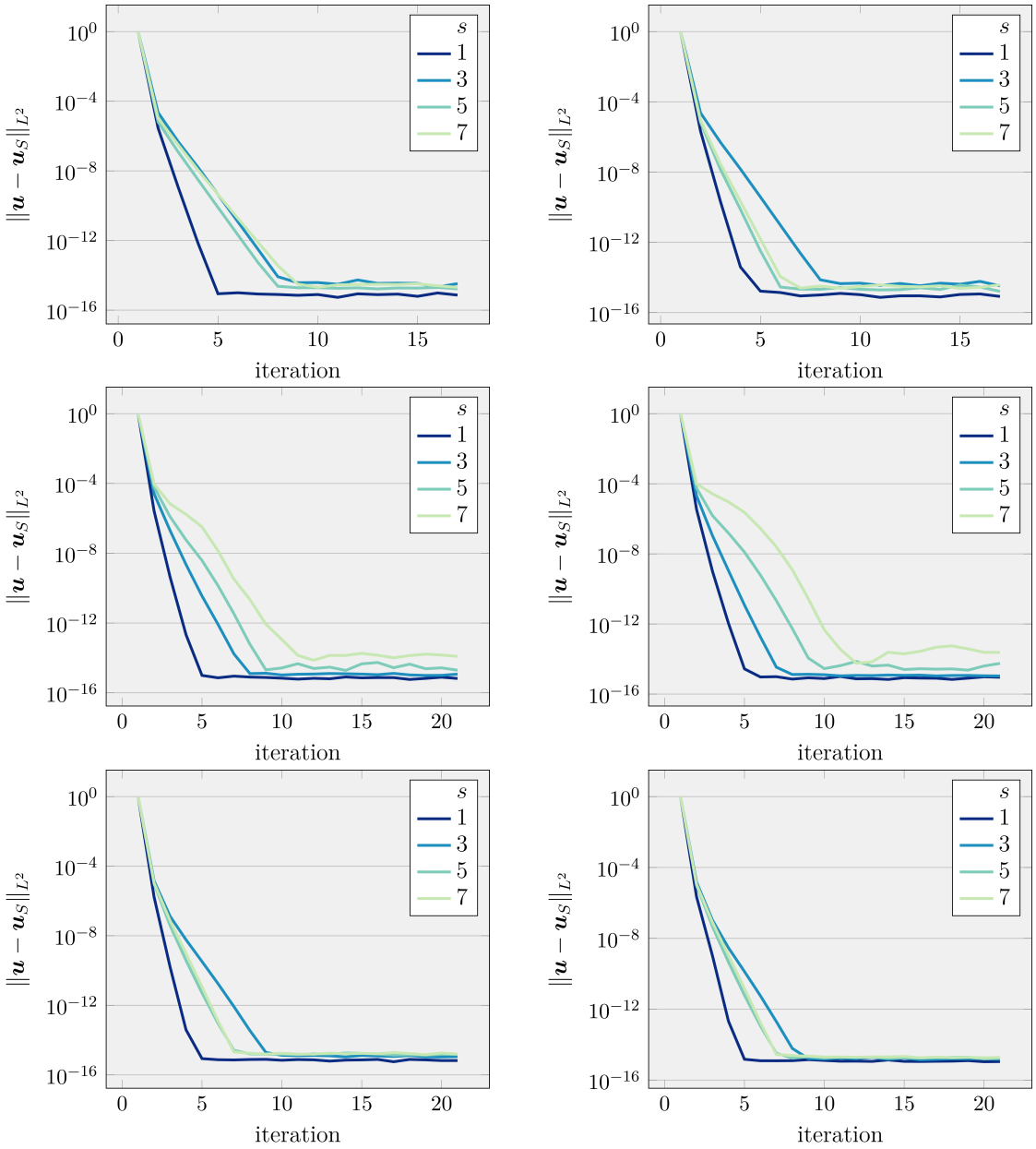


Figure 5.7: Evolution of the error in the MGRIT iterates, applied to Burgers' (top), shallow-water (middle) and Euler equations (bottom). MGRIT parameters: V-cycle, F-smoothing, 3 levels with $N_t = 100, 200, 400$, respectively, $N_x = 64$. Domain: $L = 1$, $T = 0.5$. Flux discretisation used: Lax-Friedrichs (left plots, (5.17)), and Roe (right plots, (5.26)), both employing WENO reconstruction of different orders of accuracy s. Time-stepper used: SSPRK3 (5.28) with rediscretisation at all levels.

worsen as we increase s, although $s = 3$ seems to provide an exception and behave particularly badly in some cases: the reason behind this remains unclear. Nonetheless, these higher-order methods still behave reasonably well, particularly in comparison to many results in the literature regarding more diffusive schemes. Evidence of this

is also given in Fig. 5.6: at least for $c \leq .95$, which corresponds to the dashed lines in the plots, SSPRK3 (top-right) outperforms FE (top-left), even if the latter is used in conjunction with the first order-matching procedure (5.47). Indeed it is accuracy in the *time* discretisation that helps convergence, while increasing the order of the *space* discretisation seems to produce the opposite effect, as already discussed before, and further testified by the poorer error behaviour in the bottom plots of Fig. 5.6. In particular, note how MGRIT (with rediscretisation on the coarse grid) fails to produce a converging solver for $c = .95$ (or even $c = 0.48$) in the bottom-left of Fig. 5.6, where it is used in combination with a high-order spatial discretisation and a low-order temporal discretisation.

Ultimately, however, even in this case the CFL condition has shown to be the true bottleneck in the effectiveness of such solvers. Indeed we can see from the plots in Fig. 5.6 that convergence of MGRIT using high-order schemes is much more sensitive to an increase in the CFL number c.

5.3.3 Changing accuracy per coarsening level

Experimenting with discretisations of different accuracy at different multigrid levels makes sense in the attempt to match the action of the fine solver: one can think of making up for the loss of precision due to coarsening by increasing the order of the reconstruction at a coarser level, and still obtain a result similar to what the fine integrator would provide. This might provide a feasible coarse solver, provided the additional computational cost associated with reconstruction is reasonably small. With this motivation in mind, we test applications of MGRIT where the order of the discretisations used, (both in space and time), varies across the levels. Unfortunately, the results reported in Fig. 5.8 pinpoint that this strategy is not beneficial to the algorithm performance and that, instead, simple rediscretisation is preferable. Among the alternatives considered, though, decreasing the accuracy as we descend to coarser levels has proven to be the most promising, which could make it attractive if one seeks to make the application of coarse integrators cheaper: this is also in line with the results shown in [109], where a lower-order WENO reconstruction was used at the coarse level.

5.4 Remarks on MGRIT for hyperbolic equations

From the experiments conducted in this chapter, it appears clear that the necessity of satisfying the CFL condition at each coarsening level represents a major hindrance

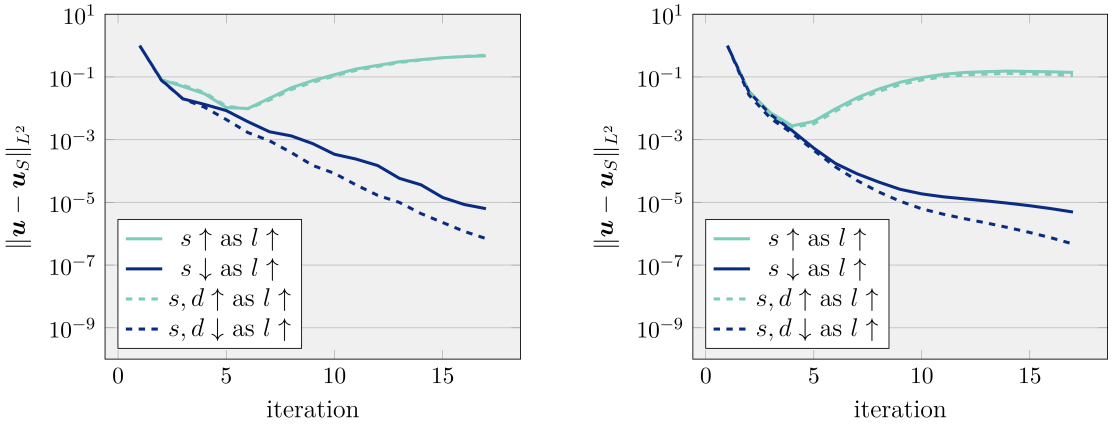


Figure 5.8: Impact of modifying the accuracy of the schemes used across the levels on the error convergence of MGRIT. Same parameters as in Fig. 5.7, but with the order of the WENO reconstruction s either increasing (green) or decreasing (blue) as we move to coarser levels. Dashed lines show error convergence if both s and the order accuracy of the time-stepper, d, are changed simultaneously, while solid lines show results if only s is varied. Left is for Lax-Friedrichs flux, right is for Roe flux. Only results for the application to Burgers' equation are shown, but the ones collected for the other test-cases follow the same qualitative behaviour.

towards the stability of MGRIT, when applied to accelerate the solution of conservation laws discretised with explicit time-stepping schemes. This seems to indicate that, for a multigrid-in-time algorithm to be effective in this framework, its coarsening strategy needs to act on both the spatial and temporal domain, so that the CFL number is kept close to 1 at each level. This complicates the analysis even further, given how most of the currently available theoretical results on the convergence of MGRIT rely heavily on the algorithm implementing a purely temporal coarsening procedure. Nonetheless, some research in this sense has already been conducted [78, 138], and is providing further evidence of the effectiveness of employing a combined *space*-time multigrid method. Indeed the AIR algorithm introduced in Sec. 2.2.4 makes use of such strategies, on top of employing restriction techniques specific to advection problems [138].

In hindsight, AIR could have been a valuable method for the focus of our analysis in tackling time parallelisation of hyperbolic equations, but at the time the research reported in this and the following chapters was conducted, this scheme was still at its preliminary stages. For this reason, instead, we decided to work on the study of algorithms that were more mature, such as MGRIT in this chapter, or that had already started to prove their applicability in the hyperbolic case. The method investigated in the following chapter represents an alternative to MGRIT in this sense.

Chapter 6

Space-time circulant preconditioning for linear wave equations

The use of circulant preconditioners to accelerate the solution of Krylov subspace iterative methods for Toeplitz systems is a well-established practice, dating back to the work of Strang in 1986 [145], and employed in a range of frameworks ever since (see *e.g.* [108] for details on the iterative solution of Toeplitz systems, as well as a number of their applications). In more recent years, this practice has been investigated for accelerating the all-at-once solution of monolithic systems arising from the simultaneous space-time discretisation of time-evolving PDEs [3,64,65,87,93,101].

When dealing with constant-coefficient, linear PDEs discretised on a uniform temporal grid, in fact, the resulting space-time system presents a block Toeplitz structure, whose blocks are constant down each diagonal, making it amenable to circulant preconditioning. Systems involving circulant matrices are relatively simple to invert by means of the Fourier transform, which effectively diagonalise the operator, thus allowing for parallelisation in its solution. This is what makes circulant preconditioning for space-time matrices an actual parallel-in-time method. An appealing property of this strategy is that, when applied to hyperbolic PDEs, it does not seem to suffer from the same stability issues afflicting other PinT methods (see Sec. 2.1.1.1, as well as the analysis in Chap. 5), thus making it a useful alternative to achieve time parallelisation in this framework.

In this chapter, we show how a theoretical result from Chan, Potts, and Steidl [16] on the convergence of circulant-preconditioned GMRES applies to the solution of space-time systems. Under certain assumptions, this ensures that the iterative method will converge (terminate) in a number of steps which is independent of the

number of nodes in the temporal mesh, thus giving an indication of the effectiveness of the space-time circulant preconditioning strategy. We provide extensive experimental results that measure its performance when applied to linear wave equations, using high-order discretisations tailored for this type of problems, which confirm the theoretical results and provide further evidence for the applicability of space-time circulant preconditioning for hyperbolic PDEs.

In Sec. 6.1, we expand on the introduction to circulant preconditioners provided in Sec. 2.2.2, and describe the main theoretical convergence result exploited in this chapter. We proceed to test the effectiveness of the space-time circulant preconditioning strategy, first applying it to a toy ODE problem in Sec. 6.2, and then to a more interesting target PDE in Sec. 6.3, providing the relevant experimental results in the corresponding sections.

The analysis and results from this chapter have been published in the *Numerical Linear Algebra with Applications* journal, and are available at [32].

6.1 Circulant preconditioning for Toeplitz systems

In the literature, a variety of circulant preconditioners for Toeplitz matrices have been proposed: some examples of these are listed in [17, Chap. 2], and indeed some of these approaches coincide for the simple test cases considered here. In this chapter we exploit some of the results from R. Chan, D. Potts, and G. Steidl [16]: therefore, we follow most of the notation used in their paper, and describe the ω-circulant preconditioner therein defined.

6.1.1 ω-circulant preconditioner

Let us introduce a 2π-periodic *generating function* $f(x)$, with the associated Toeplitz matrix $A_N(f) \in \mathbb{R}^{N\times N}$, that is:

$$[A_N(f)]_{m,n} \coloneqq a_{m-n}(f), \qquad \text{with} \qquad a_k(f) \coloneqq \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-\mathrm{i}kx}\,dx, \tag{6.1}$$

where the $a_k(f)$ are the *Fourier coefficients* of $f(x)$. In our work we are mostly concerned with *banded* Toeplitz matrices with a small band, which have *trigonometric polynomials* as generating functions, in the form

$$f(x) = \sum_{k=-s_2}^{s_1} a_k e^{\mathrm{i}kx}, \qquad \text{for some} \quad a_k \in \mathbb{R}, \quad s_1, s_2 \in \mathbb{N}, \quad 0 < s_1 + s_2 \ll N. \tag{6.2}$$

In this case, it can be shown that the Fourier coefficients (6.1) coincide with the coefficients of the polynomial (6.2); moreover, the summation extrema s_1 and s_2 in (6.2) directly denote the maximum number of non-zero sub- and super-diagonals of $A_N(f)$, respectively, so that this matrix has the following structure:

$$A_N(f) = \begin{bmatrix} a_0 & a_{-1} & \cdots & a_{-s_2} & & \\ a_1 & \ddots & \ddots & & \ddots & \\ \vdots & \ddots & & & & a_{-s_2} \\ a_{s_1} & & & & \ddots & \vdots \\ & \ddots & & \ddots & \ddots & a_{-1} \\ & & a_{s_1} & \cdots & a_1 & a_0 \end{bmatrix}. \tag{6.3}$$

We further require that $f(x)$ has no zeroes on the N equispaced nodes:

$$f\left(x_{N,l}\right) \neq 0 \qquad \forall x_{N,l} = w + \frac{2\pi l}{N}, \quad \text{with} \quad l = 0, \ldots, N-1, \tag{6.4}$$

where $w \in [0, 2\pi/N)$ is a free parameter which gives some flexibility in ensuring that we steer away from zeroes of the generating function: the ω in the name of the original preconditioner [16] is given by $\omega = e^{\mathrm{i}Nw}$. To simplify our notation, in this section we limit ourselves to the choice $w = 0$ (that is, $\omega = 1$), but the treatment would remain largely unchanged otherwise. We also introduce the diagonal matrix $D_N(f) \in \mathbb{R}^{N \times N}$, which contains evaluations of f on the nodes $x_{N,l}$ defined in (6.4):

$$D_N(f) \coloneqq \operatorname*{diag}_{l=0,\ldots,N-1} \left(f\left(x_{N,l}\right)\right), \tag{6.5}$$

as well as the *discrete Fourier transform* operator, already seen in (2.28):

$$\left[F_N\right]_{m,n} \coloneqq \frac{1}{\sqrt{N}} e^{-\frac{2\pi \mathrm{i}}{N} mn}, \qquad m, n = 0, \ldots, N-1. \tag{6.6}$$

The ω-circulant matrix $M_N(f)$ associated with $A_N(f)$ is then defined as

$$M_N(f) \coloneqq F_N \, D_N(f) \, F_N^*. \tag{6.7}$$

In our case, the elements of $M_N(f)$ can be explicitly characterised in a relatively straightforward manner:

$$\left[M_N(f)\right]_{m,n} = \begin{cases} a_{m-n} & \text{if} \quad m = n \\ a_{m-n} + a_{(m-n)-N} & \text{if} \quad m > n \\ a_{m-n} + a_{(m-n)+N} & \text{if} \quad m < n \end{cases}$$

$$\implies \quad M_N(f) = \begin{bmatrix} a_0 & (a_{-1} + a_{N-1}) & (a_{-2} + a_{N-2}) & \cdots \\ \left(a_1 + a_{-(N-1)}\right) & \ddots & \ddots & \ddots \\ \left(a_2 + a_{-(N-2)}\right) & \ddots & & \\ \vdots & \ddots & & \end{bmatrix}; \tag{6.8}$$

that is, in order to recover $M_N(f)$, the off-diagonal elements of $A_N(f)$ are summed to those N diagonals away. Notice that in our particular setting, where the band of $A_N(f)$ is small (specifically, smaller than $N/2$), and $\omega = 1$, the ω-preconditioner coincides with the *Strang circulant* preconditioner [145].

Formula (6.7) implies that inverting a circulant matrix amounts to applying a *Fast Fourier Transform* (FFT), inverting a diagonal matrix, and applying the inverse FFT: a task of overall complexity $\mathcal{O}(N \log N)$. This argument can be similarly extended to block circulant matrices, in which case we would achieve block diagonalisation. Therein also lies the concurrency of the algorithm: solving a (block) diagonal system is a task which can be trivially parallelised. Being relatively simple to invert, (6.7) consequently satisfies the first requirement of a preconditioner. As a second requirement, we would also hope for it to effectively accelerate the convergence of iterative methods. We address this in the following.

6.1.2 Convergence of circulant-preconditioned Krylov methods

There is a choice of iterative methods one can use, paired with the preconditioner (6.7) (or slight modifications thereof), to solve a system of equations involving (6.3). For example, following [101], one could symmetrise the Toeplitz system via a Hankel matrix, and use MINRES with the absolute value of (6.7) as preconditioner. An alternative explored in [16] consists in solving the normal equation using *Conjugate Gradient*, preconditioned with $M_N(|f|^2)$. In both cases, strong theoretical bounds guarantee fast convergence of the methods used. We are mostly concerned with convergence of circulant preconditioners applied to GMRES. While convergence results are harder to come by in this case [68], some experiments [3, 64] hint at it being a more effective iterative solver with respect to other choices available.

In this section, we use a result from Chen, Potts and Steidl [16], which deals with the application of (6.7) as a right-preconditioner for GMRES iterations for a system involving (6.3). Adapted to our case, the theorem states the following:

Theorem 6.1.1. *Let f be a trigonometric polynomial such as (6.2), satisfying (6.4), and generating the Toeplitz matrix $A_N(f)$, with associated ω-circulant preconditioner $M_N(f)$. Then,*

$$A_N(f)M_N(f)^{-1} = I_N + R_N(s_1 + s_2), \tag{6.9}$$

where $I_N \in \mathbb{R}^{N\times N}$ is the identity matrix, while $R_N(s) \in \mathbb{R}^{N\times N}$ identifies a matrix of rank s.

Proof. In our situation, the proof simplifies significantly. We can split $A_N(f)$ as $M_N(f) - B_N(f)$, where

$$B_N(f) = \begin{bmatrix} & & a_{s_1} & a_{s_1-1} & \dots & a_1 \\ & & & \ddots & \ddots & \vdots \\ & & & & \ddots & a_{s_1-1} \\ a_{-s_2} & & & & & a_{s_1} \\ a_{-s_2+1} & \ddots & & & & \\ \vdots & \ddots & \ddots & & & \\ a_{-1} & \dots & a_{-s_2+1} & a_{-s_2} & & \end{bmatrix}, \tag{6.10}$$

so that

$$A_N(f)M_N(f)^{-1} = I_N(f) - B_N(f)M_N(f)^{-1}, \tag{6.11}$$

from which it follows that $B_N(f)M_N(f)^{-1} = R_N(s_1 + s_2)$, since $B_N(f)$ only has non-zero elements in the top s_1 and bottom s_2 rows. □

Remark 3. Notice that the theorem allows for some flexibility in breaking the Toeplitz structure of $A_N(f)$. If we consider a perturbation $\tilde{A}_N(f)$ to $A_N(f)$, we can still decompose $A_N(f) + \tilde{A}_N(f) = M_N(f) - \tilde{B}_N(f)$, where $\tilde{B}_N(f) = B_N(f) - \tilde{A}_N(f)$. Provided that the entries in $\tilde{A}_N(f)$ which do not adhere to the Toeplitz structure are only limited to the first s_1 and last s_2 rows, then, Thm. 6.1.1 still holds. This is particularly useful if we consider that we are dealing with the solution of discrete space-time systems. In this framework, in fact, we are often forced to modify the equations corresponding to the first few temporal nodes, be it in order to accommodate for initial conditions on the derivative, and/or in order to kick-start high-order discretisations in time: this has the effect of perturbing the Toeplitz structure of the system, but not to an extent that would break the result of Thm. 6.1.1, as described more in detail in Sec. 6.2 and 6.3.

As a note, the original theorem is presented in [16] in a more general form, and deals rather with *rational* generating functions:

$$f(x) = \frac{p(x)}{q(x)} = \frac{\sum_{k=0}^{d_1} p_k e^{\mathrm{i}kx}}{\sum_{k=0}^{d_2} q_k e^{\mathrm{i}kx}}, \qquad \text{with} \quad d_1, d_2 \in \mathbb{N}, \quad d_1 d_2 \neq 0. \tag{6.12}$$

In that case, the original theorem states that the rank of the resulting perturbation of the identity is equal to $\max\{d_1, d_2\}$. It is possible to recover our version of Thm. 6.1.1 by noticing that we can rewrite (6.2) as $f(x) = \left(\sum_{k=0}^{s_1+s_2} a_{-s_2+k} e^{\mathrm{i}kx}\right) / e^{\mathrm{i}s_2 x}$.

Noticeably, Thm. 6.1.1 gives us a theoretical guarantee that GMRES with $M_N(f)$ as a right-preconditioner will converge to the solution after at most $s_1 + s_2 + 1$ iterations, at least in exact precision: if the band of $A_N(f)$ is small enough, then (6.7) provides an excellent preconditioner.

In the following sections, we ilustrate how simple systems arising from the discretisation of differential equations are (close-to-) Toeplitz in nature, which makes them amenable to be preconditioned using (6.7). The goal is to provide examples of the effectiveness of this preconditioner, as well as to discuss cases in which this theorem fails to come to our aid. To this purpose, we introduce a few model problems, and the methods used in our experiments to recover their numerical solution.

6.2 All-at-once solution of linear ODEs

As a proof of concept, useful in introducing some of the notation employed, we propose the simplest linear second-order differential equation:

$$\begin{cases} \dfrac{d^2u}{dt^2} - \lambda u = 0 & t \in (0, T] \\ u(0) = \bar{u}^0 \\ \dfrac{du}{dt}(0) = \bar{u}^1 \end{cases}, \tag{6.13}$$

where $\bar{u}^0$ and $\bar{u}^1 \in \mathbb{R}$ represent the initial values of the solution itself and its derivative, respectively.

6.2.1 Discretisation aspects

To discretise our temporal domain, we choose a uniform grid with step size $\Delta t = T/(N_t - 1)$. Furthermore, to approximate (6.13) we consider *Störmer-Verlet* schemes (SV) of various orders: this is a family of multi-step methods specifically tailored for the solution of second-order differential equations [70, Chap. III-10]. The general formulation for a k-th order SV scheme is given by

$$u^n - 2u^{n-1} + u^{n-2} = \Delta t^2 \sum_{j=0}^{k-1} \alpha_j D_-^j f^n, \tag{6.14}$$

where the right-hand side of the system f^n simply corresponds to λu^n in our case; D_- identifies the backward difference operator

$$D_-^j f^n = D_-^{j-1} f^n - D_-^{j-1} f^{n-1}, \qquad \text{with} \qquad D_-^0 f^n = f^n, \tag{6.15}$$

while the coefficients α_j are picked in such a way to ensure the desired level of accuracy [70, Table.10.2]: their values for the schemes used in this chapter are

$$[\alpha_j]_{j=0}^8 = \left[1, -1, \frac{1}{12}, 0, -\frac{1}{240}, -\frac{1}{240}, -\frac{221}{60480}, -\frac{19}{60480}, -\frac{9829}{3628800}\right]. \tag{6.16}$$

Notice that for all cases but $k = 2$, (6.14) defines an implicit method. For ease of notation, we identify discretisation schemes of different accuracies by reporting their order as a subscript to their acronym. We can see that choosing SV_1 corresponds to approximating the second-order derivative using a first-order backward difference formula, which produces the following recurrence relation:

$$\begin{cases} \dfrac{u^n - 2u^{n-1} + u^{n-2}}{\Delta t^2} - \lambda u^n = 0 & n = 2, \dots, N_t - 1 \\ \dfrac{2u^1 - 2u^0}{\Delta t^2} - \lambda u^1 = \dfrac{2}{\Delta t}\bar{u}^1 & \\ u^0 = \bar{u}^0 & \end{cases} . \tag{6.17}$$

Notice that we use a *ghost node* approach to include the initial condition on the derivative at $t = 0$; that is, we introduce a fictitious node u^{-1}, approximate the first derivative using the *central difference* (CD) formula $u^1 - u^{-1} = 2\Delta t\, \bar{u}^1$, and substitute in the first equation of (6.17) with $n = 1$ to recover the second equation. We can gather the equations (6.17) together and build the system

$$\underbrace{\left(\Phi^{\mathrm{SV}_1} - \lambda \Delta t^2 \Phi_r^{\mathrm{SV}_1}\right)}_{=: A^{\mathrm{SV}_1}} \boldsymbol{u} = \boldsymbol{g}^{\mathrm{SV}_1}, \tag{6.18}$$

where $\boldsymbol{u} = \left[u^0, u^1, \dots, u^{N_t - 1}\right]^T \in \mathbb{R}_t^N$ collects the values of the solution at each temporal node. The operator $\Phi^{\mathrm{SV}_1} \in \mathbb{R}^{N_t \times N_t}$ represents the discretisation of the time derivative (including the perturbation from the ghost node approach), and has the following structure:

$$\Phi^{\mathrm{SV}_1} \coloneqq \begin{bmatrix} 1 & & & & \\ -2 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{bmatrix} \in \mathbb{R}^{N_t \times N_t}. \tag{6.19}$$

Operator $\Phi_r^{\mathrm{SV}_1}$ stems from the evaluation of the right-hand side in (6.14), and it reduces to $\Phi_r^{\mathrm{SV}_1} = I^{N_t} \in \mathbb{R}^{N_t \times N_t}$, the identity matrix, for SV_1. The effect of the initial conditions is included in the right-hand side:

$$\boldsymbol{g}^{\mathrm{SV}_1} = \left[(1 - \lambda \Delta t^2)\, \bar{u}^0, 2\Delta t\, \bar{u}^1, 0, \dots, 0\right]^T \in \mathbb{R}^{N_t}. \tag{6.20}$$

Notice how imposing the initial conditions breaks the Toeplitz structure of the matrix (6.19), and hence that of the system A^{SV_1}: the second element in the second

row is different from the others on the diagonal. However, we can still assemble the circulant preconditioner disregarding this perturbation, as such:

$$M^{\mathrm{SV}_1} := \hat{\Phi}^{\mathrm{SV}_1} - \lambda \Delta t^2 \Phi_r^{\mathrm{SV}_1}, \tag{6.21}$$

where $\hat{\Phi}^{\mathrm{SV}_1}$ is our circulant approximation to Φ^{SV_1}, built as follows:

$$\hat{\Phi}^{\mathrm{SV}_1} := \begin{bmatrix} 1 & & & 1 & -2 \\ -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{bmatrix} \in \mathbb{R}^{N_t \times N_t}. \tag{6.22}$$

As hinted in Rmk. 3, the statement of Thm. 6.1.1 remains valid using M^{SV_1} as a preconditioner for A^{SV_1}, even if the system is not strictly Toeplitz: in fact, we only perturbed the second row on a matrix with a band of size 2.

We also experiment using higher-order discretisations. For orders > 2, this requires recovering the values of the solution at the first few time steps in some alternative way, in order to *kick-start* the multi-step method [90, Chap. 5.9.3]. This is due both to the stencil generally becoming larger as the order increases, and to the fact that the ghost-node approach used in (6.17) is only second-order accurate, and will end up polluting the global accuracy of higher-order schemes. There are a number of ways in which the kick-starting procedure might be conducted, (generally resorting to self-starting schemes to solve for the first few unknowns), which end up disrupting the Toeplitz structure of the space-time system. For simplicity, in our experiments we directly evaluate the analytical solution for the necessary nodes; however, in order to mimic the impact the perturbation stemming from a kick-starting procedure might have on the performance of the circulant preconditioner, we also modify the first few equations in the system by leaving only the contribution from the main diagonal. For example, using SV_5 gives rise to the space-time system:

$$\underbrace{\left(\Phi^{\mathrm{SV}_5} - \lambda \Delta t^2 \Phi_r^{\mathrm{SV}_5}\right)}_{=:A^{\mathrm{SV}_5}} \boldsymbol{u} = \boldsymbol{g}^{\mathrm{SV}_5}, \tag{6.23}$$

where Φ^{SV_5} and $\Phi_r^{\mathrm{SV}_5}$ play the same roles as Φ^{SV_1} and $\Phi_r^{\mathrm{SV}_1}$ in (6.19), but both have

their first 4 rows perturbed, in the following manner:

$$\Phi^{\mathrm{SV}_5} \coloneqq \begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ 0 & 0 & 1 & & & & & \\ 0 & 0 & 0 & 1 & & & & \\ 0 & 0 & 1 & -2 & 1 & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & 1 & -2 & 1 & \end{bmatrix} \in \mathbb{R}^{N_t \times N_t}, \qquad \text{and}$$

$$\Phi_r^{\mathrm{SV}_5} \coloneqq \frac{1}{240} \begin{bmatrix} 19 & & & & & & & \\ 0 & 19 & & & & & & \\ 0 & 0 & 19 & & & & & \\ 0 & 0 & 0 & 19 & & & & \\ -1 & 4 & 14 & 204 & 19 & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & -1 & 4 & 14 & 204 & 19 & \end{bmatrix} \in \mathbb{R}^{N_t \times N_t}. \tag{6.24}$$

The circulant preconditioner is then assembled as in (6.21), starting from the circulant approximations to the operators in (6.24), that is

$$M^{\mathrm{SV}_5} \coloneqq \hat{\Phi}^{\mathrm{SV}_5} - \lambda \Delta t^2 \Phi_r^{\mathrm{SV}_5}, \tag{6.25}$$

where $\hat{\Phi}^{\mathrm{SV}_5}$ is the same as $\hat{\Phi}^{\mathrm{SV}_1}$, while $\hat{\Phi}_r^{\mathrm{SV}_5}$ is built disregarding the perturbations from the underlying Toeplitz structure of $\hat{\Phi}_r^{\mathrm{SV}_5}$. Finally, the right-hand side of the space-time system is built as

$$\left[\boldsymbol{g}^{\mathrm{SV}_5}\right]_j = \begin{cases} \left[A^{\mathrm{SV}_5}\right]_{j,j} u^j & \text{if} \quad j \leq 3 \\ 0 & \text{if} \quad j > 3 \end{cases}, \tag{6.26}$$

assuming the values at the first few instants $u^j = u(j\Delta t)$, (with $j = 0, \dots, 3$), are given. Notice that Rmk. 3 still applies, and that the result of Thm. 6.1.1 remains valid even in this case, since we perturbed the first 4 rows on a matrix where $s_1 = 4$ and $s_2 = 0$. Similar considerations hold for the systems corresponding to higher-order discretisations, which are built following an analogous procedure.

6.2.2 Results

In Tab. 6.1 we collect the convergence results from the application of (6.7) as a right-preconditioner for GMRES, used to recover the solution of the systems (6.18), (6.23), and others built using SV discretisations of various orders. The theoretical bound provided in Thm. 6.1.1 is respected reasonably well: the discrepancies are attributable to round-off errors, and seem more pronounced for higher-order schemes and more

Table 6.1: Number of iterations to convergence for GMRES, right-preconditioned with (6.7), applied to the solution of systems arising from the discretisation of (6.13). Systems of different size N_t, and SV schemes of various orders of accuracy are considered. The top row reports the theoretical upper bound from Thm. 6.1.1. The two columns of values refer to different tolerances for convergence: $\varepsilon = 10^{-7}$ (left, in brackets), $\varepsilon = 10^{-10}$ (right). The initial guess for GMRES is randomly chosen (from a normal distribution). The problem parameters are: $\bar{u}^0 = 1$, $\bar{u}^1 = -1$, $\lambda = -1$, $T = 10^3$.

	SV_1	SV_2	$\text{SV}_3 (= \text{SV}_4)$	SV_5	SV_6	SV_7	SV_8
$s_1 + s_2 + 1$	3	3	3	5	6	7	8
$N_t = 10^2 \cdot 2^3$	(3) 3	(3) 3	(3) 3	(5) 5	(6) 6	(7) 7	(8) 8
$N_t = 10^2 \cdot 2^4$	(3) 3	(3) 3	(3) 3	(5) 5	(6) 6	(7) 7	(8) 8
$N_t = 10^2 \cdot 2^5$	(3) 3	(3) 3	(3) 3	(5) 5	(6) 6	(7) 7	(8) 8
$N_t = 10^2 \cdot 2^6$	(3) 3	(3) 3	(3) 3	(5) 5	(6) 6	(7) 7	(8) 8
$N_t = 10^2 \cdot 2^7$	(3) 3	(3) 3	(3) 3	(5) 5	(6) 6	(7) 8	(8) 9
$N_t = 10^2 \cdot 2^8$	(3) 3	(3) 3	(3) 4	(5) 6	(6) 7	(7) 8	(8) 9
$N_t = 10^2 \cdot 2^9$	(3) 3	(3) 4	(3) 4	(5) 6	(6) 7	(7) 8	(8) 9
$N_t = 10^2 \cdot 2^{10}$	(3) 3	(3) 4	(3) 4	(5) 6	(6) 7	(7) 8	(9) 9
$N_t = 10^2 \cdot 2^{11}$	(3) 4	(3) 4	(3) 4	(6) 6	(7) 7	(7) 8	(9) 9

refined grids. The ω-preconditioner is further tuned by picking the parameter w in (6.4) as $w = \pi/N_t$, which (at least for the lowest-order schemes) ensures that the roots of the corresponding generating function are the farthest away from 0; this in turns attains the desirable property of minimising the condition number of the preconditioner. A much more detailed analysis of the impact of this parameter on the effectiveness of the preconditioner, however, has been presented in [93].

6.3 All-at-once solution of linear PDEs

Ultimately, our interest resides in exploiting a (block) circulant preconditioner for the solution of hyperbolic partial differential equations. In presenting the test-case for this class of PDEs, we mostly follow the set-up provided in [65], and consider the *wave equation* with Dirichlet boundary conditions:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} - \dfrac{\partial^2 u}{\partial x^2} = 0 & (x,t) \in (0,L) \times (0,T] \\ u(x,0) = \bar{u}^0(x) & x \in [0,L] \\ \dfrac{\partial u}{\partial t}(x,0) = \bar{u}^1(x) & \\ u(0,t) = u(L,t) = 0 & t \in (0,T] \end{cases} . \tag{6.27}$$

6.3.1 Discretisation aspects

Most of the notation used in Sec. 6.2 can be extended to the PDE case, but we need to introduce an appropriate discretisation of the spatial operator. For this purpose, we choose once again a uniform grid spanning the interior of the spatial domain, with spacing $\Delta x = L/(N_x + 1)$, and we consider a *central difference* scheme (CD) for the second-order spatial derivative. Also for spatial discretisation we make use of approximations of different orders of accuracy, to match the temporal discretisations. Furthermore, the Dirichlet boundary conditions in (6.27) are imposed naturally. This causes some inconvenience when using high order schemes: in that case, in fact, we cannot use symmetric stencils anymore as we approach the boundaries, since we would require information from nodes outside the domain. We circumvent this issue by using, for those few nodes, finite difference of the same order of accuracy, but defined on a slanted stencil [52]. For example, for a second-order accurate discretisation, we obtain the following matrix:

$$\mathcal{K}^{\mathrm{CD}_2} \coloneqq \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2 & 1 & \\ & & & 1 & -2 \end{bmatrix} \in \mathbb{R}^{N_x \times N_x}; \tag{6.28}$$

while for a fourth-order one, we get

$$\mathcal{K}^{\mathrm{CD}_4} \coloneqq \frac{1}{12} \begin{bmatrix} -6 & 14 & -4 & -15 & 10 & & \\ 16 & -30 & 16 & 1 & & & \\ -1 & 16 & -30 & 16 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 16 & -30 & 16 & -1 \\ & & & -1 & 16 & -30 & 16 \\ & & 10 & -15 & -4 & 14 & -6 \end{bmatrix} \in \mathbb{R}^{N_x \times N_x}. \tag{6.29}$$

We point out that the scheme stemming from choosing SV_2 for the temporal discretisation and CD_2 for the spatial one is also known as the *explicit Leapfrog* method [90, Chap. 10.2.2], which is *conditionally stable*. This can be shown via Von-Neumann analysis [90, Chap. 9.6]: assuming the solution is composed of a single Fourier mode $u^n = G^n e^{\mathrm{i}kx}$, for a certain *growth factor* G and frequency k, and substituting this into the scheme, we obtain the recurrence relation

$$\frac{G^n - 2G^{n-1} + G^{n-2}}{\Delta t^2} e^{\mathrm{i}kx} = \frac{\left(e^{\mathrm{i}k\Delta x} - 2 + e^{-\mathrm{i}k\Delta x}\right)}{\Delta x^2} G^{n-1} e^{\mathrm{i}kx}. \tag{6.30}$$

We are interested in the roots of the associated characteristic polynomial, given by the solutions of

$$g^2 - 2(1 + r^2 c_2)g + 1 = 0, \tag{6.31}$$

where we named $c_2 = \cos(k\Delta x) - 1 \in (-2, 0)$, and $r = \Delta t/\Delta x$, in order to reduce notation. Its roots, denoted as $g_\pm$, satisfy the relationships

$$g_+ \, g_- = 1, \tag{6.32a}$$

$$\text{and} \quad g_+ + g_- = 2(1 + r^2 c_2). \tag{6.32b}$$

We want to ensure that $|g_\pm| \leq 1 \ \forall k$, and that roots falling on the unit circle are simple. This can only be guaranteed if these are given as a complex conjugate pair of modulus 1 (otherwise they would break (6.32a)). Consequently, we can translate (6.32b) into requesting $|\Re(g_\pm)| = |1 + r^2 c_2| \leq 1$, which only holds if $r^2 \leq 1$, given the range of c_2. For the fourth order discretisation, a similar reasoning can be applied. The roots of the characteristic polynomial associated with using SV_4 in time and CD_4 in space, are given by

$$\begin{aligned} &\left(1 - \frac{1}{12} r^2 c_4\right) g^2 - 2\left(1 + \frac{5}{12} r^2 c_4\right) g + \left(1 - \frac{1}{12} r^2 c_4\right) = 0 \\ \implies \quad &g^2 - 2\left(\frac{12 + 5r^2 c_4}{12 - r^2 c_4}\right) g + 1 = 0, \end{aligned} \tag{6.33}$$

this time with $c_4 = (16\cos(k\Delta x) - \cos(2k\Delta x) - 15)/12 \in (-8/3, 0)$. The conditions on its roots resemble those in (6.32); for stability, we then need to impose

$$|\Re(g_\pm)| = \left|\frac{12 + 5r^2 c_4}{12 - r^2 c_4}\right| \leq 1 \quad \implies \quad r^2 \leq \frac{9}{4}. \tag{6.34}$$

For higher-order methods the analysis gets more complex, as increasing the size of the stencil increases the order of the associated characteristic polynomial, but they also require constraints on the size of r in order to ensure stability.

To avoid these restrictions, we also introduce another discretisation, namely an implicit version of the Leapfrog method [105], which we consider only to second-order accuracy, and which we denote by CI_2. The only difference with respect to the explicit Leapfrog scheme lies in the fact that the spatial operator is averaged over three consecutive time steps, giving rise to the following recurrence relation:

$$\frac{u^n - u^{n-1} + u^{n-2}}{\Delta t^2} = \frac{1}{\Delta x^2} \mathcal{K}^{\mathrm{CD}_2} \frac{u^n + 2u^{n-1} + u^{n-1}}{4}. \tag{6.35}$$

Even in this case, von Neumann analysis comes to our aid in establishing that the method is *unconditionally stable*. After some algebraic manipulations, we can show that the characteristic polynomial of interest this time is given by

$$g^2 - 2\left(\frac{2 + r^2 c_2}{2 - r^2 c_2}\right) g + 1 = 0, \tag{6.36}$$

with $c_2 \in (-2, 0)$ defined as before. Similar considerations as for the case above show that its roots are a complex conjugate pair falling on the unit circle whenever $|2 + r^2 c_2| < |2 - r^2 c_2|$; given c_2 is always negative, this holds $\forall r^2 > 0$.

We are finally ready to assemble the monolithic space-time system, by opportunely combining the matrices introduced above. For CI_2, or SV_k temporal discretisations of a given order k, we do so in the following block-wise fashion:

$$\begin{aligned} A^{\mathrm{SV}_k} &:= \Phi^{\mathrm{SV}_k} \otimes I^{N_x} \quad -r^2 \left(\Phi_r^{\mathrm{SV}_k} \otimes \mathcal{K}^{\mathrm{CD}_{2\lceil \frac{k}{2} \rceil}}\right), \\ \text{and} \quad A^{\mathrm{CI}_2} &:= \Phi^{\mathrm{SV}_2} \otimes I^{N_x} \quad -\tfrac{r^2}{4} \left(\tilde{I}^{N_t} + 2I_{-1}^{N_t} + I_{-2}^{N_t}\right) \otimes \mathcal{K}^{\mathrm{CD}_2}, \end{aligned} \tag{6.37}$$

where $\otimes$ represents the *Kronecker product*, $\lceil * \rceil$ the *ceiling function*, and $I_{-i}^{N_t}$ is the *lower shift matrix*, containing only ones on the i-th sub-diagonal. Also, $\tilde{I}^{N_t}$ is a slight modification of the identity matrix, containing a 2 in the second row, to account for the initial conditions on the derivative. Notice how the part for A^{CI_2} referring to the spatial discretisation (the term containing the Kronecker product with $\mathcal{K}^{\mathrm{CD}_2}$) is made of three adjacent blocks per block row, corresponding to an averaging over three consecutive instants. The corresponding right-hand sides are assembled in a similar way as to what was done in Sec. 6.2:

$$\begin{aligned} \boldsymbol{g}^{\mathrm{SV}_1} &= \left[\left(I^{N_x} - r^2 \mathcal{K}^{\mathrm{CD}_2}\right) \bar{u}^0, 2\Delta t\, \bar{u}^1, 0, \ldots, 0\right]^T, \\ \boldsymbol{g}^{\mathrm{SV}_k} &= \left[\left[A^{\mathrm{SV}_k}\right]_{i,j=0}^{N_x - 1} \bar{u}^0, \ldots, \left[A^{\mathrm{SV}_k}\right]_{i,j=(k-2)N_x}^{(k-1)N_x - 1} \bar{u}^{k-2}, 0, \ldots, 0\right]^T \quad \text{for} \quad k \geq 3, \\ \text{and} \quad \boldsymbol{g}^{\mathrm{CI}_2} &= \left[\left(I^{N_x} - \tfrac{r^2}{4} \mathcal{K}^{\mathrm{CD}_2}\right) \bar{u}^0, 2\left(\Delta t\, I^{N_x} - \tfrac{r^2}{4} \mathcal{K}^{\mathrm{CD}_2}\right) \bar{u}^1, 0, \ldots, 0\right]^T. \end{aligned} \tag{6.38}$$

Notice that, since we defined the monolithic systems discretising (6.27) as a Kronecker product with the (quasi) Toeplitz matrices in Sec. 6.2, we still retain a (quasi) block Toeplitz structure even for the PDE case. We can then build our block circulant preconditioner.

6.3.2 Block circulant preconditioning

Analogously to what was done in Sec. 6.2, the following block circulant preconditioners are built for the systems (6.37),

$$\begin{aligned} M^{\mathrm{SV}_k} &:= \hat{\Phi}^{\mathrm{SV}_k} \otimes I^{N_x} \quad -r^2 \left(\hat{\Phi}_r^{\mathrm{SV}_k} \otimes \mathcal{K}^{\mathrm{CD}_{2\lceil \frac{k}{2} \rceil}} \right), \\ \text{and} \quad M^{\mathrm{CI}_2} &:= \hat{\Phi}^{\mathrm{SV}_2} \otimes I^{N_x} \quad -\tfrac{r^2}{4} \left(I^{N_t} + 2\hat{I}_{-1}^{N_t} + \hat{I}_{-2}^{N_t} \right) \otimes \mathcal{K}^{\mathrm{CD}_2}. \end{aligned} \tag{6.39}$$

where $\hat{I}_{-i}^{N_t}$ is the *circulant lower shift matrix*, that is the circulant counterpart to $I_{-i}^{N_t}$, containing ones both on the i-th sub-diagonal and the $(N_t - i)$-th super-diagonal. As described in Sec. 6.1, these preconditioners can be (block) diagonalised using a (block) FFT. For instance, for SV_2 we have:

$$M^{\mathrm{SV}_2} = \left(F_{N_t} \otimes I^{N_x}\right) \underbrace{\begin{pmatrix} & I^{N_t} \otimes I^{N_x} \\ - & \Lambda_1^{N_t} \otimes \left(2\, I^{N_x} + r^2 \mathcal{K}^{\mathrm{CD}_2}\right) \\ + & \Lambda_2^{N_t} \otimes I^{N_x} \end{pmatrix}}_{=: D^{\mathrm{SV}_2}} \left(F_{N_t}^* \otimes I^{N_x}\right), \tag{6.40}$$

where $\Lambda_n^{N_t}$ are the diagonal matrices containing the N_t-th roots of unity, sampled at frequency n: more explicitly, we have $\Lambda_n^{N_t} = \mathrm{diag}_{k=0,\cdots,N_t-1}\left(e^{\mathrm{i}2\pi nk/N_t}\right) = F_{N_t}^* \hat{I}_{-n}^{N_t}$. In order to apply the inverse of (6.40) to a space-time vector $\boldsymbol{b} \in \mathbb{R}^{N_x N_t}$, we then need to perform the following sequence of operations:

Step 1 *Apply* $F_{N_t}^* \otimes I^{N_x}$. This corresponds to computing a total of N_x different FFTs on the coefficients of $\boldsymbol{b}$. Each of these FFTs can be performed independently. The signals that need to be Fourier-transformed, denoted as $\boldsymbol{s}_n \in \mathbb{R}^{N_t}$, are each of length N_t and are recovered by collecting every N_x coefficients of $\boldsymbol{b}$, starting from the n-th coefficient:

$$[\boldsymbol{s}_n]_i = [\boldsymbol{b}]_{iN_x+n}, \qquad \text{with} \quad n = 0, \dots, N_x - 1, \quad \text{and} \quad i = 0, \dots, N_t - 1. \tag{6.41}$$

Basically, we are tracking how the solution at each node evolves in time, and applying a Fourier-transform to each of these evolutions. The FFTs produce the transformed signals $\hat{\boldsymbol{s}}_n = F_{N_t}^* \boldsymbol{s}_n$. Their coefficients are then re-arranged in order to recover, as a final result, the vector

$$[\hat{\boldsymbol{s}}]_i = \left[\hat{\boldsymbol{s}}_{\lfloor i/N_x \rfloor}\right]_{i\%N_x}, \qquad i = 0, \dots, N_x N_t - 1. \tag{6.42}$$

Step 2 *Invert* D^{SV_2}. As already pointed out, this matrix is block diagonal. As a consequence, this step requires solving for N_t independent systems (the blocks on

the diagonal), using $[\hat{\boldsymbol{s}}]_{nN_x}^{(n+1)N_x-1}$, $n = 0, \dots, N_t - 1$, as right-hand sides. Looking at (6.40) we can see how each of these solves requires inverting a complex-shifted discrete Laplacian operator.

Step 3 *Apply* $F_{N_t} \otimes I^{N_x}$. The last step is equivalent to the first, except N_x *inverse* FFTs (iFFT) must be performed.

6.3.3 Cost analysis of circulant preconditioning *versus* time-stepping

Canonically, approximate solutions to (6.27) are recovered via time-stepping, which corresponds to applying block forward substitution to space-time matrices such as (6.37). This is an inherently sequential procedure which, at least for implicit methods, requires inverting a shifted Laplacian at each time step. Assuming inverting the Laplacian operator has an associated cost of $C(N_x)$, the overall procedure has a complexity of $\mathcal{O}(N_t C(N_x))$. Given the availability of optimal solvers for this kind of operators [150], we can expect $C(N_x)$ to scale linearly with N_x, with spatial parallelisation eventually coming to our aid in reducing the required computational time.

Conversely, our procedure dramatically increases the global cost required for solving systems with the coefficient matrix (6.37). For each GMRES iteration, in fact, we need to multiply the space-time residual by the matrix (6.37), an operation which by itself has a complexity of $\mathcal{O}(N_t N_x)$. On top of this, applying the preconditioner involves N_x independent FFT and iFFT, of overall complexity $\mathcal{O}(N_x N_t \log(N_t))$, as well as inverting N_t systems involving a complex-shifted Laplacian. The latter can be safely assumed to have a cost comparable to $C(N_x)$ (in fact, even for this operator optimal solvers are available, for example in [44,95]), which makes the application of the preconditioner a procedure with an overall complexity of $\mathcal{O}(N_x N_t \log(N_t) + N_t C(N_x))$. Unlike time-stepping, however, solution via GMRES with circulant preconditioning exposes parallelisation along the temporal domain, as stated in Sec. 6.1.1.

This can be exploited to reduce computational time at each step of the GMRES iteration. Firstly, when multiplying by the space-time matrix, if we have N_t processors available, each assigned to a specific instant (or block row in the matrix), then the computational time necessary is effectively reduced from $\mathcal{O}(N_t N_x)$ to $\mathcal{O}(N_x)$, if we assume that time-to-solution is directly proportional to complexity[1]. Secondly, when

[1]Here we are neglecting the fact that, if each processor contains information only pertaining to a specific instant, a certain amount of communication must occur in order to perform multiplication by the space-time matrix. This overload is however limited, since the space-time matrix has a small band of size $s_1 + s_2$, implying that this exchange of information only involves $s_1 + s_2 - 1$ processors

inverting the complex shifted Laplacians, time similarly reduces from $\mathcal{O}(N_t C(N_x))$ to $\mathcal{O}(C(N_x))$, since again each system can be inverted independently, (and no communication must occur among processors). Thirdly, a similar reasoning can also be followed when applying the Fourier transforms: not only the FFTs on each of the N_x signals are independent, meaning that any additional processor for spatial parallelisation can still be employed to attack the N_x factor in the $\mathcal{O}(N_x N_t \log(N_t))$ complexity, but also parallelisation over time is perfectly feasible. Most of the FFT algorithms available involve a step consisting of operations acting over the whole signal vector (see their detailed description in [13, 22]): the operations on each element are independent and can hence be parallelised (as is done in [84, 146]), ideally dropping computational time from $\mathcal{O}(N_x N_t \log(N_t))$ to $\mathcal{O}(N_x \log(N_t))$.

The actual performance of the parallel FFT depends heavily on how communication among processors is organised, and on the architecture of the system considered [53]: since FFT represents the bottleneck in the application of the circulant preconditioner, any efficient parallel implementation will need to be fine-tuned in order to take this into account. This is however beyond the scope of this work, which rather wishes to focus on the efficacy of the preconditioner when applied to space-time systems.

To summarise, we have on the one hand time-stepping: a sure-fire direct procedure which recovers the solution to a space-time system in a number of operations scaling as $\sim N_t C(N_x)$; on the other, we have an iterative procedure whose computational time *per iteration* scales as $\sim C(N_x) + N_x \log(N_t)$. The difference between the two regimes allows some room for speedup, provided that the number of necessary iterations remains bounded, and particularly that it does not scale with N_x, nor with N_t. Even in the PDE case, we can still invoke Thm. 6.1.1 to secure ourselves from the latter, since it confirms that the maximum theoretical number of iterations required for convergence is independent of the number of time steps taken. Unfortunately, though, Thm. 6.1.1 does not shield us from the former, since in principle the number of iterations could grow with the size of the blocks composing the matrices (6.37), *i.e.*, the number of spatial nodes N_x considered in the discretisation. In practice, however, and particularly under certain regimes, this seems not to be the case. Details on this are shown in the following.

---

at most, and in particular its cost per processor is independent on N_t.

6.3.4 Results

In this section, we report the results from the application of GMRES for the solution of systems involving (6.37) as coefficient matrix, using the corresponding (6.39) as right-preconditioners. In particular, and in light of the considerations made in the previous paragraph, we are interested in its performance in terms of number of iterations to convergence.

Table 6.2: Number of iterations to convergence for GMRES, right-preconditioned with (6.7), applied to the solution of systems arising from the discretisation of (6.27). Different number of spatial and temporal nodes considered (N_x and N_t, respectively). Time derivative approximated using SV, and space derivative approximated using CD of matching orders of accuracy. The rightmost column in each table reports the theoretical upper bound from Thm. 6.1.1; a cross identifies a simulation which did not converge due to memory requirements becoming too severe. Values for $N_x > N_t$ are not reported, as they violate the CFL condition, and give rise to unstable solutions: convergence results are very poor in those cases. The different colours in the cells background provide an indication of the accuracy of the solution for the test-case considered: the L^2 error in space-time with respect to the analytical solution decreases as we move to darker shades, becoming $< 10^{-4}$, $< 10^{-6}$, and $< 10^{-8}$, respectively.

Time: SV_1 - Space: CD_2

$N_t \rightarrow$ / $N_x \downarrow$	120	240	480	960	1920	3840	$(s_1+s_2)\cdot N_x+1$
80	15	17	26	39	70	136	161
160	–	16	19	30	53	95	321
320	–	–	18	21	33	76	641
640	–	–	–	19	22	35	1281
1280	–	–	–	–	21	23	2561
2560	–	–	–	–	–	24	5121

Time: SV_2 - Space: CD_2

$N_t \rightarrow$ / $N_x \downarrow$	120	240	480	960	1920	3840	$(s_1+s_2)\cdot N_x+1$
80	66	71	96	123	125	128	161
160	–	82	116	196	242	225	321
320	–	–	83	115	149	406	641
640	–	–	–	103	153	144	1281
1280	–	–	–	–	130	204	2561
2560	–	–	–	–	–	159	5121

Time: $SV_3 = SV_4$ - Space: CD_4

$N_t \rightarrow$ / $N_x \downarrow$	120	240	480	960	1920	3840	$(s_1+s_2)\cdot N_x+1$
80	49	67	91	105	114	118	161
160	–	46	55	81	110	153	321
320	–	–	30	31	33	46	641
640	–	–	–	25	25	27	1281
1280	–	–	–	–	27	28	2561
2560	–	–	–	–	–	26	5121

Time: SV_5 - Space: CD_6

$N_t \rightarrow$ / $N_x \downarrow$	120	240	480	960	1920	3840	$(s_1+s_2)\cdot N_x+1$
80	132	140	157	189	192	199	321
160	–	71	123	120	143	151	641
320	–	–	65	183	177	137	1281
640	–	–	–	97	388	171	2561
1280	–	–	–	–	167	325	5121
2560	–	–	–	–	–	92	10241

Time: SV_6 - Space: CD_6

$N_t \rightarrow$ / $N_x \downarrow$	120	240	480	960	1920	3840	$(s_1+s_2)\cdot N_x+1$
80	181	229	205	212	222	225	401
160	–	248	244	186	169	164	801
320	–	–	263	348	264	253	1601
640	–	–	–	310	465	368	3201
1280	–	–	–	–	335	907	6401
2560	–	–	–	–	–	588	12801

Time: SV_7 - Space: CD_8

$N_t \rightarrow$ / $N_x \downarrow$	120	240	480	960	1920	3840	$(s_1+s_2)\cdot N_x+1$
80	213	258	221	242	249	256	481
160	–	312	390	259	241	176	961
320	–	–	480	506	406	270	1921
640	–	–	–	844	822	395	3841
1280	–	–	–	–	1573	943	7681
2560	–	–	–	–	–	×	15261

For all of our experiments, GMRES is set to a tolerance of 10^{-10}, starting from a null initial guess. We consider the unit square, $T = L = 1$, as a domain for the PDE

Table 6.3: Number of iterations to convergence for GMRES, right-preconditioned with (6.7), applied to the solution of systems arising from the discretisation of (6.27). Different number of spatial and temporal nodes considered (N_x and N_t, respectively). Time derivatives approximated using CI_2, space derivatives using CD with various accuracies. The rightmost column in each table reports the theoretical upper bound from Thm. 6.1.1. The different shades indicate the accuracy with respect to the analytical solution as in Tab. 6.2.

Time: CI_2 - Space: CD_2

$N_t\rightarrow$ $N_x\downarrow$	120	240	480	960	1920	3840	$(s_1+s_2)\cdot N_x+1$
80	61	75	111	124	125	128	161
160	83	92	136	203	243	226	321
320	116	119	140	146	224	372	641
640	242	188	152	157	183	137	1281
1280	767	403	227	205	221	240	2561
2560	1266	1476	582	338	234	279	5121

Time: CI_2 - Space: CD_4

$N_t\rightarrow$ $N_x\downarrow$	120	240	480	960	1920	3840	$(s_1+s_2)\cdot N_x+1$
80	64	70	74	90	109	112	161
160	82	99	87	70	70	94	321
320	124	117	100	86	70	74	641
640	285	187	136	125	115	78	1281
1280	799	497	208	153	161	144	2561
2560	1844	1546	587	248	250	162	5121

(6.27). As an initial condition, we choose a shifted Gaussian:

$$\bar{u}^0(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(x-\frac{L}{2}\right)^2\right) - c, \tag{6.43}$$

where c is picked so that $\bar{u}^0(0) = \bar{u}^0(L) = 0$, and $\sigma^2 = 0.002$; for the derivative, we simply take $\bar{u}^1(x) = 0$. The solutions to the systems involving the complex-shifted Laplacian are recovered at each iteration using the *backslash* command in MATLAB. The code used for the experiments is publicly available in its Git repository [27]. We test the two different schemes presented above, for a variety of orders of accuracy. Results are presented in Tab. 6.2 for SV discretisations of the temporal derivative, and in Tab. 6.3 for CI_2 discretisations.

The actual number of iterations to convergence remains well below the limit indicated by Thm. 6.1.1 for all cases considered. We notice however that this number is still far from being independent from the spatial mesh size. The convergence behaviour when increasing N_t is somewhat hard to grasp, but tends to show an increase in iterations count as N_t gets higher. Increasing N_x has a similar impact, and also slows convergence. This is expected from the statement of Thm. 6.1.1: N_x determines the size of the blocks composing (6.7), and hence directly affects the rank of the perturbation (6.10). Also increasing the order of accuracy of the time discretisations used has a similar negative effect on performance. This is also expected from Thm. 6.1.1: in this case it is the actual number of blocks in (6.10), rather than their dimension, that is directly affected.

The SV_1 scheme (top-left in Tab. 6.2) deserves a separate discussion: this scheme produces a solution which presents a noticeable degree of numerical diffusion, as

reported in the literature [65] and shown in Fig. 6.1. There, we can see how the solution profile with SV_1 (at the bottom) is visibly smeared out, contrasting with the result from leapfrog (top and middle): the difference is clear by comparing the shapes of the final negative peaks in the three surface plots, or equivalently by looking at how the heatmap at the bottom appears more blurred. The fact that diffusion is present seems to aid with the convergence of the scheme, a feature shared with other parallel-in-time integration methods (see [5,35], and the discussion in Sec. 2.1.1.1). In fact, we can see from Tab. 6.2 that the number of iterations to convergence remains reasonably small, even when the meshes are refined. However, the recovered solution is of very poor quality: even with the finest meshes used, the error with respect to the analytical solution remains above 10^{-4}.

The background colours of the cells in Tab. 6.2 draw a clearer picture of the performances of each scheme, by providing an indication of the accuracy of the recovered approximation: moving to darker shades, the numerical solution reaches an approximation error of 10^{-4}, 10^{-6}, and 10^{-8}, respectively, although at around 10^{-8} the convergence of high-order schemes tends to stall, likely because of round-off errors becoming relevant, which makes the results less informative for this last range. The first lesson we can extract from this is that there is no advantage in using a low-order scheme over a high-order one, so long as they share the same stencil: by comparing the tables relative to SV_2 and SV_4, (two schemes both with a stencil of size 3), we see that SV_4 achieves a much smaller error with respect to SV_2 on the same meshes, and often even with a smaller number of iterations. The latter is possibly due to the negative impact of the extra perturbation introduced to the Toeplitz system when accounting for the initial conditions, since we used a ghost-node approach for SV_2. As the order of the method increases, it becomes harder to make such strong statements, but if the stencil grows together with the order, then low-order, small-stencil schemes seem to be preferable. For example, SV_4 reaches an error of 10^{-6} with 25 iterations on a 640×960 mesh, while SV_5 needs 65 iterations (roughly 2.5 as many) on a mesh with 1/4 as many degrees of freedom, and SV_7 uses 312, (or roughly 12.5 as many) iterations on a mesh with only 1/16 as many degrees of freedom. Given how the cost per iteration scales *linearly* in N_x but *logarithmically* in N_t (as per the analysis conducted in the previous paragraph), SV_5 is of comparable cost with respect to SV_4, but SV_7 far exceeds it, and is hence not worth employing for the same target accuracy.

The nature of the initial conditions also has an impact on the speed of convergence. We test this by running a series of experiments where we progressively narrow

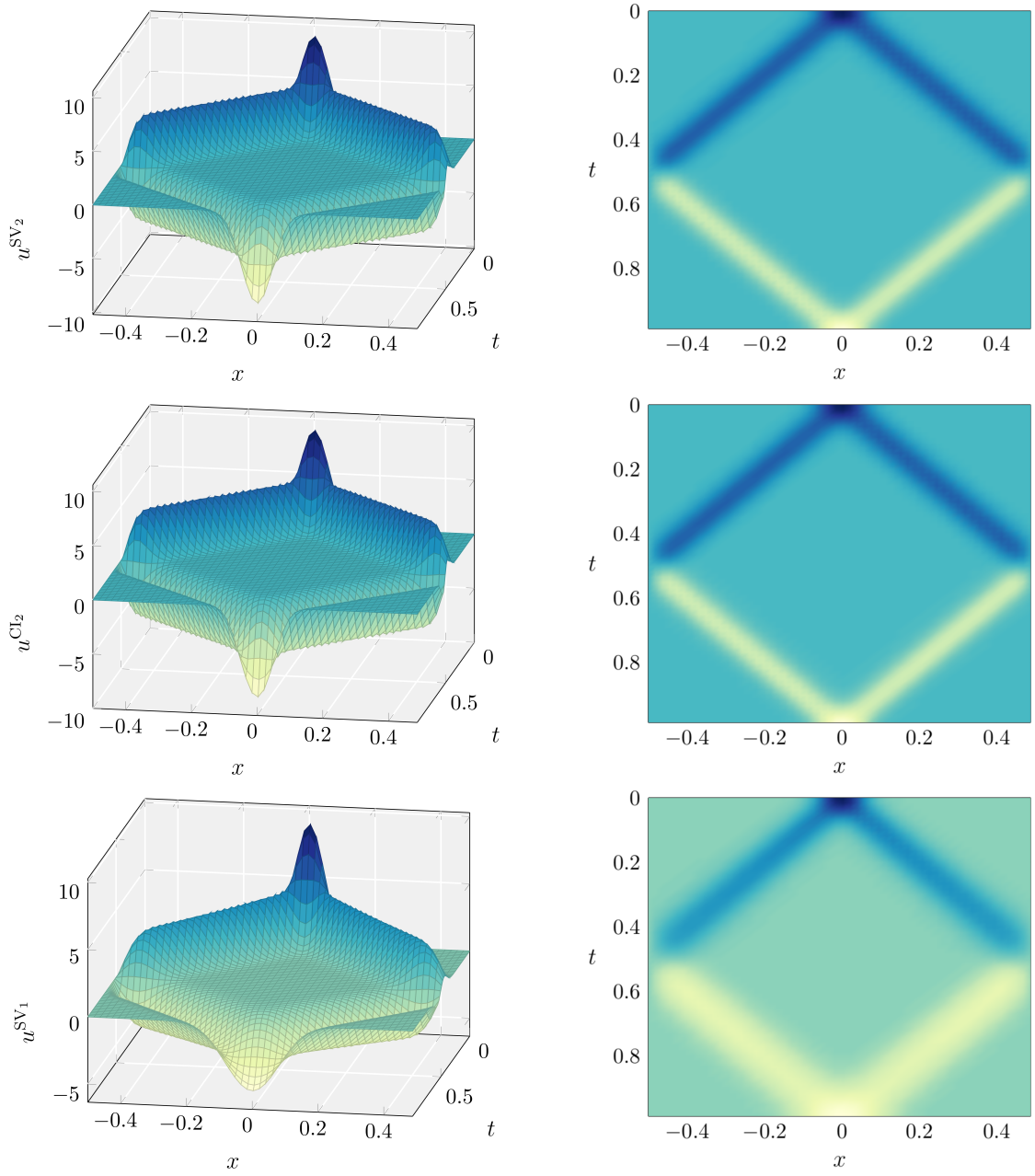


Figure 6.1: Surface plots and heatmaps of examples of numerical solutions recovered from the experiments. Top: SV_2 discretisation in time, CD_2 in space (explicit Leapfrog); middle: implicit Leapfrog, still with CD_2 space discretisation; bottom: the pair $\mathrm{SV}_1 - \mathrm{CD}_2$ is used instead. $N_t = 240$, $N_x = 160$, and other parameters as in Tab. 6.2, and Tab. 6.3. In the surface plots, time progresses towards the reader.

the Gaussian used as an initial condition. The number of iterations to convergence increases as the Gaussian becomes steeper, regardless of the discretisation used, as is shown in Tab. 6.4. This is not entirely surprising: with steeper profiles, the range of relevant modes in the solution extends to higher frequencies, and we can expect to

Table 6.4: Number of iterations to convergence for GMRES, right-preconditioned with (6.7), applied to the solution of systems arising from the discretisation of (6.27), with different initial conditions. The first row reports results for a standing wave, while the others refer to Gaussian curves with different values of σ^2 in (6.43). System size: $N_t = 240$, $N_x = 160$. Various discretisations for the temporal derivatives considered; spatial derivatives approximated using CD of matching accuracy.

	SV_1	SV_2	$SV_3 = SV_4$	SV_5	SV_6	SV_7	CI_2
$\bar{u}^0 = \sin(3\pi x/L)$	2	2	47	16	22	49	2
$\sigma^2 = 10^{-3}2^{1}$	16	82	46	71	248	312	92
$\sigma^2 = 10^{-3}2^{0}$	17	110	80	113	267	347	107
$\sigma^2 = 10^{-3}2^{-1}$	17	123	103	169	288	373	125
$\sigma^2 = 10^{-3}2^{-2}$	17	135	120	208	302	388	139
$\sigma^2 = 10^{-3}2^{-3}$	17	142	134	228	310	397	146
$\sigma^2 = 10^{-3}2^{-4}$	18	146	139	277	338	400	149
$\sigma^2 = 10^{-3}2^{-5}$	18	148	141	285	395	401	151

require a richer Krylov subspace in order to properly approximate them. An extreme example of the converse can be shown by picking a single mode as an initial condition, of any frequency. This gives rise to a stationary wave, which is particularly simple to track, since the spatial and the temporal components of the solution factorise: the solution at each instant is nothing but the initial condition, opportunely rescaled. In this case, the circulant preconditioner is extremely effective, as it is capable of building a Krylov subspace which captures the degrees of freedom of the solution in very few iterations: only 2, for all schemes of order of accuracy 2 or smaller, which matches some observations in the literature [64,65]. This is however a peculiar feature of the circulant preconditioner, limited to the very specific combination of parameters employed: if we vary the type of spatial discretisation used, (or if we add a forcing term, or vary the initial guess for the GMRES algorithm), we cannot expect such a dramatic convergence anymore, as shown in the first row of Tab. 6.4.

Since the time this research was conducted, an account on the parallel performance of this algorithm has been provided in [93]. This, and the analysis conducted in this chapter, provide evidence for stating that circulant preconditioning is a feasible PinT method for hyperbolic PDEs. The biggest obstacle in its application, however, lies in the scarce generalisability of the scheme, which relies on the Toeplitz structure of the target space-time system. How to overcome this limitation is still largely a matter of research: some suggestions for future work in this regard, as well as for the other topics covered in the scope of this thesis, are reported in the next and final chapter.

Chapter 7

Conclusion

We conclude this thesis by providing a brief summary of the results of our work. This is done in the following section, where for each chapter we indicate the major findings stemming from the analyses conducted therein. Section 7.2 instead lists a number of issues which have not been addressed with our research, together with suggestions for possible solutions, which will be the matter of future research. As a final comment, in Sec. 7.3 we report our personal intuition behind the scope of space-time block preconditioning, the main technique developed in the course of this project.

7.1 Summary of findings

Space-time block preconditioning. In Chap. 3, we introduce a novel approach to block preconditioning for space-time systems. Its purpose is to reduce the solution of a space-time system of PDEs to that of a single-variable space-time PDE, which is a much more tractable problem for PinT solvers. We consider the discretised time-dependent Oseen equations as an example to describe our approach. In designing our space-time block preconditioner, we take inspiration from heuristics that have proven to be successful in the single time-step or steady-state setting, and extend similar principles to the whole space-time formulation. The resulting preconditioning procedure involves two operations: the space-time solution of a single-variable, time-dependent advection-diffusion equation, and a (spatial) pressure mass- and stiffness-matrix inverse at each time point. The latter operation is trivially parallelisable over the time-steps, so that the computational time it requires is comparable to its analogue for sequential time-stepping. For the former, efficient time-parallelisation techniques are readily available. We demonstrate the effectiveness of our preconditioner by applying it to a variety of model problems taken from the literature, which cover a range of flow configurations. The main measure for performance is given by the number

of iterations required for convergence, which we show to be scalable in spatial and temporal mesh size, and comparable to those required for a classical time-stepping (non-time-parallel) routine. We investigate a simple extension to a nonlinear Navier-Stokes problem as well: this is solved via Picard iterations, and even in this case we observe near perfect scalability, for both linear and nonlinear solvers.

Applications to incompressible magnetohydrodynamics. In Chap. 4, we extend the work from the previous chapter, and design a space-time block preconditioner for a more complicated, nonlinear problem in magnetohydrodynamics. This gives evidence for the flexibility of our STBP approach, paving the way for its application to other systems of PDEs. The system considered represents the main target application of our thesis, and is of particular relevance for our industrial partner, due to the fundamental role it plays in modelling the behaviour of plasma within a fusion reactor. Following the principles outlined in the previous chapter, we first identify a preconditioner which has shown its effectiveness in the time-stepping framework, and then proceed to extend it to the whole space-time setting. The resulting space-time block preconditioner acts by addressing two simplified subproblems in sequence: it solves first a system coupling magnetic field and velocity, followed by one coupling velocity and pressure. The latter is equivalent to the one tackled in Chap. 3, while the former requires inverting the discretisation of a single-variable, high-order space-time differential operator, analogous to the one arising in the sequential time-stepping setting. How to address the parallel-in-time solution of this operator remains a matter of future research, and to keep our work contained we focus on evaluating the performance of the space-time block preconditioner when using exact solvers. This is done by measuring iterations to convergence for both the linear and nonlinear solvers. Applying the space-time block preconditioner requires limited overhead with respect to its sequential time-stepping counterpart, and it shows excellent scalability with respect to spatial refinement and only marginal convergence degradation with temporal refinement.

Multigrid Reduction in Time for nonlinear hyperbolic equations. With Chap. 5, our focus moves from the time-parallel solution of systems of PDEs to that of single-variable PDEs. In particular, we consider the performance of PinT methods when applied to hyperbolic equations. In this chapter, the algorithm of choice is MGRIT, in light of its maturity and the flexibility it offers as a multigrid method, and we investigate its applicability to the solution of nonlinear conservation laws.

The aim is to understand how the choice of integrators used at each level of the multigrid algorithm impacts its convergence behaviour. To this purpose, we measure the performance of MGRIT applied using a combination of existing integrators, commonly used for the solution of conservation laws. From the results we can infer that dissipative schemes behave better in general, which is in line with the literature; however, higher-order methods for spatial reconstruction coupled with matching order time-discretisations still provide satisfactory convergence results, under suitable CFL limits. We note the importance of choosing coarse integrators that closely follow the action of the fine integrator, by showcasing the effectiveness of a new method proposed for the construction of accurate coarse solvers. This approach seeks to directly approximate the action of the fine solver: even though it comes at an increased cost with respect to simple rediscretisation, it offers superior performance. Its range of applicability remains, however, limited. In all cases, in fact, the performance is seen to degrade fairly quickly as the CFL number increases. This seems to be a strong limitation for the application of MGRIT to hyperbolic systems directly discretised with explicit time-steppers. The level of coarsening that can be applied to the temporal grid in this case is, thus, effectively capped. This suggests that MGRIT should be paired with a spatial coarsening strategy as well, in order to control the CFL number adequately across all levels.

Space-time circulant preconditioning for linear wave equations. Chapter 6 revolves around the investigation of a recently suggested method for the all-at-once solution of space-time systems arising from the discretisation of linear, constant-coefficient PDEs. We explore this approach as an alternative to MGRIT for the solution of hyperbolic equations, since preliminary results showed it not to suffer from the same stability limitations when applied to this type of PDEs. Under the assumptions considered, the resulting monolithic system presents a block Toeplitz structure. This is exploited to design a block circulant preconditioner for the acceleration of the iterative solution of the whole space-time system. Given that circulant operators are diagonalisable via FFT, the preconditioner can be inverted in parallel with relative ease, making this approach a *de facto* time-parallel method. We first prove the effectiveness of the resulting scheme theoretically, applying a result on the convergence of circulant-preconditioned GMRES, which guarantees that convergence is reached in a number of iterations independent on the number of temporal nodes. We then measure its performance with numerical experiments, where the preconditioner is applied to the solution of linear wave equations, discretised using high-order

schemes. The results support the theoretical findings, and testify on the effectiveness of this preconditioning strategy.

7.2 Future work

Measurement of parallel performance. The first extension of relevance to the work conducted in this thesis consists in designing an efficient parallel implementation of the space-time block preconditioners introduced in Chap. 3 and 4, in order to actually measure the parallel performance of the strategies proposed here. For the largest part, in our work we restrict ourselves to discussing the *ideal* performance of these preconditioners, focusing on giving theoretical (rather than experimental) justifications for their potential in providing parallel speedup. This choice is on the one hand dictated by the fact that the methods proposed are either relatively recent and still subject of research, or entirely novel, so it is reasonable to first test their effectiveness in a best-case scenario. On the other, including the implementation and design of efficient parallel code would have been hardly feasible in the timescale allocated for this project. Nonetheless, giving actual measurements of the time-to-solution required by these methods would be valuable in consolidating the theoretical results on their viability. In particular, for the space-time block preconditioners, this would help in gaining additional insight on how their performance depends on the choice of their internal PinT components, since in Tab. 3.1 we give only one such example. This is particularly true for the operator (4.71) appearing in the space-time magnetic Schur complement approximation (4.73), as its parallelisation in time, to our knowledge, is not covered in the literature.

Stability of STBP for advection-dominated regimes. In most of our experiments in Chap. 3 and 4, we choose the parameters for viscosity μ in (3.1) and (4.1), and of electric resistivity η and magnetic permeability μ_0 in (4.1), all to be $= 1$, thus steering away from the more challenging advection-dominated, turbulent regimes for these problems. This is justified by similar considerations as those for the previous paragraph, as well as by the fact that the single-time step PCD operator (3.38) our space-time block preconditioners rely on was not originally designed with robustness with respect to μ in mind (indeed we remark on this in Sec. 3.4.2.2). In light of the discussion in Sec. 1.2, though, the advection-dominated regime is of particular relevance for the development of fusion reactors. Fortunately, block preconditioners whose performance remains stable with respect to the problem parameters are

available in the literature: see for example [88]. This is originally designed in the single time-step framework, so the question remains open whether similar ideas can be extended to the whole space-time setting.

MGRIT with semi-implicit Lagrangian methods for hyperbolic equations. In Chap. 5 we investigate the application of MGRIT for the time parallelisation of the solution of conservation laws discretised with high-order explicit time-integration methods, and in Sec. 5.4 we remark on how one of the biggest obstacles to the stability of the algorithm is given by the need to satisfy the CFL condition at all coarsening levels. One way to circumvent this, without renouncing the computational advantages of employing explicit methods, is given by using *semi-implicit Lagrangian integrators* [80, 117]. By approximately backtracking the characteristics in the system and including a correction term in the flux definition, these schemes have the advantage of easing the restriction imposed by the CFL condition. Employing these solvers at the lower coarsening levels would allow for more leeway in considering larger time-steps, thus possibly improving the overall stability of MGRIT. A similar setup, showing promising results, is investigated for Parareal in [132].

Circulant preconditioning for nonlinear PDEs. The most cumbersome limitation behind the use of the space-time circulant preconditioner introduced in Chap. 6 lies in its reduced scope for applicability. If the block Toeplitz structure of the target space-time matrix is severely broken (be it by choosing uneven temporal grids, by introducing some temporal variation into the PDE coefficients, or by considering nonlinearities), two problems arise: on the one hand, it is unclear how one should extract an appropriate circulant preconditioner from a non-Toeplitz matrix; on the other, as hinted in Rmk. 3, the theoretical result from Thm. 6.1.1 ensuring the effectiveness of the circulant preconditioner does not hold any more. A workaround for us to still make use of the circulant preconditioner, consists in designing an effective projection operator, which allows us to recover a block-Toeplitz approximation of the target space-time system. Possibly the most straightforward way to achieve this is given by diagonally averaging the entries in the monolithic system (which corresponds to a projection in the Frobenius norm), and indeed some preliminary results on a similar approach applied to equations with time-varying coefficients are reported in [93]. The next step would consist in investigating the effectiveness of this method in the framework of nonlinear equations.

7.3 Final remarks

In our opinion, the main contribution of this research work is represented by the space-time block preconditioning approach outlined in Chap. 3 and 4. In addition to its potential for parallel efficiency, which we showcase in applications to problems in fluid dynamics and magnetohydrodynamics in the corresponding chapters, we believe that its major strength lies in the flexibility that such approach provides in dealing with time-dependent systems of PDEs.

Indeed, STBP offers a fundamentally different take on time parallelisation for multi-variable PDEs: rather than focusing on designing *ad hoc* PinT solvers for the target system, this strategy allows to link the solution of such system to that of a (hopefully simpler to parallelise) single-variable, time-dependent PDE, represented by the approximation of a space-time Schur complement. As such, it takes what is accomplished by the block preconditioning strategy commonly used for *spatial* operators, and naturally extends it to the framework of *space-time* operators.

This space-time block preconditioning approach is by no means flawless, and its worth remains in large part to be proven: the work conducted in the scope of this thesis is, after all, introductory at most. However, if the comparison with its spatial counterpart holds, and the development of efficient PinT methods for single-variable PDEs continues to progress, it is not unreasonable to believe that it can become a valid tool in the arsenal of time-parallelisation techniques.

Bibliography

[1] J. H. Adler, T. Benson, E. C. Cyr, P. E. Farrell, S. MacLachlan, and R. Tuminaro, *Monolithic multigrid for magnetohydrodynamics*, ArXiv preprint, arXiv:2006.15700 [math.NA], (2020), https://arxiv.org/abs/2006.15700.

[2] H. Alfvén, *Existence of electromagnetic-hydrodynamic waves*, Nature, 150 (1942), pp. 405–406, https://doi.org/10.1038/150405d0.

[3] G. Antonucci, *Iterative solution of evolutionary PDEs via all-at-once methods*, master's thesis, University of Oxford, 2017.

[4] D. N. Arnold, *Finite Element Exterior Calculus*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2018, https://doi.org/10.1137/1.9781611975543.

[5] 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, 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, eds., Lecture Notes in Computational Science and Engineering, Berlin, Heidelberg, 2005, Springer Berlin Heidelberg, pp. 425–432.

[6] G. Bal and Y. Maday, *A "Parareal" time discretization for non-linear PDE's with application to the pricing of an American put*, in Recent Developments in Domain Decomposition Methods, L. F. Pavarino and A. Toselli, eds., Lecture Notes in Computational Science and Engineering, Berlin, Heidelberg, 2002, Springer Berlin Heidelberg, pp. 189–202, https://doi.org/10.1007%2F978-3-642-56118-4_12.

[7] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp,

D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc Web page*. https://www.mcs.anl.gov/petsc, 2019, https://www.mcs.anl.gov/petsc. Software.

[8] A.-M. Baudron, J.-J. Lautard, Y. Maday, and O. Mula, *The parareal in time algorithm applied to the kinetic neutron diffusion equation*, in Domain Decomposition Methods in Science and Engineering XXI, J. Erhel, M. J. Gander, L. Halpern, G. Pichot, T. Sassi, and O. Widlund, eds., Cham, 2014, Springer International Publishing, pp. 437–445.

[9] D. Biskamp, *Magnetic Reconnection in Plasmas*, Cambridge Monographs on Plasma Physics, Cambridge University Press, UK, 2000, https://doi.org/10.1017/CBO9780511599958.

[10] M. Bolten, D. Moser, and R. Speck, *A multigrid perspective on the parallel full approximation scheme in space and time*, Numerical Linear Algebra with Applications, 24 (2016).

[11] M. Bolten, D. Moser, and R. Speck, *Asymptotic convergence of the parallel full approximation scheme in space and time for linear problems*, Numerical Linear Algebra with Applications, 25 (2018), p. e2208, https://doi.org/10.1002/nla.2208.

[12] W. Briggs, V. Henson, and S. McCormick, *A Multigrid Tutorial*, Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, 2nd ed., 2000.

[13] C. Burrus, M. Frigo, and G. Johnson, *Fast Fourier Transforms*, Samurai Media Limited, 2018.

[14] J. Cahouet and J.-P. Chabard, *Some fast 3d finite element solvers for the generalized Stokes problem*, International Journal for Numerical Methods in Fluids, 8 (1988), pp. 869–895, https://doi.org/10.1002/fld.1650080802.

[15] L. Chacón, D. Knoll, and J. Finn, *An implicit, nonlinear reduced resistive MHD solver*, Journal of Computational Physics, 178 (2002), pp. 15–36, https://doi.org/10.1006/jcph.2002.7015.

[16] R. H. Chan, D. Potts, and G. Steidl, *Preconditioners for non-Hermitian Toeplitz systems*, Numerical Linear Algebra with Applications, 8 (2001), pp. 83–98, https://doi.org/10.1002/1099-1506(200103)8:2<83::AID-NLA231>3.0.CO;2-X.

[17] R. H.-F. Chan and X.-Q. Jin, *An Introduction to Iterative Toeplitz Solvers*, Society for Industrial and Applied Mathematics, 2007, https://doi.org/10.1137/1.9780898718850.

[18] F. Chen, J. S. Hesthaven, Y. Maday, and A. S. Nielsen, *An adjoint approach for stabilizing the parareal method*. 2015, http://infoscience.epfl.ch/record/211097.

[19] F. Chen, J. S. Hesthaven, and X. Zhu, *On the Use of Reduced Basis Methods to Accelerate and Stabilize the Parareal Method*, Springer International Publishing, Cham, 2014, pp. 187–214, https://doi.org/10.1007/978-3-319-02090-7_7.

[20] A. T. Clarke, C. J. Davies, D. Ruprecht, and S. M. Tobias, *Parallel-in-time integration of kinematic dynamos*, Journal of Computational Physics: X, 7 (2020), p. 100057, https://doi.org/10.1016/j.jcpx.2020.100057.

[21] R. Codina and N. Hernández-Silva, *Stabilized finite element approximation of the stationary magneto-hydrodynamics equations*, Computational Mechanics, 38 (2006), pp. 344–355, https://doi.org/10.1007/s00466-006-0037-x.

[22] J. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Mathematics of Computation, 19 (1965), pp. 297–301, https://doi.org/10.1090/S0025-5718-1965-0178586-1.

[23] R. Croce, D. Ruprecht, and R. Krause, *Parallel-in-space-and-time simulation of the three-dimensional, unsteady Navier-Stokes equations for incompressible flow*, in Modeling, Simulation and Optimization of Complex Processes-HPSC 2012, Springer, 2014, pp. 13–23, https://doi.org/10.1007/978-3-319-09063-4_2.

[24] E. C. Cyr, J. N. Shadid, and R. S. Tuminaro, *Stabilization and scalable block preconditioning for the Navier-Stokes equations*, Journal of Computational Physics, 231 (2012), pp. 345–363, https://doi.org/10.1016/j.jcp.2011.09.001.

[25] E. C. Cyr, J. N. Shadid, R. S. Tuminaro, R. P. Pawlowski, and L. Chacón, *A new approximate block factorization preconditioner for two-dimensional incompressible (reduced) resistive MHD*, SIAM Journal on Scientific Computing, 35 (2013), pp. B701–B730, https://doi.org/10.1137/12088879X.

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

[27] F. Danieli, *All-at-once_code*. https://gitlab.com/fdanieli/all-at-once_code. Software repository.

[28] F. Danieli, *MGRIT_Non-linear_hyperbolic*. https://gitlab.com/fdanieli/mgrit_non-linear_hyperbolic. Software repository.

[29] F. Danieli, *STBP-incompressible-flow*. https://gitlab.com/fdanieli/stbp-incompressible-flow, 2020. Software repository.

[30] F. Danieli and S. MacLachlan, *Multigrid reduction in time for non-linear hyperbolic equations*, ArXiv preprint arXiv:2104.09404 [math.NA], (2021), https://arxiv.org/abs/2104.09404.

[31] F. Danieli, B. S. Southworth, and A. J. Wathen, *Space-time block preconditioning for incompressible flow*, ArXiv preprint arXiv:2101.07003 [math.NA], (2021), https://arxiv.org/abs/2101.07003.

[32] F. Danieli and A. J. Wathen, *All-at-once solution of linear wave equations*, Numerical Linear Algebra with Applications, p. e2386, https://doi.org/10.1002/nla.2386.

[33] H. De Sterck, R. D. Falgout, S. Friedhoff, O. A. Krzysik, and S. P. MacLachlan, *Optimizing multigrid reduction-in-time and Parareal coarse-grid operators for linear advection*, Numerical Linear Algebra with Applications, (2021), https://doi.org/10.1002/nla.2367.

[34] J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Society for Industrial and Applied Mathematics, 1996, https://doi.org/10.1137/1.9781611971200.

[35] V. Dobrev, T. Kolev, N. Petersson, and J. Schroder, *Two-level convergence theory for multigrid reduction in time (MGRIT)*, SIAM Journal on Scientific Computing, 39 (2017), pp. S501–S527, https://doi.org/10.1137/16M1074096.

[36] J. Dongarra and P. Luszczek, *TOP500*, Springer US, Boston, MA, 2011, pp. 2055–2057, https://doi.org/10.1007/978-0-387-09766-4_157.

[37] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT Numerical Mathematics, 40 (2000), pp. 241–266, https://doi.org/10.1023/A:1022338906936.

[38] A. Eghbal, A. G. Gerber, and E. Aubanel, *Acceleration of unsteady hydrodynamic simulations using the parareal algorithm*, Journal of Computational Science, 19 (2017), pp. 57 – 76, https://doi.org/10.1016/j.jocs.2016.12.006.

[39] H. Elman, V. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, *A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations*, Journal of Computational Physics, 227 (2008), pp. 1790–1808, https://doi.org/10.1016/j.jcp.2007.09.026.

[40] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Numerical Mathematics and Scientific Computation, Oxford University Press, UK, 2014, https://doi.org/10.1093/acprof:oso/9780199678792.001.0001.

[41] H. Elman and R. Tuminaro, *Boundary conditions in approximate commutator preconditioners for the Navier-Stokes equations*, Electronic Transactions on Numerical Analysis, 35 (2009), pp. 257–280.

[42] H. C. Elman, A. Ramage, and D. J. Silvester, *IFISS: A computational laboratory for investigating incompressible flow problems*, SIAM Review, 56 (2014), pp. 261–273, https://doi.org/10.1137/120891393.

[43] M. Emmett and M. L. Minion, *Toward an efficient parallel in time method for partial differential equations*, Communications in Applied Mathematics and Computational Science, 7 (2012), pp. 105–132, https://doi.org/10.2140/camcos.2012.7.105.

[44] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik, *A novel multigrid based preconditioner for heterogeneous Helmholtz problems*, SIAM Journal on Scientific Computing, 27 (2006), pp. 1471–1492, https://doi.org/10.1137/040615195.

[45] V. M. Fadeev, I. F. Kvabtskhava, and N. N. Komarov, *Self-focusing of local plasma currents*, Nuclear Fusion, 5 (1965), pp. 202–209, https://doi.org/10.1088/0029-5515/5/3/003.

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

[47] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid*, SIAM Journal on Scientific Computing, 36 (2014), pp. C635–C661, https://doi.org/10.1137/130944230.

[48] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, J. B. Schroder, and S. Vandewalle, *Multigrid methods with space-time concurrency*, Computing and Visualization in Science, 18 (2017), pp. 123–143, https://doi.org/10.1007/s00791-017-0283-9.

[49] R. D. Falgout, M. Lecouvez, and C. S. Woodward, *A parallel-in-time algorithm for variable step multistep methods*, in LLNL Technical Report, 2017.

[50] P. Fischer, F. Hecht, and Y. Maday, *A parareal in time semi-implicit approximation of the Navier-Stokes equations*, vol. 40, 01 2005, pp. 433–440, https://doi.org/10.1007/3-540-26825-1_44.

[51] H. Foerster, K. Stüben, and U. Trottenberg, *Non-standard multigrid techniques using checkered relaxation and intermediate grids*, in Elliptic Problem Solvers, M. H. Schultz, ed., Academic Press, 1981, pp. 285–300, https://doi.org/doi.org/10.1016/B978-0-12-632620-8.50027-9.

[52] B. Fornberg, *Generation of finite difference formulas on arbitrarily spaced grids*, Mathematics of Computation, 51 (1988), pp. 699–706, https://doi.org/10.2307/2008770.

[53] M. Frigo and S. G. Johnson, *The design and implementation of FFTW3*, Proceedings of the IEEE, 93 (2005), pp. 216–231, https://doi.org/10.1109/JPROC.2004.840301.

[54] M. J. Gander, *Analysis of the Parareal algorithm applied to hyperbolic problems using characteristics*, Boletín de la Sociedad Española de Matemática Aplicada, 42 (2008), pp. 21–35.

[55] M. J. Gander, *50 years of time parallel time integration*, in Multiple Shooting and Time Domain Decomposition Methods, T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds., Cham, 2015, Springer International Publishing, pp. 69–113, https://doi.org/10.1007/978-3-319-23321-5_3.

[56] M. J. Gander and S. Güttel, *ParaExp: A parallel integrator for linear initial-value problems*, SIAM Journal on Scientific Computing, 35 (2013), pp. C123–C142, https://doi.org/10.1137/110856137.

[57] M. J. Gander, S. Güttel, and M. Petcu, *A nonlinear ParaExp algorithm*, in Domain Decomposition Methods in Science and Engineering XXIV, P. E. Bjørstad, S. C. Brenner, L. Halpern, H. H. Kim, R. Kornhuber, T. Rahman, and O. B. Widlund, eds., Cham, 2018, Springer International Publishing, pp. 261–270.

[58] M. J. Gander and L. Halpern, *Time parallelization for nonlinear problems based on diagonalization*, in Domain Decomposition Methods in Science and Engineering XXIII, C.-O. Lee, X.-C. Cai, D. E. Keyes, H. H. Kim, A. Klawonn, E.-J. Park, and O. B. Widlund, eds., Cham, 2017, Springer International Publishing, pp. 163–170.

[59] 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, T. Dickopf, M. J. Gander, L. Halpern, R. Krause, and L. F. Pavarino, eds., Cham, 2016, Springer International Publishing, pp. 491–499.

[60] M. J. Gander, J. Liu, S.-L. Wu, X. Yue, and T. Zhou, *ParaDiag: parallel-in-time algorithms based on the diagonalization technique*, ArXiv preprint arXiv:2005.09158 [math.NA], (2021), https://arxiv.org/abs/2005.09158.

[61] M. J. Gander and M. Neumüller, *Analysis of a new space-time parallel multigrid algorithm for parabolic problems*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2173–A2208, https://doi.org/10.1137/15M1046605.

[62] M. J. Gander and S. Vandewalle, *Analysis of the Parareal time-parallel time-integration method*, SIAM Journal on Scientific Computing, 29 (2007), pp. 556–578, https://doi.org/10.1137/05064607X.

[63] Gander, M. and Petcu, M., *Analysis of a Krylov subspace enhanced parareal algorithm for linear problems*, ESAIM Proceedings, 25 (2008), pp. 114–129, https://doi.org/10.1051/proc:082508.

[64] A. Goddard, *An implementation of the all-at-once method to evolutionary PDEs*, master's thesis, University of Oxford, 2018.

[65] A. Goddard and A. Wathen, *A note on parallel preconditioning for all-at-once evolutionary PDEs*, Electronic Transactions on Numerical Analysis, 51 (2019), pp. 135–150.

[66] R. J. Goldston and P. H. Rutherford, *Introduction to plasma physics*, Institute of Physics Publishing, Bristol, 1995.

[67] S. Gottlieb, C. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Review, 43 (2001), pp. 89–112, https://doi.org/10.1137/S003614450036757X.

[68] A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM Journal on Matrix Analysis and Applications, 17 (1996), pp. 465–469, https://doi.org/10.1137/S0895479894275030.

[69] S. Günther, N. Gauger, and J. Schoder, *A non-intrusive parallel-in-time adjoint solver with the XBraid library*, Computing and Visualization in Science, 19 (2018), https://doi.org/10.1007/s00791-018-0300-7.

[70] E. Hairer, S. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer Series in Computational Mathematics, Springer Berlin Heidelberg, 2008, https://doi.org/10.1007/978-3-540-78862-1.

[71] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer Series in Computational Mathematics, Springer Berlin Heidelberg, 2010, https://doi.org/10.1007/978-3-642-05221-7.

[72] A. Harten and J. M. Hyman, *Self adjusting grid methods for one-dimensional hyperbolic conservation laws*, Journal of Computational Physics, 50 (1983), pp. 235 – 269, https://doi.org/10.1016/0021-9991(83)90066-9.

[73] T. Haut and B. Wingate, *An asymptotic parallel-in-time method for highly oscillatory PDEs*, SIAM Journal on Scientific Computing, 36 (2014), pp. A693–A713, https://doi.org/10.1137/130914577.

[74] A. Hessenthaler, R. D. Falgout, J. B. Schroder, A. de Vecchi, D. Nordsletten, and O. Röhrle, *Time-periodic steady-state solution of fluid-structure interaction and cardiac flow problems through multigrid-reduction-in-time*, ArXiv preprint, arXiv:2105.00305 [cs.CE], (2021), https://arxiv.org/abs/2105.00305.

[75] J. Hesthaven, *Numerical Methods for Conservation Laws*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017, https://doi.org/10.1137/1.9781611975109.

[76] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numerica, 19 (2010), p. 209–286, https://doi.org/10.1017/S0962492910000048.

[77] G. Horton and S. Vandewalle, *A space-time multigrid method for parabolic partial differential equations*, SIAM Journal on Scientific Computing, 16 (1995), pp. 848–864, https://doi.org/10.1137/0916050.

[78] A. Howse, H. Sterck, R. Falgout, S. MacLachlan, and J. Schroder, *Parallel-in-time multigrid with adaptive spatial coarsening for the linear advection and inviscid burgers equations*, SIAM Journal on Scientific Computing, 41 (2019), pp. A538–A565, https://doi.org/10.1137/17M1144982.

[79] K. Hu. Private communication.

[80] C.-S. Huang, T. Arbogast, and C.-H. Hung, *A semi-Lagrangian finite difference WENO scheme for scalar nonlinear conservation laws*, Journal of Computational Physics, 322 (2016), pp. 559–585, https://doi.org/10.1016/j.jcp.2016.06.027.

[81] *HYPRE: Scalable linear solvers and multigrid methods*. www.llnl.gov/casc/hypre/. Software.

[82] *IFISS: Incompressible flow and iterative solver software*. https://personalpages.manchester.ac.uk/staff/david.silvester/ifiss/default.htm. Software.

[83] J. D. Jackson, *Classical electrodynamics*, Wiley, New York, NY, 3rd ed., 1999.

[84] L. Jamieson, P. T. Mueller, and H. Siegel, *FFT algorithms for SIMD parallel processing systems*, Journal of Parallel and Distributed Computing, 3 (1986), pp. 48–71, https://doi.org/10.1016/0743-7315(86)90027-4.

[85] D. Kay, D. Loghin, and A. Wathen, *A preconditioner for the steady-state Navier-Stokes equations*, SIAM Journal on Scientific Computing, 24 (2002), pp. 237–256, https://doi.org/10.1137/S106482759935808X.

[86] A. Kreienbuehl, A. Naegel, D. Ruprecht, R. Speck, G. Wittum, and R. Krause, *Numerical simulation of skin transport using parareal*, Computing and Visualization in Science, 17 (2015), pp. 99–108, https://doi.org/10.1007/s00791-015-0246-y.

[87] I. Kulchytska-Ruchka and S. Schöps, *Efficient parallel-in-time solution of time-periodic problems using a multiharmonic coarse grid correction*, SIAM Journal on Scientific Computing, 43 (2021), pp. C61–C88, https://doi.org/10.1137/20M1314756.

[88] F. Laakmann, P. E. Farrell, and L. Mitchell, *An augmented lagrangian preconditioner for the magnetohydrodynamics equations at high Reynolds and coupling numbers*, ArXiv preprint, arXiv:2104.14855 [math.NA], (2021), https://arxiv.org/abs/2104.14855.

[89] R. LeVeque, *Numerical Methods for Conservation Laws*, Lectures in Mathematics ETH Zürich, Department of Mathematics Research Institute of Mathematics, Springer, 1992.

[90] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*, Society for Industrial and Applied Mathematics, 2007, https://doi.org/10.1137/1.9780898717839.

[91] A. Limache, S. Idelsohn, R. Rossi, and E. Oñate, *The violation of objectivity in Laplace formulations of the Navier-Stokes equations*, International Journal for Numerical Methods in Fluids, 54 (2007), pp. 639–664, https://doi.org/10.1002/fld.1480.

[92] J.-L. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps ⟨⟨pararéel⟩⟩*, Comptes Rendus de l'Académie des Sciences - Series I - Mathematics, 332 (2001), pp. 661 – 668, https://doi.org/10.1016/S0764-4442(00)01793-6.

[93] J. Liu and S.-L. Wu, *A fast block α-circulant preconditioner for all-at-once system from wave equations*, SIAM Journal on Matrix Analysis and Applications, (2020), https://doi.org/10.1137/19M1309869.

[94] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, Journal of Computational Physics, 115 (1994), pp. 200 – 212, https://doi.org/10.1006/jcph.1994.1187.

[95] S. MacLachlan and K. Oosterlee, *Algebraic multigrid solvers for complex-valued matrices*, SIAM Journal on Scientific Computing, 30 (2008), pp. 1548–1571, https://doi.org/10.1137/070687232.

[96] Y. Maday, *The parareal in time algorithm*, in Substructuring Techniques and Domain Decomposition Methods, F. Magoulès, ed., Stirlingshire, UK, 2010, Saxe-Coburg Publications, pp. 19–44, https://doi.org/10.4203/csets.24.2.

[97] Y. Maday and E. M. Rønquist, *Parallelization in time through tensor-product space–time solvers*, Comptes Rendus Mathématique, 346 (2008), pp. 113 – 118, https://doi.org/10.1016/j.crma.2007.09.012.

[98] S. MahmoudKaber and Y. Maday, *Parareal in time approximation of the Korteveg-deVries-Burgers' equations*, PAMM, 7 (2007), pp. 1026403–1026404, https://doi.org/10.1002/pamm.200700574.

[99] T. A. Manteuffel, S. Münzenmaier, J. Ruge, and B. Southworth, *Nonsymmetric reduction-based algebraic multigrid*, SIAM Journal on Scientific Computing, 41 (2019), pp. S242–S268, https://doi.org/10.1137/18M1193761.

[100] T. A. Manteuffel, J. Ruge, and B. S. Southworth, *Nonsymmetric algebraic multigrid based on local approximate ideal restriction (ℓAIR)*, SIAM Journal on Scientific Computing, 40 (2018), pp. A4105–A4130, https://doi.org/10.1137/17M1144350.

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

[102] *MFEM: Modular finite element methods [software].* mfem.org, https://doi.org/10.11578/dc.20171025.1248.

[103] Z. Miao, Y.-L. Jiang, and Y.-B. Yang, *Convergence analysis of a parareal-in-time algorithm for the incompressible non-isothermal flows*, International Journal of Computer Mathematics, 96 (2019), pp. 1398–1415, https://doi.org/10.1080/00207160.2018.1498484.

[104] M. L. Minion, R. Speck, M. Bolten, M. Emmett, and D. Ruprecht, *Interweaving PFASST and parallel multigrid*, SIAM Journal on Scientific Computing, 37 (2015), pp. S244–S263, https://doi.org/10.1137/14097536X.

[105] A. Mitchell and D. Griffiths, *The Finite Difference Method in Partial Differential Equations*, John Wiley & Sons, Chichester-New York-Brisbane-Toronto, 1980.

[106] K. Miyamoto, *Plasma physics for controlled fusion*, Springer-Verlag, Berlin, Germany, 2nd ed., 2016.

[107] M. Murphy, G. Golub, and A. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1969–1972, https://doi.org/10.1137/S1064827599355153.

[108] M. K. Ng, *Iterative Methods for Toeplitz Systems*, Numerical Mathematics and Scientific Computation, Oxford University Press, UK, 2004.

[109] A. S. Nielsen, G. Brunner, and J. S. Hesthaven, *Communication-aware adaptive Parareal with application to a nonlinear hyperbolic system of partial differential equations*, Journal of computational physics, 15 (2017), pp. 483–505, https://doi.org/10.1016/j.jcp.2018.04.056.

[110] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Communications of the ACM, 7 (1964), pp. 731–733, https://doi.org/10.1145/355588.365137.

[111] B. W. Ong and J. B. Schroder, *Applications of time parallelization*, Computing and Visualization in Science, 23 (2020), https://doi.org/10.1007/s00791-020-00331-4.

[112] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, 12 (1975), pp. 617–629, https://doi.org/10.1137/0712047.

[113] J. W. Pearson, M. Stoll, and A. J. Wathen, *Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems*, SIAM Journal on Matrix Analysis and Applications, 33 (2012), pp. 1126–1152, https://doi.org/10.1137/110847949.

[114] B. Philip, L. Chacón, and M. Pernice, *Implicit adaptive mesh refinement for 2d reduced resistive magnetohydrodynamics*, Journal of Computational Physics, 227 (2008), pp. 8855–8874, https://doi.org/10.1016/j.jcp.2008.06.029.

[115] É. Picard, *Sur l'application des méthodes d'approximations successives à l'étude de certaines équations différentielles ordinaires*, Journal de Mathématiques Pures et Appliquées, 9 (1893), pp. 217–272, https://doi.org/10.2307/2369869.

[116] J. Qiu and C.-W. Shu, *On the construction, comparison, and local characteristic decomposition for high-order central WENO schemes*, Journal of Computational Physics, 183 (2002), pp. 187 – 209, https://doi.org/doi.org/10.1006/jcph.2002.7191.

[117] J.-M. Qiu and C.-W. Shu, *Conservative high order semi-Lagrangian finite difference WENO methods for advection in incompressible flow*, Journal of Computational Physics, 230 (2011), pp. 863–889, https://doi.org/10.1016/j.jcp.2010.04.037.

[118] A. Quarteroni, *Numerical Models for Differential Problems*, MS&A, Springer International Publishing, 2017, https://doi.org/10.1007/978-3-319-49316-9.

[119] J. Reynolds-Barredo, D. Newman, R. Sánchez, D. Samaddar, L. Berry, and W. Elwasif, *Mechanisms for the convergence of time-parallelized, parareal turbulent plasma simulations*, Journal of Computational Physics, 231 (2012), pp. 7851–7867, https://doi.org/10.1016/j.jcp.2012.07.028.

[120] M. Ries, U. Trottenberg, and G. Winter, *A note on MGR methods*, Linear Algebra and its Applications, 49 (1983), pp. 1–26, https://doi.org/doi.org/10.1016/0024-3795(83)90091-5.

[121] J. W. Ruge and K. Stüben, *Algebraic Multigrid*, Society for Industrial and Applied Mathematics, 1987, ch. 4, pp. 73–130, https://doi.org/10.1137/1.9781611971057.ch4.

[122] K. Rupp, *42 years of microprocessor trend data*. https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/. Raw data available at https://github.com/karlrupp/microprocessor-trend-data. Accessed on: 2019/06/19.

[123] D. Ruprecht, *Parareal's discrete dispersion relation*. https://www.birs.ca/events/2016/5-day-workshops/16w5030/videos/watch/201611281030-Ruprecht.html. Presentation at the *Fifth Parallel-in-Time Integration Workshop*. Accessed on: 2021/06/11.

[124] D. Ruprecht, *Wave propagation characteristics of parareal*, Computing and Visualization in Science, 19 (2018), pp. 1–17, https://doi.org/10.1007/s00791-018-0296-z.

[125] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM Journal on Scientific Computing, 14 (1993), pp. 461–469, https://doi.org/10.1137/0914028.

[126] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, 2nd ed., 2003, https://doi.org/10.1137/1.9780898718003.

[127] Y. Saad and M. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, Siam Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869, https://doi.org/10.1137/0907058.

[128] D. Samaddar, D. Coster, X. Bonnin, L. Berry, W. Elwasif, and D. Batchelor, *Application of the parareal algorithm to simulations of elms in iter plasma*, Computer Physics Communications, 235 (2019), pp. 246–257, https://doi.org/10.1016/j.cpc.2018.08.007.

[129] D. Samaddar, D. Newman, and R. Sánchez, *Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm*, Journal of Computational Physics, 229 (2010), pp. 6558–6573, https://doi.org/10.1016/j.jcp.2010.05.012.

[130] W. Schiesser, *The Numerical Method of Lines: Integration of Partial Differential Equations*, Academic Press, 1991.

[131] Schmid-Burgk, *Zweidimensionale selbstkonsistente Lösungen der stationären Wlassowgleichung für Zweikomponententplasmen*, master's thesis, Ludwig-Maximilians-Universität München, 1965.

[132] A. Schmitt, M. Schreiber, P. Peixoto, and M. Schäfer, *A numerical study of a semi-lagrangian Parareal method applied to the viscous Burgers equation*, Computing and Visualization in Science, 19 (2018), pp. 45–57.

[133] D. Schötzau, *Mixed finite element methods for stationary incompressible magneto-hydrodynamics*, Numerische Mathematik, 96 (2004), pp. 771–800.

[134] J. Shadid, R. Pawlowski, J. Banks, L. Chacón, P. Lin, and R. Tuminaro, *Towards a scalable fully-implicit fully-coupled resistive MHD formulation with stabilized FE methods*, Journal of Computational Physics, 229 (2010), pp. 7649–7671, https://doi.org/10.1016/j.jcp.2010.06.018.

[135] J. Shadid, R. Pawlowski, E. Cyr, R. Tuminaro, L. Chacón, and P. Weber, *Scalable implicit incompressible resistive MHD with stabilized fe*

and fully-coupled Newton-Krylov-AMG, Computer Methods in Applied Mechanics and Engineering, 304 (2016), pp. 1–25, https://doi.org/10.1016/j.cma.2016.01.019.

[136] C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 325–432, https://doi.org/10.1007/BFb0096355.

[137] D. Silvester, H. Elman, D. Kay, and A. Wathen, *Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow*, Journal of Computational and Applied Mathematics, 128 (2001), pp. 261 – 279, https://doi.org/10.1016/S0377-0427(00)00515-X. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[138] A. A. Sivas, B. S. Southworth, and S. Rhebergen, *AIR algebraic multigrid for a space-time hybridizable discontinuous Galerkin discretization of advection (-diffusion)*, ArXiv preprint, arXiv:2010.11130 [math.NA], (2020), https://arxiv.org/abs/2010.11130.

[139] B. Southworth, *Necessary conditions and tight two-level convergence bounds for parareal and multigrid reduction in time*, SIAM Journal on Matrix Analysis and Applications, 40 (2019), pp. 564–608, https://doi.org/10.1137/18M1226208.

[140] B. S. Southworth, W. Mitchell, A. Hessenthaler, and F. Danieli, *Tight two-level convergence of linear parareal and MGRIT: Extensions and implications in practice*, ArXiv preprint, arXiv:2010.11879 [math.NA], (2020), https://arxiv.org/abs/2010.11879.

[141] B. S. Southworth, A. A. Sivas, and S. Rhebergen, *On fixed-point, Krylov, and 2×2 block preconditioners for nonsymmetric problems*, SIAM Journal on Matrix Analysis and Applications, 41 (2020), pp. 871–900, https://doi.org/10.1137/19M1298317.

[142] R. Speck and D. Ruprecht, *Toward fault-tolerant parallel-in-time integration with PFASST*, ArXiv preprint, arXiv:1510.08334 [cs.DC], (2015), https://arxiv.org/abs/1510.08334.

[143] G. A. Staff and E. M. Rønquist, *Stability of the parareal algorithm*, in Domain Decomposition Methods in Science and Engineering, 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, eds., Berlin, Heidelberg, 2005, Springer Berlin Heidelberg, pp. 449–456, https://doi.org/10.1007/3-540-26825-1_46.

[144] M. Stoll and A. Wathen, *All-at-once solution of time-dependent Stokes control*, Journal of Computational Physics, 232 (2013), https://doi.org/10.1016/j.jcp.2012.08.039.

[145] G. Strang, *A proposal for Toeplitz matrix calculations*, Studies in Applied Mathematics, 74 (1986), pp. 171–176, https://doi.org/10.1002/sapm1986742171.

[146] P. N. Swarztrauber, *Multiprocessor FFTs*, Parallel Computing, 5 (1987), pp. 197 – 210, https://doi.org/10.1016/0167-8191(87)90018-4. Proceedings of the International Conference on Vector and Parallel Computing - Issues in Applied Research and Development.

[147] A. Toselli and O. Widlund, *Domain Decomposition Methods - Algorithms and Theory*, Springer Series in Computational Mathematics, Springer Berlin Heidelberg, 2006.

[148] J. Trindade and J. Pereira, *Parallel-in-time simulation of the unsteady Navier–Stokes equations for incompressible flow*, International journal for numerical methods in fluids, 45 (2004), pp. 1123–1136, https://doi.org/10.1002/fld.732.

[149] J. Trindade and J. Pereira, *Parallel-in-time simulation of two-dimensional, unsteady, incompressible laminar flows*, Numerical Heat Transfer, Part B: Fundamentals, 50 (2006), pp. 25–40, https://doi.org/10.1080/10407790500459379.

[150] U. Trottenberg, C. Oosterlee, and A. Schuller, *Multigrid*, Elsevier Science, 2000.

[151] UK Atomic Energy Authority, *Culham Centre for Fusion Energy*. https://ccfe.ukaea.uk/. Accessed on: 2021/06/01.

[152] Q. Wang, S. A. Gomez, P. J. Blonigan, A. L. Gregory, and E. Y. Qian, *Towards scalable parallel-in-time turbulent flow simulations*, Physics of Fluids, 25 (2013), p. 110818, https://doi.org/10.1063/1.4819390.

[153] A. Wathen, *Some observations on preconditioning for non-self-adjoint and time-dependent problems*, Computers & Mathematics with Applications, (2021), https://doi.org/10.1016/j.camwa.2021.05.037.

[154] A. Wathen and T. Rees, *Chebyshev semi-iteration in preconditioning for problems including the mass matrix*, Electronic Transactions on Numerical Analysis, 34 (2008-2009), pp. 125–135.

[155] A. J. Wathen, *Realistic eigenvalue bounds for the Galerkin mass matrix*, IMA Journal of Numerical Analysis, 7 (1987), pp. 449–457, https://doi.org/10.1093/imanum/7.4.449.