**Electronic Transactions on Numerical Analysis.**
**Volume 58, pp. 43–65, 2023.**
**Copyright © 2023, Kent State University.**
**ISSN 1068–9613.**
**DOI: 10.1553/etna_vol58s43**

**ETNA**
Kent State University **and**
Johann Radon Institute (RICAM)

# MULTIGRID REDUCTION IN TIME FOR NON-LINEAR HYPERBOLIC EQUATIONS[*]

FEDERICO DANIELI[†] AND SCOTT MACLACHLAN[‡]

**Abstract.** Time-parallel algorithms seek greater concurrency by decomposing the temporal domain of a partial differential equation, providing possibilities for accelerating the computation of its solution. While parallelisation in time has allowed remarkable speed-ups in applications involving parabolic equations, its effectiveness in the hyperbolic framework remains debatable: growth of instabilities and slow convergence are both strong issues in this case, which often negate most of the advantages from time-parallelisation. Here, we focus on the Multigrid Reduction in Time algorithm, investigating in detail its performance when applied to non-linear conservation laws with a variety of discretisation schemes. Specific attention is given to high-accuracy Weighted Essentially Non-Oscillatory reconstructions, coupled with Strong Stability Preserving integrators, which are often the discretisations of choice for such equations. A technique to improve the performance of MGRIT when applied to a low-order, more dissipative scheme is also outlined. This study aims at identifying the main causes for degradation in the convergence speed of the algorithm and finds the Courant-Friedrichs-Lewy limit to be the principal determining factor.

**Key words.** parallel-in-time integration, multigrid, conservation laws, WENO, high-order methods.

**AMS subject classifications.** 65M08, 35L65, 65M55, 65Y05, 65Y20

## 1. Introduction.

The growing complexity of computations arising from the numerical approximation of solutions to *Partial Differential Equations* (PDEs) demands for ever-increasing computational power in order to tackle such problems in a reasonable amount of time. The frequency of computations that can be performed on a single processor unit, however, is limited (see Figure 1.1) and represents an upper bound for the efficiency of serial algorithms. To overcome this limit, much attention has been directed towards the implementation of parallel algorithms, thus providing an alternative to sheer processing power to speed up the computation of such numerical solutions.

Some of the most standard parallelisation methods involve the decomposition of the spatial domain of the target PDE into smaller sub-domains, where some computations can be carried out independently [34]. Information about solutions near interfaces needs to be exchanged between these sub-domains, usually in an iterative fashion. However, this procedure causes an additional overhead cost with respect to the serial solution so that increasing the number of sub-domains above a certain limit becomes detrimental to the purpose of speeding up the computation, and spatial parallelisation is said to *saturate*. For time-dependent PDEs and ODEs, the time domain presents an additional direction along which to seek parallelisation: if efficiently employed, this would allow us to make the most out of the parallel computation capabilities of modern supercomputers.

To achieve time-parallelisation, a variety of schemes have been introduced over the last few decades. Pioneering in this regard was the work by J-L. Lions, Y. Maday, and G. Turinici, who proposed the *Parareal* algorithm in [25], but other methods quickly followed: among these, we point out the *Parallel Full Approximation Scheme in Space and Time* (PFASST [2, 11]) and the *Revisionist Integral Deferred Correction* method (or RIDC [4, 5, 28]), as well as

[*]Received January 17, 2021. Accepted September 29, 2022. Published online on November 11, 2022. Recommended by Stefan Vandewalle. This publication is based on work partially supported by the EPSRC Centre For Doctoral Training in Industrially Focused Mathematical Modelling (EP/L015803/1) in collaboration with the Culham Centre for Fusion Energy, and by the National Productivity Investment Fund (NPIF). The work of S. MacLachlan was partially supported by NSERC discovery grants RGPIN-2014-06032 and RGPIN-2019-05692.

[†]Mathematical Institute, University of Oxford, Oxford, Oxfordshire, UK (danieli@maths.ox.ac.uk).

[‡]Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's, NL, Canada (smaclachlan@mun.ca).

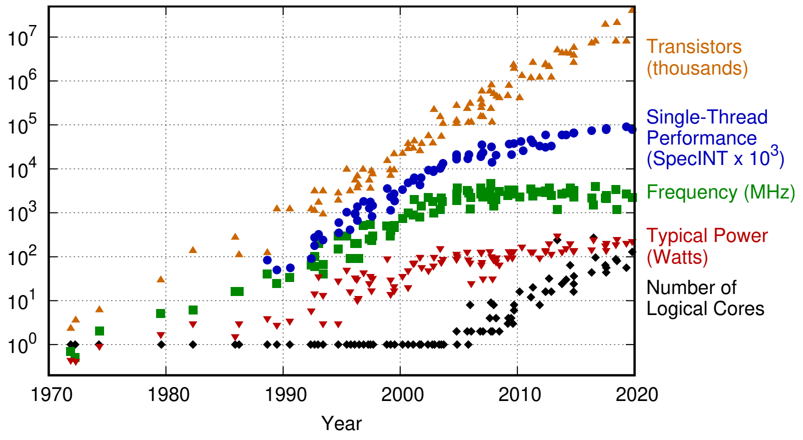


FIG. 1.1. *Evolution of microprocessors performance. In recent years, while the clock frequency of a single processor (green squares) has plateaued, the number of cores per processors (black diamonds) has been steadily increasing, which reflects the focus on parallelisation over power. Source: [31, Creative Commons Attribution 4.0 International Public License].*

the *Multigrid Reduction In Time* algorithm (MGRIT [12, 13, 15]), which is the main focus of this paper. For a more thorough review of parallel-in-time methods, we refer to [16]. Although different in their behaviour, at the core these schemes share a similar idea to achieve parallelisation: that is, pairing a *fine integrator* (which is expensive to use but is applied in parallel) together with a *coarse integrator* (whose action is cheaper to compute but is applied serially). The former is responsible for solving the target equation to the desired level of accuracy, while the latter takes care of quickly propagating updates along the time domain. Time-parallelisation has proven very effective if used to speed up the solution of parabolic equations (see, for example, [15, 23, 29] for an analysis of some such applications from a range of different fields); unfortunately, though, its applicability to advection-dominated or hyperbolic PDEs still remains a matter of research. Currently available algorithms simply do not offer performance comparable to serial-in-time integration approaches for most hyperbolic PDEs, which motivates the current study.

Recovering an accurate numerical solution to hyperbolic equations *per se* is challenging in many ways [24, Chap. 1.4]. One of the biggest difficulties lies in properly capturing shocks present in the solution, while retaining the desired level of accuracy and ensuring stability of the scheme used. These objectives are often in conflict with one another: better accuracy can be achieved using high-order approximations, but these approximations might generate spurious oscillations in the 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 paper. This problem is even more pronounced if time-parallelisation comes into play, when solutions from different integrators need to be combined together. Even if both fine and coarse solver 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. These issues were reported already in [1] and emerge clearly from the analyses in [17] and [32]. Regardless, there remain some examples of successful applications: for example, in [6], stability of Parareal applied to Burgers' equation is guaranteed by projecting the solution back to an energy manifold after every iteration; in [10] the authors succeed in speeding up the solution to a hydrodynamic problem characterised by a Reynolds number of $5 \cdot 10^4$; in [27], choosing a Roe-average Riemann solver seems to be key for fast convergence when solving the shallow-water equations; in [8], an optimisation approach is used to determine coarse-grid operators that achieve excellent performance of MGRIT for the linear advection equation.

In our work, we investigate in detail the performance of MGRIT when applied to some non-linear 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 the knowledge of the authors, somewhat rare in the literature. Noticeable exceptions are: publications involving the PFASST algorithm [11], which achieves high-order accuracy via repeated SDC corrections but is limited to this very specific class of integrators; the works conducted in [13, 14], where $k$-th order multi-step BDF methods are analysed; [27], which shares a similar setup to ours but only shows results for a specific configuration of the integrators used; finally, [8], which is possibly the work whose goal is the closest to ours. 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 non-linear 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 remainder of this article is structured as follows. In Section 2, we present the test problems considered here and the discretisations used in our experiments. In Section 3, we describe the MGRIT algorithm used in our simulations and discuss the important choice of the coarse integrator for a given fine integrator. In Section 4, we present and discuss numerical experiments.

## 2. 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, even though the algorithms introduced in this manuscript can be extended to more general applications. The target equations can be written in the following general form:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \dfrac{\partial \boldsymbol{f}(\boldsymbol{u})}{\partial x} = 0 & (x,t) \in [0,L] \times [0,T]\,, \\ \boldsymbol{u}(x,0) = \bar{\boldsymbol{u}}^0(x) & x \in [0,L]\,, \\ \boldsymbol{u}(0,t) = \boldsymbol{u}(L,t) & t \in [0,T]\,, \end{cases} \tag{2.1}$$

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 initial condition. We consider three different systems, ubiquitous in the literature.

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

  $$\boldsymbol{u} = u, \qquad \boldsymbol{f}(\boldsymbol{u}) = \frac{u^2}{2}, \tag{2.2}$$

  with $u$ representing the flow velocity.

- The *shallow-water equations*, also considered without viscosity, 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,

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

  with $g$ being the gravitational constant and $h$ the height of the fluid column.
- The *Euler equations* are another system governing compressible fluid flow, with

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

  where $\rho$ is the flow density, $E$ 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:

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

These three problems are presented in detail in [24], in 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.

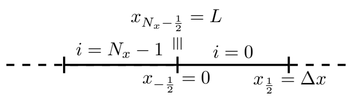
**2.1. Spatial discretisation.** To simplify our notation, in this section we consider the state of the conservation law being a scalar, unless specified otherwise, although the treatment described here can be seamlessly extended to the systems listed above.

Equation (2.1) 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 $\boldsymbol{u}(t) \in \mathbb{R}^{N_x} \approx u(x,t)$, the $i$-th component of which represents the cell-average

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

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

$$\begin{aligned} x_{i-\frac{1}{2}} &= i\Delta x \\ x_{i+\frac{1}{2}} &= (i+1)\Delta x \end{aligned} \qquad i = 0, \dots, N_x - 1,$$



,

where the right-interface of cell $N_x - 1$, (at coordinate $x_{N_x-\frac{1}{2}} = L$), logically coincides with the left-interface of cell 0, (at coordinate $x_{-\frac{1}{2}} = 0$), due to the periodic boundary conditions being applied. Integrating equation (2.1) on cell $i$, we obtain

$$\frac{\partial u_i(t)}{\partial t} = -\frac{f_{i+\frac{1}{2}}(t) - f_{i-\frac{1}{2}}(t)}{\Delta x} =: \left[\mathcal{A}(\boldsymbol{u}(t))\right]_i, \tag{2.3}$$

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

$$f_{i+\frac{1}{2}}(t) \approx f\left(u\left(x_{i+\frac{1}{2}}, t\right)\right). \tag{2.4}$$

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 (2.4). Our experiments are based primarily on the WENO scheme, which is briefly introduced in Section 2.1.1. In Section 2.1.2 and Section 2.1.3, we illustrate the two procedures employed to recover the fluxes in (2.4) using WENO.

**2.1.1. WENO reconstruction.** The *Weighted Essentially Non-Oscillatory* scheme (or WENO) was originally developed by Liu, Chan, and Osher in [26]. In this section, we aim at providing a short description of the main idea behind the method, but we refer to [21, 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

$$q^{\pm}_{i+\frac{1}{2}} \approx q\left(x^{\pm}_{i+\frac{1}{2}}\right). \tag{2.5}$$

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 (2.5). 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 Section 1, solutions to (2.1) 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 the 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 Figure 2.1). Each of these approximations can be expressed

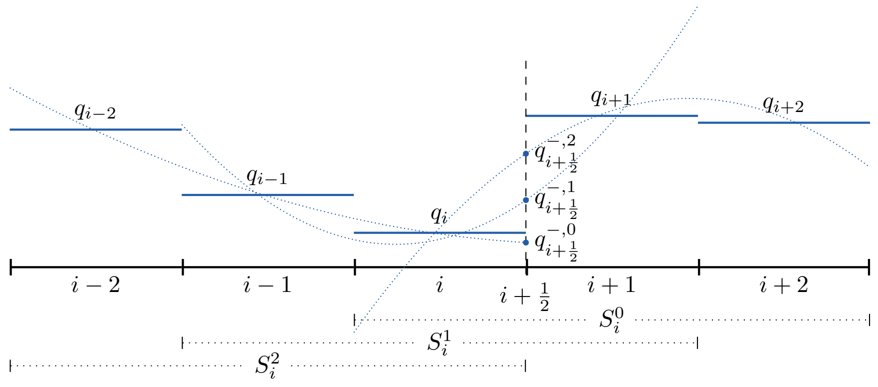


FIG. 2.1. *Choice of the stencils $S^r_i$ for the polynomial reconstructions of degree $k = 2$ of the value left of the interface, $q^{-}_{i+\frac{1}{2}}$, starting from the cell values $q_{i-k}, \ldots, q_{i+k}$ (flat blue lines). Different stencils produce different approximating polynomials $\tilde{q}^r_i(x)$ (dotted blue lines), which in turn give different values at the interface (blue dots).*

as a linear combination of the cell-averages,

$$q^{-,r}_{i+\frac{1}{2}} = \tilde{q}^r_i\left(x_{i+\frac{1}{2}}\right) = \sum_{j\in S^r_i} c_{r,j-(i+r-k)}\, q_j, \tag{2.6}$$

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* $\beta^r_i$, 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,

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

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 $\omega^r_i$ as follows:

$$\omega^r_i = \frac{\alpha^r_i}{\sum_{r=0}^{k}\alpha^r_i}, \qquad \text{and} \qquad \alpha^r_i = \frac{d_r}{(\varepsilon+\beta^r_i)^2}, \tag{2.7}$$

where $\varepsilon$ 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, it holds

$$q^{\pm}_{i+\frac{1}{2}} = \sum_{r=0}^{k} d_r q^{\pm,r}_{i+\frac{1}{2}} = q\left(x_{i+\frac{1}{2}}\right) + \mathcal{O}(\Delta x^s),$$

with $s = 2k+1$. This way, the choice (2.7) of the weights $\omega^r_i$ 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 (2.6),

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

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:

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

$$B_0 = \frac{1}{6}\begin{bmatrix} 20 & -31 & 11 \\ -31 & 50 & -19 \\ 11 & -19 & 8 \end{bmatrix}, \qquad B_1 = \frac{1}{6}\begin{bmatrix} 8 & -13 & 5 \\ -13 & 26 & -13 \\ 5 & -13 & 8 \end{bmatrix},$$

$$B_2 = \frac{1}{6}\begin{bmatrix} 8 & -19 & 11 \\ -19 & 50 & -31 \\ 11 & -31 & 20 \end{bmatrix}.$$

A routine for the evaluation of such parameters for a generic $k$ is described in [21, Example 11.5].

An analogous procedure is used to recover the value on 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 [30]. To limit this effect, a more stable approach consists, rather, of independently reconstructing the *characteristic* variables of the system since these are more readily associated with the information carried by the characteristics [21, 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 [33, 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 decided 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 to the right of each interface, we proceed to approximate the numerical flux (2.4) in two different ways. These are described next.

**2.1.2. Lax-Friedrichs flux.** We first consider the Lax-Friedrichs definition for the numerical flux [33, Chap. 2.3.1]. For a system of conservation laws, this is given by

$$\boldsymbol{f}^{LF}_{i+\frac{1}{2}} = \frac{1}{2}\left(\boldsymbol{f}\left(\boldsymbol{u}^+_{i+\frac{1}{2}}\right) + \boldsymbol{f}\left(\boldsymbol{u}^-_{i+\frac{1}{2}}\right)\right) + \frac{1}{2}\alpha_{LF}\left(\boldsymbol{u}^+_{i+\frac{1}{2}} - \boldsymbol{u}^-_{i+\frac{1}{2}}\right). \tag{2.8}$$

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 $\alpha_{LF}$: this can in general vary with $i$, but in our implementation we consider it constant for simplicity and choose the (somewhat loose) value of

$$\alpha_{LF} = \max_{i,k}\left\{\left|\lambda_k\left(\boldsymbol{u}^\pm_{i+\frac{1}{2}}\right)\right|\right\}, \qquad \begin{array}{l} i = 0, \dots, N_x - 1 \\ k = 0, \dots, D - 1 \end{array},$$

where the $\lambda_k$'s are the $D$ eigenvalues (if the system is composed of $D$ equations) of the Jacobian of the flux:

$$J_{\boldsymbol{f}}\left(\boldsymbol{u}\right)\boldsymbol{r}_k = \lambda_k \boldsymbol{r}_k, \quad \text{with} \quad \left[J_{\boldsymbol{f}}\left(\boldsymbol{u}\right)\right]_{m,n} = \frac{\partial\left[\boldsymbol{f}(\boldsymbol{u})\right]_m}{\partial\left[\boldsymbol{u}\right]_n},$$

while the $\boldsymbol{r}_k$'s represent the corresponding eigenvectors.

**2.1.3. Roe flux.** The second scheme investigated in our experiments uses Roe's approximate Riemann solver [24, Chap. 14.2] in order to recover the numerical flux. For each interface, this solver targets a linearisation of the Riemann problem defined by (2.1) and the left and right values computed using the WENO procedure. This problem is written as:

$$\boldsymbol{u}_t + J_{\boldsymbol{f}}(\hat{\boldsymbol{u}})\,\boldsymbol{u}_x = 0, \tag{2.9}$$

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 (2.9), 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

$$\begin{aligned} \lambda_0^S &= u - c_S \\ \lambda_1^S &= u + c_S \end{aligned} \qquad \text{and} \qquad \begin{aligned} \boldsymbol{r}_0^S &= \begin{bmatrix} 1, & \lambda_0^S \end{bmatrix}^T \\ \boldsymbol{r}_1^S &= \begin{bmatrix} 1, & \lambda_1^S \end{bmatrix}^T \end{aligned}$$

for the shallow-water system, with *speed of sound* $c_S = \sqrt{gh}$, and by

$$\begin{aligned} \lambda_0^E &= u - c_E \\ \lambda_1^E &= u \\ \lambda_2^E &= u + c_E \end{aligned} \qquad \text{and} \qquad \begin{aligned} \boldsymbol{r}_0^E &= \begin{bmatrix} 1, & \lambda_0^E, & H - uc_E \end{bmatrix}^T \\ \boldsymbol{r}_1^E &= \begin{bmatrix} 1, & \lambda_1^E, & u^2/2 \end{bmatrix}^T \\ \boldsymbol{r}_2^E &= \begin{bmatrix} 1, & \lambda_2^E, & H + uc_E \end{bmatrix}^T \end{aligned}$$

for the Euler equations, with speed of sound $c_E = \sqrt{(\gamma-1)(H-u^2/2)}$ and $H = (E+p)/\rho$ 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 (2.9), (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

$$\hat{u} = \frac{u^+ + u^-}{2};$$

an average height and velocity for the Shallow-water equation prescribed as

$$\hat{h} = \frac{h^+ + h^-}{2} \qquad \text{and} \qquad \hat{u} = \frac{u^+\sqrt{h^+} + u^-\sqrt{h^-}}{\sqrt{h^+} + \sqrt{h^-}};$$

and finally, an average velocity and specific enthalpy for the Euler equations:

$$\hat{u} = \frac{u^+\sqrt{\rho^+} + u^-\sqrt{\rho^-}}{\sqrt{\rho^+} + \sqrt{\rho^-}} \qquad \text{and} \qquad \hat{H} = \frac{(E^+ + p^+)\sqrt{\rho^+} + (E^- + p^-)\sqrt{\rho^-}}{\sqrt{\rho^+} + \sqrt{\rho^-}}.$$

Since the target problem (2.9) is linear, this procedure is known to provide a non-entropic weak solution when transonic rarefaction waves arise [24, Chap. 14.2.2]. To counteract this, we also apply the *entropy fix* proposed by Harten and Hyman in [19]. 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:

$$\boldsymbol{f}^R = \frac{1}{2}\left(\boldsymbol{f}\left(\boldsymbol{u}^-\right) + \boldsymbol{f}\left(\boldsymbol{u}^+\right)\right) - \sum_{k=0}^{D-1} q_H(\hat{\lambda}_k)\,\alpha_k\,\hat{\boldsymbol{r}}_k. \tag{2.10}$$

Here, $\alpha_k$ is the *shock strength*, that is, the jump in the $k$-th characteristic variable between the states left and right of the interface: $\alpha_k = (\boldsymbol{u}^+ - \boldsymbol{u}^-)\cdot\hat{\boldsymbol{r}}_k$. Also, $q_H(\hat{\lambda}_k)$ defines the actual entropy fix:

$$q_H(\hat{\lambda}_k) = \begin{cases} \frac{1}{2}\left(\frac{\hat{\lambda}_k}{\delta_k} + \delta_k\right) & \text{if} \quad |\hat{\lambda}_k| < \delta_k \\ |\hat{\lambda}_k| & \text{else} \end{cases},$$

where $\delta_k = \max\left\{0, \hat{\lambda}_k - \lambda_k^-, \lambda_k^+ - \hat{\lambda}_k\right\}$ and $\lambda_k^\pm$ are the eigenvalues of $J_{\boldsymbol{f}}(\boldsymbol{u}^\pm)$.

Substituting (2.8) or (2.10) into (2.3) 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.

**2.2. Temporal 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 $t_n = n\Delta t$, $n = 0, \dots, N_t - 1$ 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 [18] 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 (2.3) is given by

$$\begin{aligned}
\boldsymbol{u}^{n,(1)} &= \boldsymbol{u}^n + \Delta t\, \mathcal{A}\left(\boldsymbol{u}^n\right) \\
\boldsymbol{u}^{n,(2)} &= \frac{3}{4}\boldsymbol{u}^n + \frac{1}{4}\boldsymbol{u}^{n,(1)} + \frac{1}{4}\Delta t\, \mathcal{A}\left(\boldsymbol{u}^{n,(1)}\right) \\
\boldsymbol{u}^{n+1} &= \frac{1}{3}\boldsymbol{u}^n + \frac{2}{3}\boldsymbol{u}^{n,(2)} + \frac{2}{3}\Delta t\, \mathcal{A}\left(\boldsymbol{u}^{n,(2)}\right);
\end{aligned} \tag{2.11}$$

a second-order scheme (SSPRK2), defined by

$$\begin{aligned}
\boldsymbol{u}^{n,(1)} &= \boldsymbol{u}^n + \Delta t\, \mathcal{A}\left(\boldsymbol{u}^n\right) \\
\boldsymbol{u}^{n+1} &= \frac{1}{2}\boldsymbol{u}^n + \frac{1}{2}\boldsymbol{u}^{n,(1)} + \frac{1}{2}\Delta t\, \mathcal{A}\left(\boldsymbol{u}^{n,(1)}\right);
\end{aligned}$$

and, finally, the first-order scheme, which reduces to the well-known Forward Euler (FE) method,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t\, \mathcal{A}\left(\boldsymbol{u}^n\right).$$

## 3. MGRIT.

The Multigrid Reduction In Time method can be interpreted as a multigrid scheme with the coarsening procedure acting along the time domain. First introduced in [12], it has quickly become a mature algorithm for time-parallelisation. In this section, we provide a brief description of the scheme.

**3.1. Method overview.** As the name suggests, MGRIT is, at its root, a multigrid reduction technique applied to the monolithic system arising from the space-time discretisation of a PDE such as (2.1). In the constant-coefficient, linear, homogeneous case with a single-step time integrator, such a system takes the form of a block bi-diagonal, block Toeplitz matrix:

$$A\boldsymbol{u} = \boldsymbol{g} \Leftrightarrow \begin{bmatrix} I & & & \\ -\Phi & I & & \\ & \ddots & \ddots & \\ & & -\Phi & I \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^0 \\ \boldsymbol{u}^1 \\ \vdots \\ \boldsymbol{u}^{N_t-1} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{u}}^0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \tag{3.1}$$

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 consider only periodic boundary conditions and forcing terms are not present, the rest of $\boldsymbol{g}$ is filled with zeros. The *fine integrator* $\Phi$ represents the action of the time-stepper, so that, for each instant $n$, we have $\boldsymbol{u}^{n+1} = \Phi\boldsymbol{u}^n$. Following the multigrid philosophy, the nodes composing the temporal discretisation are split into two sets, respectively, of *coarse* (denoted

ETNA
Kent State University and
Johann Radon Institute (RICAM)

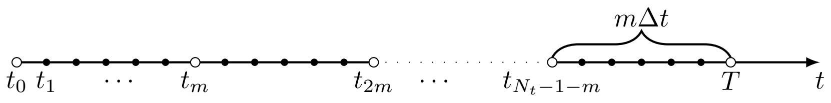


FIG. 3.1. *Sketch of the partitioning of the time discretisation: white dots represent coarse nodes, black dots are fine nodes. The last time chunk is also highlighted.*

with $C$) and *fine* (denoted with $F$) nodes. In our case, we simply pick the coarse nodes to be spaced every $m$ nodes, with $m$ being the *coarsening factor* (see Figure 3.1). The coarse nodes thus subdivide the time domain into $(N_t-1)/m$ different *time chunks*, each of size $m\Delta t$. The variables in the monolithic space-time discretisation $\boldsymbol{u}$ and the coefficients in $A$ can also be rearranged accordingly, so that the matrix can be factorised in the following way:

$$A=\begin{bmatrix}A_{FF} & A_{FC}\\ A_{CF} & A_{CC}\end{bmatrix}=\begin{bmatrix}I_F & 0\\ A_{CF}A_{FF}^{-1} & I_C\end{bmatrix}\begin{bmatrix}A_{FF} & 0\\ 0 & S\end{bmatrix}\begin{bmatrix}I_F & A_{FF}^{-1}A_{FC}\\ 0 & I_C\end{bmatrix}, \tag{3.2}$$

where $I_C$ and $I_F$ are identity matrices of appropriate sizes. This factorisation allows us to separate the solution over the $F$ and the $C$ nodes. It can be easily shown that the $A_{FF}$ sub-matrix in (3.2) presents the following block-diagonal structure, which implies that systems involving it can be readily solved in parallel:

$$A_{FF}=\begin{bmatrix}B_\Phi & & \\ & \ddots & \\ & & B_\Phi\end{bmatrix}, \quad B_\Phi=\begin{bmatrix}I & & & \\ -\Phi & I & & \\ & \ddots & \ddots & \\ & & -\Phi & I\end{bmatrix}. \tag{3.3}$$

Here, 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 $B_\Phi$ of size $(m-1)N_x\times(m-1)N_x$. Each solve with $B_\Phi$ involves "stepping" forward an approximate solution on the spatial mesh through $m-1$ applications of $\Phi$, where the first application of $\Phi$ comes from the calculation of the right-hand side for the $F$-update, which solves $A_{FF}\boldsymbol{u}^F=\boldsymbol{g}^F-A_{FC}\boldsymbol{u}^C$ for $\boldsymbol{u}^F$ given $\boldsymbol{u}^C$. Each block row of $A_{FC}$ contains a single entry of $\Phi$ in the column corresponding to the preceding $C$ point to a block of $F$ points. Each block of this solve is completely independent, and thus, the block solves can be performed in parallel. The challenge lies rather in finding the solution to the system associated with the Schur complement:

$$S=A_{CC}-A_{CF}A_{FF}^{-1}A_{FC}=\begin{bmatrix}I & & & \\ -\Phi^m & I & & \\ & \ddots & \ddots & \\ & & -\Phi^m & I\end{bmatrix}. \tag{3.4}$$

Attempting to solve this directly is equivalent to applying forward block-substitution to (3.1), (i.e., time-stepping over all temporal nodes) and would, thus, nullify any advantage gained from parallelisation. Rather, in the scope of the MGRIT algorithm, we resort to solving a modified system obtained by substituting another operator $\Psi\approx\Phi^m$ in (3.4), giving

$$S_\Psi=\begin{bmatrix}I & & & \\ -\Psi & I & & \\ & \ddots & \ddots & \\ & & -\Psi & I\end{bmatrix}\approx S. \tag{3.5}$$

The structure of this system is equivalent to that of (3.1), so that $\Psi$ 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 nodes can be applied in a recursive fashion, further extracting a hierarchy of $N_l$ grids, together with their corresponding integrators $\Phi_{(l)}$, in a true *multi*-grid spirit.

The hope is that, by opportunely alternating between solving for the fine nodes (inverting $A_{FF}$) and for the coarse nodes (inverting $S_\Psi$) and iterating, we can quickly converge to a suitable approximation of the solution of the original system (3.1).

**3.2. The algorithm.** In more detail, an iteration of the MGRIT algorithm consists of the following building blocks:

*Relaxation.* Update the solution at the fine nodes of the current level given a guess at the coarse nodes. This involves solving a system of the form

$$A^{(l)}_{FF}\boldsymbol{u}^F_{(l)} = \boldsymbol{g}^F_{(l)} - A^{(l)}_{FC}\boldsymbol{u}^C_{(l)}, \tag{3.6}$$

where $A^{(l)}_{FF}$ has the same structure as in (3.3) but whose subdiagonal blocks contain $\Phi_{(l)}$, while $\boldsymbol{u}^F_{(l)}$ and $\boldsymbol{u}^C_{(l)}$, 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 $\Phi_{(l)}$, 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 two types of relaxation: *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 then overwrites the previously updated values on the following time chunk (adding another F-relaxation). This is an *overlapping* form of relaxation that requires more work per level of the hierarchy but can be implemented with the same parallel efficiency.

*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:

$$R^{(l+1)}_{(l)}\boldsymbol{u}_{(l)} = \boldsymbol{u}^C_{(l)}. \tag{3.7}$$

Since we are dealing with non-linear equations, we also implement the *Full Approximation Storage* (FAS) algorithm [3, Chap. 5.3.4]. This modifies the right-hand side of the system at the coarse level by adding a correction term:

$$S_{\Phi_{(l+1)}}\left(\boldsymbol{u}_{(l+1)}\right) = R^{(l+1)}_{(l)}\left(\boldsymbol{g}_{(l)} - S_{\Phi_{(l)}}\left(\boldsymbol{u}_{(l)}\right)\right) + \underbrace{S_{\phi_{(l+1)}}\left(R^{(l+1)}_{(l)}\boldsymbol{u}_{(l)}\right)}_{\text{FAS correction}} = \boldsymbol{g}_{(l+1)}, \tag{3.8}$$

where $S_{\Phi_{(l+1)}}$ has the same structure as in (3.5) but with $\Phi_{(l+1)}$ on the subdiagonal. We point out that the same restriction operator (3.7) is applied to both the residual and the state, which is justified by the structure of the temporal integrator. Given the particular structure of the operators involved, formula (3.8) simplifies so that the right-hand side of the coarse-level system $\boldsymbol{g}_{(l+1)}$ reduces to

$$\boldsymbol{g}^i_{(l+1)} = \boldsymbol{g}^{mi}_{(l)} + \Phi_{(l)}\left(\boldsymbol{u}^{mi-1}_{(l)}\right) - \Phi_{(l+1)}\left(\boldsymbol{u}^{m(i-1)}_{(l)}\right), \tag{3.9}$$

for each temporal node $i = 1, \ldots, (N_t - 1)/m^{l+1} + 1$ 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

$\boldsymbol{u}_{(l)}^{C}$, however we have found that a better alternative is given by

$$\boldsymbol{u}_{(l+1)}^{i} = \Phi_{(l)}\left(\boldsymbol{u}_{(l)}^{mi-1}\right) + \boldsymbol{g}_{(l)}^{mi}, \tag{3.10}$$

that is, by performing the last integration step for each chunk and injecting the resulting value. This operation comes at virtually no additional cost (since the quantity in (3.10) is already computed as part of (3.9)), and in early experiments it was seen to improve convergence: in Figure 4.3, we give an example of the effectiveness of (3.10) over simple injection. Notice that this procedure is *not* equivalent to performing an additional C-relaxation before injection since the FAS correction (3.8) is still based on the non-updated values at the coarse nodes $\boldsymbol{u}_{(l)}^{m(i-1)}$; however, this approach *does* retain the fixed-point property of the FAS algorithm since (3.7) and (3.10) coincide when $\boldsymbol{u}_{(l)}$ is the exact solution on level $l$.

*Coarse grid correction.* Update the solution at the coarsest level. This involves solving the system

$$S_{\Phi_{(N_l-1)}}\boldsymbol{u}_{(N_l-1)} = \boldsymbol{g}_{(N_l-1)}. \tag{3.11}$$

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 the information from the nodes at the coarser level $l+1$ to those at the current level $l$. *Ideal interpolation* is chosen, which in our case consists of two steps: an injection from the coarse to the fine grid, followed by an F-relaxation on the fine level, starting from the freshly updated values at the coarse nodes. Overall then we have the following definition for the interpolation operator:

$$\begin{aligned} &\left[I_{(l+1)}^{(l)}\left(\boldsymbol{u}_{(l+1)}\right)\right]^{i} = F_{(l)}^{\left\lfloor\frac{i}{m}\right\rfloor, i\%m}\left(\boldsymbol{u}_{(l+1)}^{\left\lfloor\frac{i}{m}\right\rfloor}\right), \quad \text{with} \\ &F_{(l)}^{k,r}(\boldsymbol{u}) = \begin{cases} \boldsymbol{u} & \text{if } r=0, \\ \Phi_{(l)}\left(F_{(l)}^{k,r-1}(\boldsymbol{u})\right) + \boldsymbol{g}_{(l)}^{k+r} & \text{else,} \end{cases} \end{aligned} \tag{3.12}$$

where $\%$ represents the *modulo* operator while $\lfloor * \rfloor$ denotes the *floor* operator. Notice that, if the interpolation step is followed by a relaxation step, then the F-relaxation within interpolation

**Algorithm 1:** MGRIT, V-cycle iteration.

```
1 if $l < N_l - 1$ then
      /* Parallel step                                              */
2     Perform relaxation at current level and recover $\boldsymbol{u}_{(l)}^{F}$ by solving (3.6);
3     Compute residual and FAS correction, and restrict to coarser level to compute
        $\boldsymbol{g}_{(l+1)}$ and $\boldsymbol{u}_{(l+1)}$ following formulae (3.9) and (3.10);
      /* Recursive step                                             */
4     Invoke this function at coarser level, and find $\boldsymbol{u}_{(l+1)}$;
      /* Parallel step                                              */
5     Interpolate from coarser level to recover $\boldsymbol{u}_{(l)}$ applying (3.12);
6 else
      /* Serial step - base step of recursion                       */
7     Recover $\boldsymbol{u}_{(N_l-1)}$ by solving system (3.11);
```

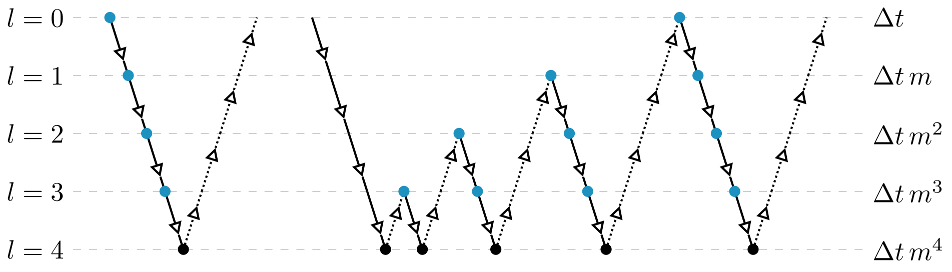


FIG. 3.2. *Sketch of the flow of the MGRIT algorithm if prescribing a V-cycle (left) or an F-cycle (right) on 5 levels. Restriction (full lines) is applied when moving from a level to a coarser one (dashed lines: the size of the time-step at each level l is reported on the right). Relaxation (blue dots) 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.*

becomes redundant since the results would get overwritten by the relaxation step. Here we consider only pre-relaxation within the multigrid cycles and write the interpolation step as an injection followed by an F-relaxation step.

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* [3, Chap. 3]. In Algorithm 1, 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 Figure 3.2.

**3.3. 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 (3.5) represents a valid approximation to (3.4); at the same time, it should be cheap enough to compute so that the solution to systems involving (3.5) can be promptly recovered. The necessity of overcoming this trade-off between cost and accuracy is common to many other time-parallelisation algorithms, and a number of approaches have been proposed to address this. The most straightforward is simple rediscretisation: the coarse integrator is none other than the fine integrator applied on a coarser time grid [12, 15]; an alternative consists in choosing integrators of varying levels of accuracy [27]; yet another lies in neglecting or simplifying certain physics at the coarse level when timescale separation is possible [20]. These solutions focus mostly on finding a coarse integrator which is cheap to compute, but it is unclear to what extent it remains faithful to the fine integrator. Indeed, in the hyperbolic framework, simple rediscretisation has been shown to fail in many cases (see Figure 4.1), particularly if low-order time discretisations are employed: evidence of this is given in [8] as well as in the results of Figure 4.4, at least for certain regimes. When sticking to simple time-steppers, additional care is needed to ensure the stability of MGRIT. To address this, we propose an 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.

**3.3.1. Fine integrator matching.** To illustrate our approach, we consider our model equation (2.2). If a simple Lax-Friedrichs flux discretisation is applied in conjunction with a

Forward Euler time discretisation, then the resulting time-stepping formula reads

$$
\begin{aligned}
u_i^{n+1} &= \underbrace{\frac{u_{i+1}^n + u_{i-1}^n}{2}}_{} - \Delta t \underbrace{\frac{f(u_{i+1}^n) - f(u_{i-1}^n)}{2\Delta x}}_{} \\
&= \quad E u_i^n \quad - \Delta t \quad D f_i^n,
\end{aligned}
\tag{3.13}
$$

with $f_i^n = f(u_i^n)$ and the *central difference* operator $D((*)_i) = ((*)_{i+1} - (*)_{i-1})/2\Delta x$ and the *average* operator $E((*)_i) = ((*)_{i+1} + (*)_{i-1})/2$. If we were to simply rediscretise at the coarse level and apply the same scheme to a coarser grid with time step $2\Delta t$, then the resulting formula would similarly read

$$
u_i^{n+1} = E u_i^n - 2\Delta t D f_i^n. \tag{3.14}
$$

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 (3.13) twice (computing the non-linear analogue of $\Phi^2$, as in (3.4) with $m = 2$), which results in

$$
\begin{aligned}
u_i^{n+2} =& E(E u_i^n - \Delta t D f_i^n) - \Delta t D f(E u_i^n - \Delta t D f i^n) \\
\approx& E^2 u_i^n - \Delta t \left( E D f_i^n + D f(E u_i^n) \right) \\
&+ \Delta t^2 D \left( f'(E u_i^n) D f_i^n \right) - \Delta t^3 D \left( f''(E u_i^n)(D f_i^n)^2 \right).
\end{aligned}
\tag{3.15}
$$

Here we Taylor-expanded the flux in the second term around $E(u_i^n)$, exploiting the fact that $\Delta t$ is a small parameter. In the case of (2.2), this formula is exact since the flux $f$ is a polynomial of degree 2. By directly comparing (3.15) and (3.14), 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 (3.14) so that it matches (3.15) up to a certain order of $\Delta t$. This gives rise to the following integrators:

$$
u_i^{n+1} = \left[ \Psi^{(0)}(\boldsymbol{u}^n) \right]_i := E^2 u_i^n - 2\Delta t D f_i^n, \tag{3.16}
$$

if we aim for a zeroth-order match, leaving the rest untouched, or

$$
u_i^{n+1} = \left[ \Psi^{(1)}(\boldsymbol{u}^n) \right]_i := E^2 u_i^n - \Delta t \left( E D f_i^n + D f(E u_i^n) \right), \tag{3.17}
$$

for a first-order match. Formula (3.15) refers to a coarsening factor $m = 2$, but this can be easily extended to any value of $m > 1$ as follows:

$$
u_i^{n+m} = E^m u_i^n - \Delta t \sum_{i=1}^{m} E^{i-1} D f(E^{m-i} u_i^n) + \mathcal{O}(\Delta t^2), \tag{3.18}
$$

and the corresponding (3.16) and (3.17) can be modified accordingly

REMARK 3.1. We point out that (3.16) and (3.17) can be interpreted as stand-alone integrators. By conducting an analysis similar to that in [24, Chap. 11.1], one can show that the zeroth-order matching scheme (3.16) introduces additional numerical diffusion (increasing with $m$) with respect to (3.13). The first-order matching scheme (3.17) instead differs from (3.16) in that it prescribes a different degree of numerical dispersion.

Unfortunately, the fine-integrator matching approach described in this section presents a number of downsides. First of all, it carries an increased computational cost with respect to

**ETNA**
**Kent State University and**
**Johann Radon Institute (RICAM)**

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 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 (3.18): 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 (3.13) corresponds to the Lax-Friedrichs flux (2.8) 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 Section 4 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.

## 4. 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 [7].

*Fine integrator matching.* As a first test, we check the effectiveness of the time-stepper proposed in Section 3.3.1 applied to Burgers' equation: the convergence results for a number of runs are reported in Figure 4.1. 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 Figure 4.1 (first-order matching, (3.17)) lie consistently below the green ones (zeroth-order matching, (3.16)). This is in line with expectations, since we are employing a more accurate solver. This convergence behaviour remains substantially unchanged as we further refine our grids, as shown by the fact that the different blue graphs superimpose almost perfectly.

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 even gets slightly better initial error reductions in the moving-shock scenario than for the stationary shock. Unfortunately, though, we observe that the error associated with this scheme starts rising again after 5-10 iterations and progressively earlier as we refine the grids further. This growth is caused by the amplification of high-frequency modes which are not effectively damped by the zeroth-order matching method, as shown in more detail in the spectral analysis of the errors illustrated in Figure 4.2. There, we can see that for the very first iteration, the low-frequency modes of the error dominate with many components having magnitude $\sim 10^{-4}$, while the high-frequency components are basically negligible, of magnitude $\sim 10^{-10}$. Scheme (3.16) manages to progressively shrink the former, while the latter begin to grow until the low- and high-frequencies reach a comparable magnitude of $\sim 10^{-8}$ around iteration 10. By iteration 20, the low-frequency modes have essentially disappeared, but the high-frequency modes have reached a magnitude of $\sim 10^{-7}$. This negative effect seems to be heightened as we refine the grid further, with the error starting to increase earlier. Nonetheless, we point out that the fine-integrator matching approach presented in Section 3.3.1 represents a drastic improvement over naïve rediscretisation, for

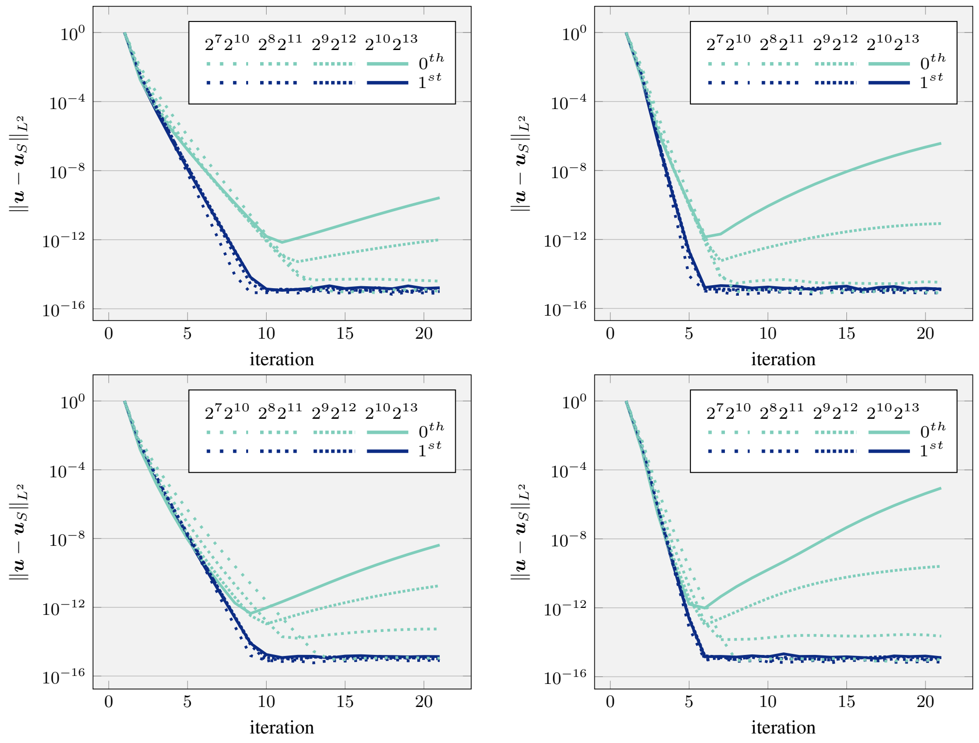


FIG. 4.1. *Impact of grid refinement and degree of fine-integrator matching on the evolution of the error between the MGRIT iterates and the serial solution $\boldsymbol{u}_S$. The relative $L^2$-norm in space-time is considered. MGRIT parameters: F-smoothing, 5 levels, coarsening factor $m = 2$, V-cycle (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 Section 3.3.1 are employed: FE with Lax-Friedrichs flux (3.13) for the finest level and the corresponding zeroth- (3.16) and first-order matching (3.17), 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. Data generated with* `testMGRIT_Burgers_Matching.m` *[7].*

which the error blows up already after the very first iteration (and is, hence, not reported in Figure 4.1). Such behaviour is caused by a similar unchecked growth in the high-frequency modes, except in this case it occurs much more quickly than for (3.16). The fact that the first-order matching scheme (3.16) seems to properly prevent the growth of such modes could be explained by the different dispersion relation it prescribes with respect to (3.16), as pointed out in Remark 3.1. This is in line with the findings of [32], which underline the impact of phase errors between fine and coarse solvers on the convergence of Parareal.

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 changing 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 (3.10). An example of this is shown in Figure 4.3, where a comparison of the error evolution of MGRIT is shown when using simple injection as a restriction operator versus formula (3.10). 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.

Rather, the factor playing the main role in determining the performance of the algorithm lies in how faithfully the *Courant-Friedrichs-Lewy* CFL condition [24, Chap. 10.6] is respected. In Figure 4.1, the number of nodes $N_t$ and $N_x$ is chosen to guarantee that the CFL number on

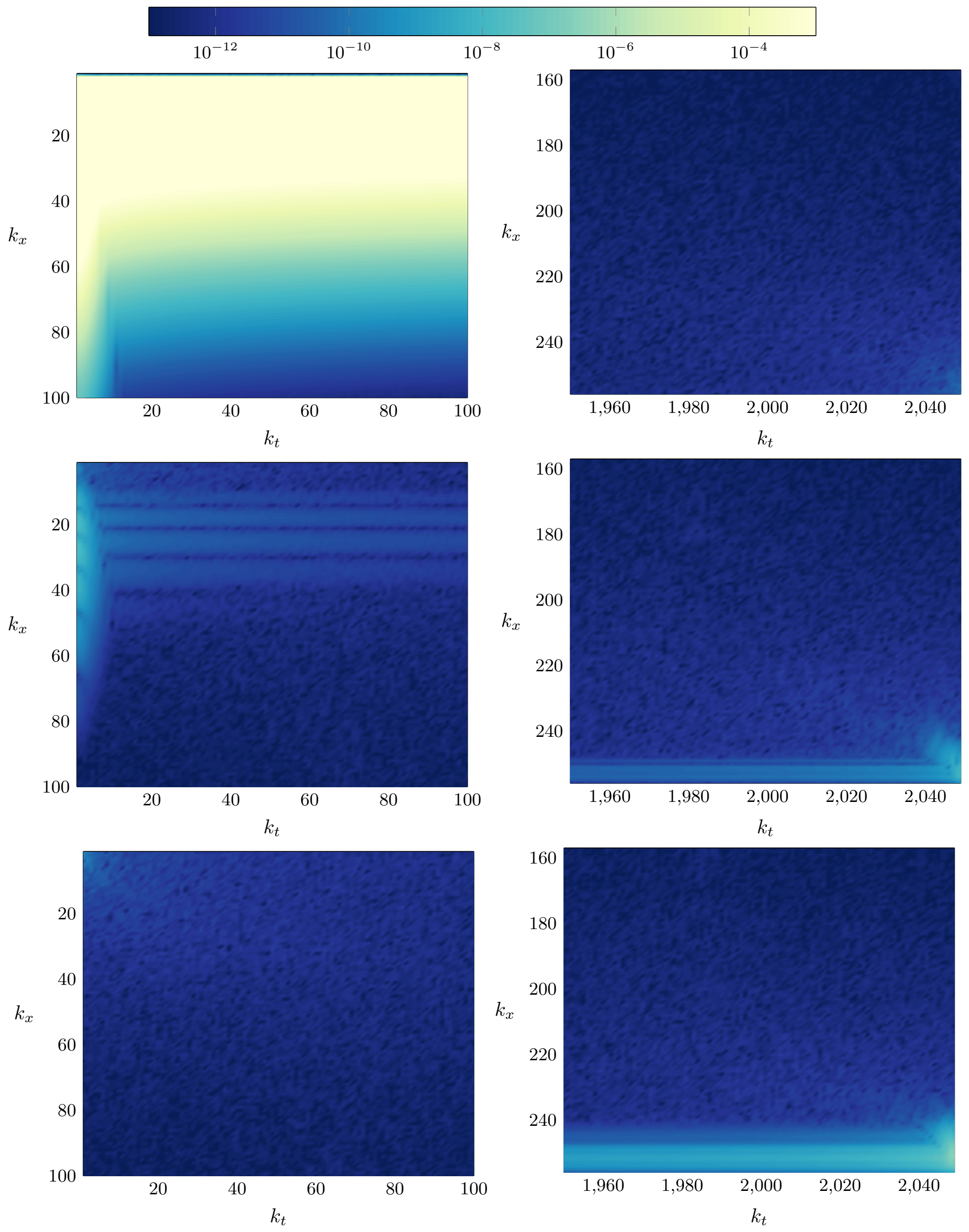


FIG. 4.2. *Evolution of the magnitude of the error modes for the zeroth-order fine-integrator matching scheme* (3.16). *The data in the heatmaps is recovered by applying a 2D discrete Fourier transform to the space-time error at iterations* 1 *(top),* 10 *(middle) and* 20 *(bottom). The various error modes are identified by their spatial and temporal wave numbers* $k_x$ *and* $k_t$*: low frequencies (both in space and time) are reported on the left, while high frequencies are on the right. The plots refer to the experiment in the top-left graph in Figure* 4.1 *with* $N_x N_t = 2^9 2^{12}$.

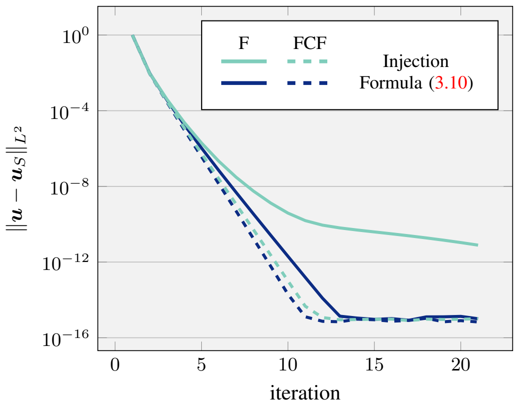


FIG. 4.3. *Effect of modifying the initial guess for the coarse solution on the error convergence of MGRIT: comparison of error convergence between classical injection (green) and formula* (3.10) *(blue). Relaxation used: F-relaxation (full lines) and FCF-relaxation (dashed lines); otherwise, same parameters as in the top-left graph in Figure 4.1. Only results for $N_xN_t = 2^72^{10}$ (zeroth-order) are reported, but finer grids give comparable results.*

the *coarsest* temporal mesh is $\sim 0.95$ in all cases. This also implies that the solution might be over-resolved, since usually one aims for a CFL number of $\sim 1$ at the *finest* level. The impact of relaxing the coarse-grid CFL condition can be clearly seen in the top-left graph of Figure 4.4: convergence degrades noticeably, even using high-order matching, although we still see convergence as $c$ gets larger (noting $c = 7.6$ corresponds to a finest-grid CFL number of 0.475).

*High-order schemes.* While the integrator proposed in Section 3.3.1 seems effective, it remains very dissipative. On the one hand, it is well-documented (see for example [9]) that smoothing effects improve the 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 apply MGRIT to the test cases introduced in Section 2, for discretisations using the Lax-Friedrichs and Roe definitions of numerical fluxes, (2.8) and (2.10), and employing WENO reconstructions at various orders of accuracy $s$. The initial conditions for the various problems are picked to ensure that the solutions achieve a similar maximum CFL number $c \sim 0.43$. These are:

$$\bar{u}_B^0 = \frac{4}{3}\sin\left(\frac{2\pi x}{L}\right), \qquad \bar{\boldsymbol{u}}_S^0 = \frac{1}{11}\left[1 + \tfrac{1}{2}\sin\left(\tfrac{2\pi x}{L}\right), \quad 0\right]^T,$$
$$\bar{\boldsymbol{u}}_E^0 = \left[1, \quad 0, \quad 1 + \tfrac{1}{2}\sin\left(\tfrac{2\pi x}{L}\right)\right]^T,$$

for Burgers', shallow-water, and Euler equations, respectively. From Figure 4.5, 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 completely disregards 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 for performance is to 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

**ETNA**
**Kent State University and**
**Johann Radon Institute (RICAM)**

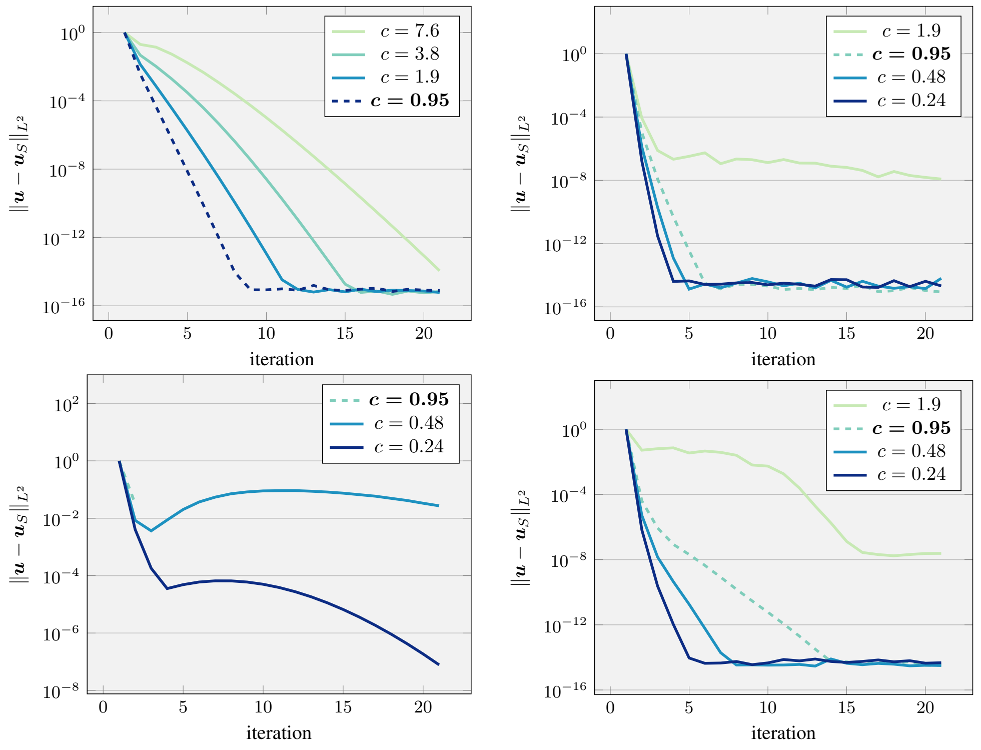


FIG. 4.4. *Influence of increasing the CFL number c at the coarsest level on the convergence behaviour of MGRIT applied to Burgers' equation. $N_x = 2^7$, while $N_t$ is progressively halved to increase c. The other parameters are the same as in the top-left graph in Figure 4.1, but different schemes 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). Data generated using* `testMGRIT_highOrderCFL.m` *[7].*

unclear. Nonetheless, for some values of $c$, these higher-order methods still behave reasonably well, particularly in comparison with many results in the literature regarding more diffusive schemes. Evidence of this is also given in Figure 4.4: at least for $c \leq 0.95$, 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 (3.17). Indeed it is the 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 Figure 4.4. In particular, note how MGRIT (with rediscretisation on the coarse grid) fails to produce a converging solver for $c = 0.95$ (or even $c = 0.48$) in the bottom-left of Figure 4.4, where it is used in combination with a high-order spatial discretisation and a low-order temporal discretisation.

Ultimately, however, increasing the order of the time-discretisation employed can only help us so far in recovering convergence since also for high-order solvers, satisfying the CFL condition remains the true bottleneck in determining their effectiveness. Indeed, looking at the plots in Figure 4.4, we can see that high-order schemes are much more sensitive to an increase in $c$. This clearly highlights that, if explicit solvers are to be used, limiting ourselves to coarsening just in time and employing coarse integrators based on rediscretisation is not a viable option: an effective algorithm needs to pair coarsening in space and time together in order to keep the CFL number close to 1 at each level.

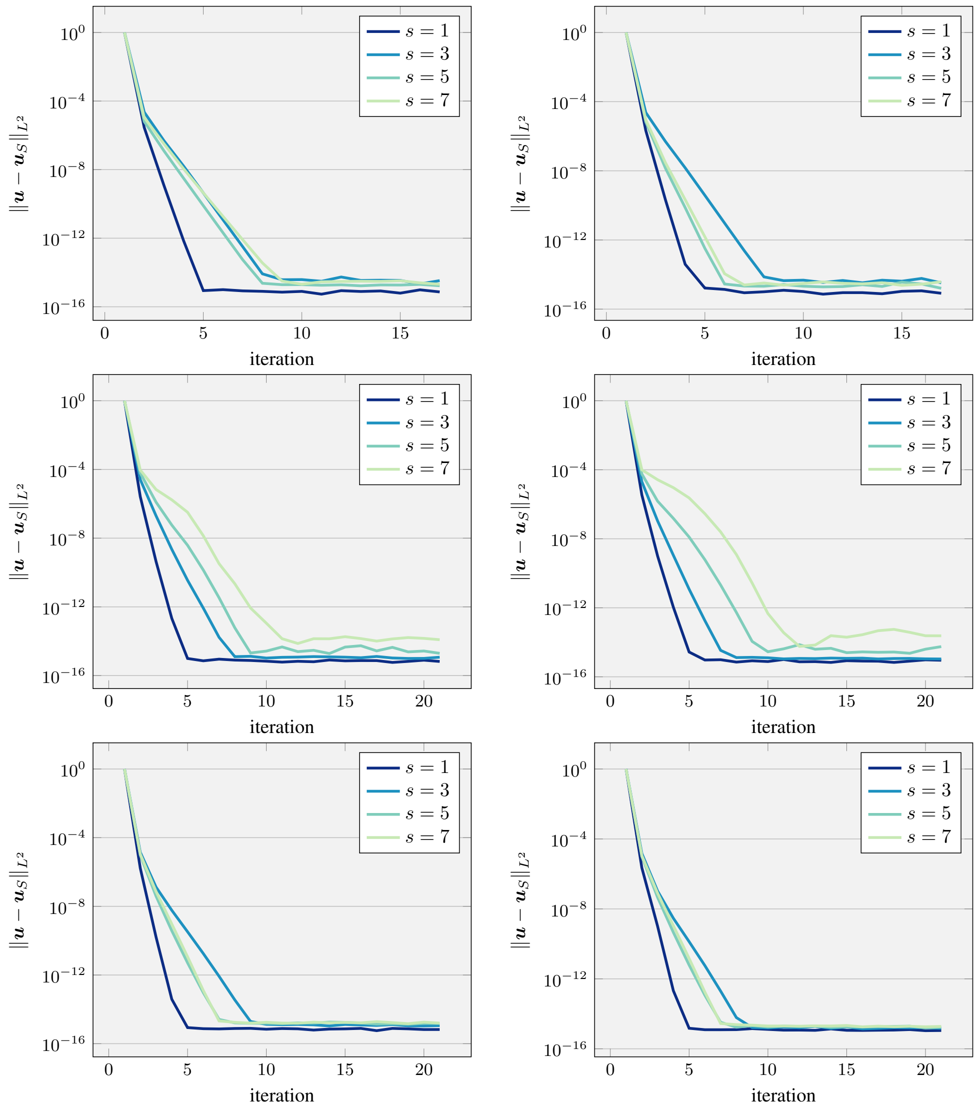


FIG. 4.5. *Evolution of the error in the MGRIT iterates, applied to Burgers' (top), shallow-water (middle) and Euler equations (bottom). MGRIT parameters: V-cycle, F-smoothing,* $3$ *levels with* $N_t = 100, 200, 400$, *respectively,* $N_x = 64$. *Domain:* $L = 1$, $T = 0.5$. *Flux discretisation used: Lax-Friedrichs (left plots,* (2.8)*), and Roe (right plots,* (2.10)*), both employing WENO reconstruction of different orders of accuracy s. Time-stepper used: SSPRK3* (2.11) *with rediscretisation at all levels. Data generated running* `testMGRIT_highOrder.m` *[7].*

*Changing accuracy.* Experimenting with discretisations of different accuracy at different multigrid levels makes sense in attempting 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. 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 Figure 4.6 pinpoint that this strategy is not beneficial to the algorithm performance and that instead simple rediscretisation is preferable. Among the alternatives

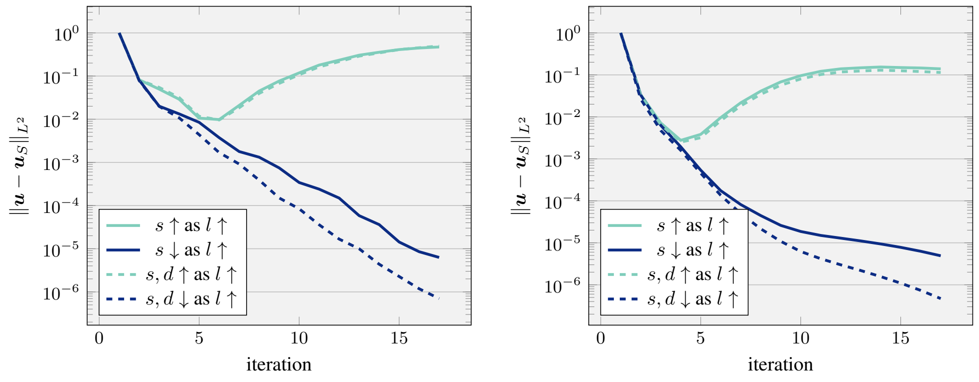


FIG. 4.6. *Impact of modifying the accuracy of the schemes used across the levels on the error convergence of MGRIT. Same parameters as in Figure 4.5, but with the order of the WENO reconstruction $s$ either increasing (green) or decreasing (blue) as we move to coarser levels. Dashed lines show error 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. Data generated via* `testMGRIT_highOrder.m` *[7].*

considered, though, decreasing the accuracy as we descend to coarser levels has proven 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 [27], where a lower-order WENO reconstruction was used at the coarse level.

## 5. Conclusion.

In this paper, we consider the use of MGRIT for the parallelisation in time of the solution of non-linear hyperbolic PDEs. In particular, we aim to understand how the choice of integrators used at each level of the multigrid algorithm impacts its convergence behaviour. For this purpose, we have measured the performance of MGRIT applied to a number of test problems, using a combination of existing integrators commonly used for the solution of conservation laws. The results show 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 can still provide satisfactory convergence results under strict 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, 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. Since the time of writing, some studies in this regard have already been conducted; see for example [22]. Another attractive option for stability is to consider the use of implicit or implicit-explicit (IMEX) schemes for the fine and/or coarse integrators in this setting, but this requires care to balance the added cost of these schemes with their advantageous stability properties.

## REFERENCES

[1] 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., vol. 40 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2005, pp. 425–432.

[2] M. Bolten, D. Moser, and R. Speck, *Asymptotic convergence of the parallel full approximation scheme in space and time for linear problems*, Numer. Linear Algebra Appl., 25 (2018), Art. e2208, 22 pages.

[3] W. L. Briggs, V. E. Henson, and S. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

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

[5] A. J. Christlieb, C. B. Macdonald, B. W. Ong, and R. Spiteri, *Revisionist integral deferred correction with adaptive stepsize control*, Commun. Appl. Math. Comput. Sci., 10 (2013), pp. 1–25.

[6] X. Dai and Y. Maday, *Stable parareal in time method for first- and second-order hyperbolic systems*, SIAM Journal on Scientific Computing, SIAM J. Sci. Comput., 35 (2013), pp. A52–A78.

[7] F. Danieli, *Repository MGRIT_Non-linear_hyperbolic*.
https://gitlab.com/fdanieli/mgrit_non-linear_hyperbolic.

[8] H. De Sterck, R. Falgout, S. Friedhoff, O. Krzysik, and S. MacLachlan, *Optimizing multigrid reduction-in-time and parareal coarse-grid operators for linear advection*, Numer. Linear Alg. Appl., 28 (2021), Art. e2367, 22 pages.

[9] V. A. Dobrev, T. Kolev, N. A. Petersson, and J. B. Schroder, *Two-level convergence theory for multigrid reduction in time (MGRIT)*, SIAM J. Sci. Comput., 39 (2017), pp. S501–S527.

[10] A. Eghbal, A. G. Gerber, and E. Aubanel, *Acceleration of unsteady hydrodynamic simulations using the parareal algorithm*, J. Comput. Sci., 19 (2017), pp. 57–76.

[11] M. Emmett and M. L. Minion, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci., 7 (2012), pp. 105–132.

[12] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid*, SIAM J. Sci. Comput., 36 (2014), pp. C635–C661.

[13] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, J. B. Schroder, and S. Vandewalle, *Multigrid methods with space-time concurrency*, Comput. Vis. Sci., 18 (2017), pp. 123–143.

[14] R. D. Falgout, M. Lecouvez, and C. S. Woodward, *A parallel-in-time algorithm for variable step multistep methods*, J. Comput. Sci., 37 (2019), Art. 101029, 12 pages.

[15] R. D. Falgout, T. A. Manteuffel, B. O'Neill, and J. B. Schroder, *Multigrid reduction in time for nonlinear parabolic problems: a case study*, SIAM J. Sci. Comput., 39 (2017), pp. S298–S322.

[16] 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., vol. 9 of Contrib. Math. Comput. Sci., Springer, Cham, 2015, pp. 69–113.

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

[18] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Rev., 43 (2001), pp. 89–112.

[19] A. Harten and J. M. Hyman, *Self-adjusting grid methods for one-dimensional hyperbolic conservation laws*, J. Comput. Phys., 50 (1983), pp. 235–269.

[20] T. Haut and B. Wingate, *An asymptotic parallel-in-time method for highly oscillatory PDEs*, SIAM J. Sci. Comput., 36 (2014), pp. A693–A713.

[21] J. S. Hesthaven, *Numerical Methods for Conservation Laws*, SIAM, Philadelphia, 2018.

[22] A. J. Howse, H. De Sterck, R. D. Falgout, S. MacLachlan, and J. Schroder, *Parallel-in-time multigrid with adaptive spatial coarsening for the linear advection and inviscid Burgers equations*, SIAM J. Sci. Comput., 41 (2019), pp. A538–A565.

[23] A. Kreienbuehl, A. Naegel, D. Ruprecht, R. Speck, G. Wittum, and R. Krause, *Numerical simulation of skin transport using Parareal*, Comput. Vis. Sci., 17 (2015), pp. 99–108.

[24] R. J. LeVeque, *Numerical Methods for Conservation Laws*, 2nd ed., Birkhäuser, Basel, 1992.

[25] J.-L. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps "pararéel"*, C. R. Acad. Sci. Paris Sér. I Math., 332 (2001), pp. 661–668.

[26] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.

[27] A. S. Nielsen, G. Brunner, and J. S. Hesthaven, *Communication-aware adaptive parareal with application to a nonlinear hyperbolic system of partial differential equations*, J. Comput. Phys., 371 (2018), pp. 483–505.

[28] B. W. Ong, R. D. Haynes, and K. Ladd, *Algorithm 965: RIDC methods: a family of parallel time integrators*, ACM Trans. Math. Software, 43 (2016), Art. 8, 13 pages.

**ETNA**
**Kent State University and**
**Johann Radon Institute (RICAM)**

[29] G. PAGÈS, O. PIRONNEAU, AND G. SALL, *The parareal algorithm for American options*, C. R. Math. Acad. Sci. Paris, 354 (2016), pp. 1132–1138.
[30] J. QIU AND C.-W. SHU, *On the construction, comparison, and local characteristic decomposition for high-order central WENO schemes*, J. Comput. Phys., 183 (2002), pp. 187–209.
[31] K. RUPP, *42 years of microprocessor trend data*.
https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/
[32] D. RUPRECHT, *Wave propagation characteristics of parareal*, Comput. Vis. Sci., 19 (2018), pp. 1–17.
[33] C.-W. SHU, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations (Cetraro, 1997), A. Quarteron, ed., vol. 1697 of Lecture Notes in Math., Springer, Berlin, 1998, pp. 325–432.
[34] A. TOSELLI AND O. B. WIDLUND, *Domain Decomposition Methods—Algorithms and Theory*, Springer, Berlin, 2005.