Computers and Fluids 202 (2020) 104490

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

A simplified Cauchy-Kowalewskaya procedure for the local implicit solution of generalized Riemann problems of hyperbolic balance laws

Gino I. Montecinos[a,*], Dinshaw S. Balsara[b]

[a] Department of Natural Sciences and Technology, Universidad de Ayseén, Coyhaique, Chile
[b] Department of Physics, University of Notre Dame, USA

ARTICLE INFO

Article history:
Received 21 July 2019
Revised 28 January 2020
Accepted 28 February 2020
Available online 29 February 2020

Keywords:
Finite volume schemes
ADER schemes
Generalized Riemann Problems
stiff source terms

ABSTRACT

The Cauchy-Kowalewskaya (CK) procedure is a key building block in the design of solvers for the Generalised Riemann Problem (GRP) based on Taylor series expansions in time. The CK procedure allows us to express time derivatives in terms of purely space derivatives. This is a very cumbersome procedure, which often requires the use of software manipulators. In this paper, a simplification of the CK procedure is proposed in the context of implicit Taylor series expansion for GRP, for hyperbolic balance laws in the framework of [Journal of Computational Physics 303 (2015) 146-172]. A recursive formula for the CK procedure, which is straightforwardly implemented in computational codes, is obtained. The proposed GRP solver is used in the context of the ADER approach and several one-dimensional problems are solved to demonstrate the applicability and efficiency of the present scheme. An enhancement in terms of efficiency, is obtained. Furthermore, the expected theoretical orders of accuracy are achieved, conciliating accuracy and stability.

1. Introduction

This paper concerns the solution of Generalized Riemann Problems (GRP) in the context of ADER schemes, a family of high-order finite volume methods. The ADER (Arbitrary Accuracy DERivative Riemann problem method), first put forward by Toro et al. [26], is of particular interest in this work. The method in [26] was devoted to develop a procedure able to compute the numerical solution, of the one-dimensional linear advection problem, of arbitrary order of accuracy in both space and time. This method can be considered as a generalization of Godunov's method, where the numerical fluxes can be obtained from the local solution of GRP where the initial condition consists of polynomial functions of suitable order. Subsequently, ADER was extended to solve linear systems of hyperbolic conservation laws in [31,38]. The ADER philosophy was extended by Toro and Titarev in [41] to solve the non-linear systems with source terms which are not stiff; inhomogeneous Burgers equation and the nonlinear shallow-water equations with variable bed elevation. In [35], the ADER approach was extended, by Toro and Titarev, to nonlinear but homogeneous hyperbolic systems. Furthermore, the extension of ADER to scalar balance laws was investigated by Toro and Takakura, [34].

The original ADER scheme [26,35], has two main steps, reconstruction and flux calculation. The marching in time generates cell averages, then the reconstruction procedure generates a special type of interpolation polynomial of the solution from cell averages. The flux calculation is carried out from the solution of the GRP, which is proposed in terms of a Taylor series expansion in time, where the time derivatives are completely expressed in terms of spatial derivatives by means of the Cauchy-Kowalewskaya or Lax-Wendroff procedure. The space derivatives are obtained from homogeneous linearized Riemann problems constructed from the governing equation and an initial condition given by the derivatives of interpolation polynomials. This was the summary of the pioneering ADER method, which in principle is able to generate approximations of arbitrary order of accuracy. The accuracy of ADER methods depends on the number of terms in the Taylor series expansions. In the case of conservation laws, for the first order it recovers Godunov's method and for the second order ADER recovers the second-order method of Ben-Artzi and Falcoviz [5].

We mention here the pioneering work of Ben-Artzi [5], Ben-Artzi and Falcovitz [5], Ben-Artzi and Li [6], where GRP consists of Cauchy problems of hyperbolic balance laws and initial condition given by two vectors of polynomials of first order separated by a discontinuity allowed to build numerical scheme of second order of accuracy. The GRP solver in these works is based on a Taylor series expansion in time, where the leading term of the expansion is obtained from an associated Riemann problem like in the case

* Corresponding author.
E-mail address: gino.montecinos@uaysen.cl (G.I. Montecinos).

https://doi.org/10.1016/j.compfluid.2020.104490
0045-7930/

of the GRP in [26,35] and the time derivative in the expansion is obtained from an algebraic system obtained from characteristic curves and contributions of the source term, if present. The solution of the Riemann problem can be done exactly or using some approximate Riemann solver (see [37]). However, the issue of obtaining the algebraic equation for the time derivative is very difficult for general systems and a bit cumbersome to be extended to high orders of accuracy. In [36], Toro has introduced a simplification to the GRP of Ben-Artzi and Falcovitz [5], where the time derivative was obtained from the Cauchy-Kowalewskaya or Lax-Weondroff procedure using the solution of a linearized Riemann problem. This approach can be straightforwardly applied to any hyperbolic system and this work has opened the door to high order extensions of GRP as those of in [26,35]. Some related GRP solvers are, [7,22,24,25] to mention but a few.

In [8], a re-interpretation carried out by Castro and Toro of the high-order numerical method proposed by Harten et al. [20] has allowed us to formulate GRP solutions and thus ADER schemes in a different way. In this new interpretation of ADER, a classical Riemann problem is built from the governing equation and a piece-wise initial condition which is formed from two constant states. In this approach, the constant states are local predictors of the solution within computational cells, that is, these are extrapolation values of the solution at both sides of the cell interfaces at a given time. The evolution of these extrapolated values is carried out by using Taylor series expansion in time, where the time derivatives are still expressed via Cauchy-Kowalewskaya functionals but the arguments are now the spatial derivates of the reconstruction polynomials. In this form the predictors in two adjacent computational cells are interacted at the cell interface through the classical Riemann problem. In [8], the approach based on the Harten et al. is called the HEOC solver and the original GRP solver of Toro and Titarev scheme is referred to as the TT solver. So, the difference between the HEOC formulation and the TT formulation is that, in the HEOC case only one classical Riemann problem is required but it needs to be solved at each quadrature point, whereas, in the TT approach a sequence of classical Riemann problems are required only once, one Riemann problem for the leading term of Taylor expansions and a linearised Riemann problems for every spatial derivative involved in the Cauchy-Kowalewskaya functional. So, once these terms are available the computation of GRP solution via TT is reduced just to evaluate a polynomial in time. The similarities between both approaches are the use of Taylor series expansion and the use of the Cauchy-Kowalewskaya procedure. A detailed review of GRP solvers is done in [8,27]. Similarly, further details of ADER schemes can be found in Chapters 19 and 20 of the textbook by Toro [37]. Notice that, the GRP solvers require the ability of solving classical Riemann problems. The number of hyperbolic systems for which the exact solution of Riemann problems is available, is limited. In general, the exact solution of Riemann problems for several hyperbolic systems can be very difficult to be obtained. Fortunately, the ADER approaches described above allow using approximate Riemann solvers, see [19]. In this sense, the search for approximate Riemann solvers for general hyperbolic systems is a very relevant area of research, where the ADER philosophy can be benefited. Balsara [1,2] and Balsara et al. [3] have extended multidimensional HLL and HLLC to Euler and MHD equations. Goetz et al. [17] have shown that approximate Riemann solvers can be obtained from the well-known HLL solver. See also [4,16] where universal approximate GRP solvers based on HLL method and the inclusion of intermediate waves have been reported.

ADER approach allows the flexibility for incorporating the finite element approach into the finite volume framework. In an intermediate stage in the mixing between finite volume and finite element approach, Dumbser and Munz, Dumbser and Munz [13] and Dumbser and Munz [14], have implemented the ADER scheme for the Discontinuous Galerkin approach applied to the aeroacoustics and the Euler equations in two dimensions. In this approach, ADER is used to evolve polynomials in space and time through the evolution of their degrees of freedom by using Taylor series expansion and the Cauchy-Kowalewskaya procedure, but instead of using reconstruction and the derivatives of reconstruction polynomials, the authors proposed to use the test functions of the finite element space. In this sense, the work of Dumbser et al. [11] in their pioneering work, has introduced the Galerkin framework for obtaining the predictor within cells. The difference between this approach and that of Dumbser and Munz is that this neither requires the use of any Taylor series expansion nor Cauchy-Kowalewskaya procedure. Some contributions to the developments of this class of solver can be found in [9,10,15,18,28], to mention but a few.

The methodologies based on the Galerkin approach requires the inversion of matrices and the solution of non-linear algebraic equations which are very time-consuming processes. On the other hand, the methodologies based on Taylor series suffer from the Cauchy-Kowalewskaya procedure which becomes cumbersome when the accuracy increases. Furthermore, the Taylor series expansions without modifications cannot deal with stiff source terms. In this sense, in [29,39] Toro and Montecinos have introduced the implicit Taylor series expansion and Cauchy-Kowalewskaya procedure to deal with hyperbolic balance laws with stiff source terms. In this approach, the Cauchy-Kowalewskaya procedure requires the spatial derivatives evolved in time, which cannot be obtained straightforwardly from Riemann problems as in conventional ADER methods discussed above. This is basically because conventional ADER methods based on TT solver uses linearised Riemann problems which are homogeneous. Thus, the influence of the source term is only involved in the Cauchy-Kowalewskaya procedure. In [39] the spatial derivatives are evolved by using two approaches, which differ in the number of terms in the Taylor expansion, so Complete Implicit Taylor Approach (CITA) and Reduced Implicit Taylor Approach (RITA), in both HEOC and TT approaches, are investigated. A limitation of the approach in [39], is the computational cost, for high orders of accuracy, the CPU time increases dramatically. However, second order approaches work very well as reported in [42] where an extension to transport phenomena on unstructured meshes has been reported.

In this paper, we propose a strategy related to the implicit Taylor series expansion and the Cauchy-Kowalewskaya procedure. The difference between the conventional approaches described above and the proposed one is that spatial derivatives do not need to be evolved in terms of any Taylor series expansion and the Cauchy-Kowalewskaya procedure is modified by expressing high-order time derivatives not only in terms of spatial derivatives of the data but also on space and time derivatives of the Jacobian matrices of the flux and source functions as well. The derivatives are obtained from an interpolation fashion on selected nodal points. This simplification allows us to provide a closed form for the Cauchy-Kowalewskaya functionals, in a recursive formula. Furthermore, this approach requires the solution of one algebraic equation but the number of variables is the same for all orders of accuracy. Closed forms of the Cauchy-Kowalewskaya functional are available for some partial differential equations as linear advection systems, linear systems with constant matrices [21,30] and for the non-linear two-dimensional Euler equation, [14]. For the general case, algebraic software manipulators are required for generating the Cauchy-Kowalewskaya functionals and gigantic expressions may be obtained and lead, for instance, to long compile times. However, the expression obtained here is very compact and useful for all hyperbolic balance laws.

This paper is organized as follows. In Section 2, the general framework is presented. In Section 3, the new predictor step is in-

troduced. In the Section 4, numerical tests are concerned. Finally, in Section 5 conclusions and remarks are drawn.

2. The framework

In this paper we present a strategy for solving hyperbolic balance law in the conservative form

$$\begin{aligned}\partial_t \mathbf{Q} + \partial_x \mathbf{F}(\mathbf{Q}) &= \mathbf{S}(\mathbf{Q}),\\ \mathbf{Q}(x,0) &= \mathbf{H}_0(x),\end{aligned} \tag{1}$$

where $\mathbf{H}_0(x)$ is a prescribed function in $\mathbb{R}^m$. Here $\mathbf{Q}(x,t) \in \mathbb{R}^m$ is the vector of unknowns, $\mathbf{F}(\mathbf{Q}) \in \mathbb{R}^m$ is the physical flux function and $\mathbf{S}(\mathbf{Q}) \in \mathbb{R}^m$ is the source term.

To compute a numerical solution of (1), we divide the computational domain into N uniform cells of the form $I_i^n := [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [t^n, t^{n+1}]$ and then by integrating on I_i^n we obtain the well-known one-step formula.

$$\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+\frac{1}{2}} - \mathbf{F}_{i-\frac{1}{2}}\right) + \Delta t \mathbf{S}_i, \tag{2}$$

where

$$\mathbf{Q}_i^n = \frac{1}{\Delta x}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{Q}(x,t^n)dx \tag{3}$$

and the numerical flux $\mathbf{F}_{i+\frac{1}{2}}$ as well as the source term $\mathbf{S}_i$ are computed by adopting the ADER strategy, [26,37,40,41]. In this paper

$$\begin{aligned}\mathbf{F}_{i+\frac{1}{2}} &= \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \mathbf{F}_h\left(\mathbf{Q}_i\left(x_{i+\frac{1}{2}},t\right), \mathbf{Q}_{i+1}\left(x_{i+\frac{1}{2}},t\right)\right)dt,\\ \mathbf{S}_i &= \frac{1}{\Delta t \Delta x}\int_{t^n}^{t^{n+1}}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{S}(\mathbf{Q}_i(x,t))dxdt\ ,\end{aligned} \tag{4}$$

where $\mathbf{F}_h(\mathbf{Q}_L, \mathbf{Q}_R)$ is a classical numerical flux function, which can be seen as a function of two states $\mathbf{Q}_L$ and $\mathbf{Q}_R$. It is possible to design an approximate GRP solver out of the HLL, [17]. In this paper we will use the Rusanov solver, obtained from HLL by taking extreme left and right maximum waves speed to be the same but in opposite directions. Here $\mathbf{Q}_i(x, t)$ corresponds to a predictor within the computational cell I_i^n. In the next section, further details of the predictor step are provided.

3. The predictor step

In this section we provide the details to obtain the predictor $\mathbf{Q}_i(x, t)$. We modify the strategy of the implicit Taylor series expansion and the Cauchy-Kowalewskaya procedure presented in [39]. For the sake of completeness, we provide a brief review of the approach in [39]. The predictor is computed as

$$\mathbf{Q}_i(x,\tau) = \mathbf{Q}_i(x,0_+) - \sum_{k=1}^{M}\frac{(-\tau)^k}{k!}\partial_t^{(k)}\mathbf{Q}_i(x,\tau)\ , \tag{5}$$

which by means of the Cauchy-Kowalewskaya procedure, can be written as

$$\begin{aligned}\mathbf{Q}_i(x,\tau) = \mathbf{Q}_i(x,0_+)\\ - \sum_{k=1}^{M}\frac{(-\tau)^k}{k!}\mathbf{G}^{(k)}\left(\mathbf{Q}_i(x,\tau), \partial_x\mathbf{Q}_i(x,\tau), ..., \partial_x^{(k)}\mathbf{Q}_i(x,\tau)\right),\end{aligned} \tag{6}$$

where M is an integer which corresponds to the order of accuracy $M+1$, in both space and time. Here, $\mathbf{G}^{(k)}$ corresponds to the Cauchy-Kowalewskaya functional and it is the function which expresses the time derivatives in terms of spatial derivatives, $\partial_t^{(k)}\mathbf{Q}_i(x,\tau) = \mathbf{G}^{(k)}(\mathbf{Q}_i(x,\tau), \partial_x\mathbf{Q}_i(x,\tau), ..., \partial_x^{(k)}\mathbf{Q}_i(x,\tau))$. Notice that this functional requires the information of spatial derivatives at τ, which must be estimated. In [39] two strategies are proposed, where the time spatial derivatives are obtained from implicit Taylor series as well. These approaches require the solution of algebraic equations.

In the next section we provide a brief review of the conventional Cauchy-Kowalewskaya procedure and subsequently, a simplification of this procedure is presented.

3.1. A brief review of the Cauchy-Kowalewskaya procedure

Here, we briefly describe the Cauchy-Kowalewskaya procedure for obtaining the time derivatives of the data. For the sake of simplicity, in this section we omit the sub index i in $\mathbf{Q}_i$, to indicate the approximation within the cell $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$.

The time derivatives are obtained from the governing Eq. (1). As for example, the first derivative is given by

$$\partial_t \mathbf{Q} = -\mathbf{A}(\mathbf{Q})\partial_x\mathbf{Q} + \mathbf{S}(\mathbf{Q})\ , \tag{7}$$

where $\mathbf{A}(\mathbf{Q})$ is the Jacobian matrix of $\mathbf{F}(\mathbf{Q})$ with respect to $\mathbf{Q}$. So, to obtain the second time derivative we differentiate in time the Eq. (A.1), so we have

$$\begin{aligned}\partial_t^{(2)}\mathbf{Q}_i = &-\sum_{j=1}^{m}\sum_{l=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_l}\partial_t\mathbf{Q}_l\partial_x\mathbf{Q}_j - \sum_{j=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\partial_t(\partial_x\mathbf{Q}_j)\\ &+ \sum_{j=1}^{m}\mathbf{B}(\mathbf{Q})_{i,j}\partial_t\mathbf{Q}_j\ ,\end{aligned} \tag{8}$$

where $\mathbf{B}(\mathbf{Q})$ is the Jacobian matrix of $\mathbf{S}(\mathbf{Q})$ with respect to $\mathbf{Q}$, $\partial_t\mathbf{Q}_i$ and $\partial_t\mathbf{Q}_j$ are the ith and the jth components of the vector state $\partial_t\mathbf{Q}$, respectively and then by differentiating the expression (A.1) with respect to x, we obtain

$$\begin{aligned}\partial_x(\partial_t\mathbf{Q}_j) = &-\sum_{j=1}^{m}\sum_{l=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_l}\partial_x\mathbf{Q}_l\partial_x\mathbf{Q}_j - \sum_{j=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\partial_x^{(2)}\mathbf{Q}_j\\ &+ \sum_{j=1}^{m}\mathbf{B}(\mathbf{Q})_{i,j}\partial_x\mathbf{Q}_j\ ,\end{aligned} \tag{9}$$

the same procedure is applied to obtain $\partial_t^{(3)}\mathbf{Q}$ and so on, in principle any high order time derivative can be obtained through this procedure. However, the procedure becomes very cumbersome for derivatives of orders higher than two, furthermore the complexity scales with the number of unknowns m and the order of accuracy as well, see Appendix A. This justifies the requirement of finding some efficient strategy to approximate temporal derivatives by following the Cauchy-Kowalewskaya ideas.

3.2. The simplified Cauchy-Kowalewskaya procedure

In this section, we derive a simplified Cauchy-kowalewskaya procedure, to approximate time derivatives. For the sake of simplicity, in this section we also omit the sub index i in $\mathbf{Q}_i$, to indicate the approximation within the cell $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$. As seen in the previous section, the conventional Cauchy-Kowalewskaya procedure provides the first time derivative as

$$\partial_t \mathbf{Q} = -\mathbf{A}(\mathbf{Q})\partial_x\mathbf{Q} + \mathbf{S}(\mathbf{Q})\ . \tag{10}$$

At this point we introduce the *first simplification*. Instead of considering the previous equation, we are going to use the approximation

$$\partial_t \mathbf{Q} = -\mathbf{A}(x,t)\partial_x\mathbf{Q} + \mathbf{S}(\mathbf{Q})\ , \tag{11}$$

which means, the matrix $\mathbf{A}$ is considered just a space-time dependent matrix. This idea arises from the fact that $\mathbf{A}(\mathbf{Q}(x, t))$ is actually a matrix which depends on (x, t) through the state $\mathbf{Q}$. It is very

important to remark that $\mathbf{A}(x, t)$ is not a linearization of the Jacobian matrix around any state, this must be formally considered as a matrix which now depends on two variables (x, t) and if this is regular enough, then this can be differentiated with respect to x and t. Further details are provided in the Section 3.3.

By taking into account the approximation (11), the second time derivative can be approximated in three steps described below.

Step I. We differentiate $\partial_t \mathbf{Q}$ with respect to t, thus

$$\partial_t(\partial_t \mathbf{Q}) = -\mathbf{A}_t \partial_x \mathbf{Q} - \mathbf{A}\partial_t(\partial_x \mathbf{Q}) + \mathbf{B}\partial_t \mathbf{Q}\,, \tag{12}$$

where $\mathbf{B}$ is the Jacobian matrix of the source term with respect to $\mathbf{Q}$. For the remaining part of this paper, we use the notation $\partial_x \mathbf{A} = \mathbf{A}_x$ for any matrix $\mathbf{A}$. Similarly, we use the convention $\partial_x^{(l)} \mathbf{A} = \mathbf{A}_x^{(l)}$ for the l-th partial derivative of the matrix $\mathbf{A}$ with respect to x. Do not confuse with $\mathbf{A}^l$ which means matrix multiplication, l times.

Step II. At this point we introduce the *second simplification*, which is to consider also the matrix $\mathbf{B}$ as a space-time dependent matrix rather than a state dependent matrix. We assume regularity enough such that the spatial and time derivatives can be interchanged. So

$$\partial_t^{(2)} \mathbf{Q} = -\mathbf{A}_t \partial_x \mathbf{Q} - \mathbf{A}\partial_x(\partial_t \mathbf{Q}) + \mathbf{B}\partial_t \mathbf{Q}\,. \tag{13}$$

Similarly to the first simplification, this idea arises from the fact that $\mathbf{B}(\mathbf{Q}(x, t))$ at the end is a matrix which depends on (x, t) through the state $\mathbf{Q}$. So, if $\mathbf{B}(x, t)$ is a regular enough matrix function, then this can be differentiated with respect to x and t. Further details are provided in the Section 3.3.

Step III. Here, we differentiate in space the expression $\partial_t \mathbf{Q}$, taking into account the simplifications introduced above, to obtain

$$\begin{aligned}\partial_x(\partial_t \mathbf{Q}) &= -\mathbf{A}_x \partial_x \mathbf{Q} - \mathbf{A}\partial_x^{(2)} \mathbf{Q} + \mathbf{B}\partial_x \mathbf{Q} \\ &= -\mathbf{A}\partial_x^{(2)} \mathbf{Q} + \left(\mathbf{B} - \mathbf{A}_x^{(1)}\right)\partial_x \mathbf{Q}\,.\end{aligned} \tag{14}$$

This step regards the main difference with respect to the conventional Cauchy-Kowalewskaya procedure. By inserting the previous expression into (13), we obtain

$$\begin{aligned}\partial_t^{(2)} \mathbf{Q} &= -\mathbf{A}_t \partial_x \mathbf{Q} - \mathbf{A}\left(-\mathbf{A}\partial_x^{(2)} \mathbf{Q} + \left(\mathbf{B} - \mathbf{A}_x^{(1)}\right)\partial_x \mathbf{Q}\right) + \mathbf{B}\partial_t \mathbf{Q} \\ &= \mathbf{A}^2 \partial_x^{(2)} \mathbf{Q} + \left(-\mathbf{A}_t - \mathbf{A}\left(\mathbf{B} - \mathbf{A}_x^{(1)}\right)\right)\partial_x \mathbf{Q} + \mathbf{B}\partial_t \mathbf{Q}\,.\end{aligned} \tag{15}$$

The last expression can be written as

$$\partial_t^{(2)} \mathbf{Q} = \mathbf{C}(2,2)\partial_x^{(2)} \mathbf{Q} + \mathbf{C}(2,1)\partial_x \mathbf{Q} + \mathbf{B}\partial_t \mathbf{Q}\,, \tag{16}$$

where $\mathbf{C}(2, i)$, $i = 1, 2$ represent the matrix coefficients

$$\mathbf{C}(2,2) = \mathbf{A}^2\,,\ \mathbf{C}(2,1) = -\mathbf{A}_t - \mathbf{A}\left(\mathbf{B} - \mathbf{A}_x^{(1)}\right)\,. \tag{17}$$

The novel contribution of this paper is as follows. We are going to show that this procedure can be generalized. Before giving this main result, we need to express $\partial_x^{(l)}(\partial_t \mathbf{Q})$ only in terms of spatial derivatives of the data and the Jacobian matrices. Notice that from simplifications introduced above, we obtain

$$\begin{aligned}\partial_x(\partial_t \mathbf{Q}) &= -\mathbf{A}\partial_x^{(2)} \mathbf{Q} + (\mathbf{B} - \mathbf{A}_x)\partial_x \mathbf{Q}\,, \\ \partial_x^{(2)}(\partial_t \mathbf{Q}) &= -\mathbf{A}\partial_x^{(3)} \mathbf{Q} + (\mathbf{B} - 2\mathbf{A}_x)\partial_x^{(2)} \mathbf{Q} + \left(\mathbf{B}_x - \mathbf{A}_x^{(2)}\right)\partial_x \mathbf{Q}\,, \\ \partial_x^{(3)}(\partial_t \mathbf{Q}) &= -\mathbf{A}\partial_x^{(4)} \mathbf{Q} + (\mathbf{B} - 3\mathbf{A}_x)\partial_x^{(3)} \mathbf{Q} + \left(2\mathbf{B}_x - 3\mathbf{A}_x^{(2)}\right)\partial_x^{(2)} \mathbf{Q} \\ &\quad + \left(\mathbf{B}_x^{(2)} - \mathbf{A}_x^{(3)}\right)\partial_x \mathbf{Q}\,, \\ \partial_x^{(4)}(\partial_t \mathbf{Q}) &= -\mathbf{A}\partial_x^{(5)} \mathbf{Q} + (\mathbf{B} - 4\mathbf{A}_x)\partial_x^{(4)} \mathbf{Q} + \left(3\mathbf{B}_x - 6\mathbf{A}_x^{(2)}\right)\partial_x^{(3)} \mathbf{Q} \\ &\quad + \left(3\mathbf{B}_x^{(2)} - 4\mathbf{A}_x^{(3)}\right)\partial_x^{(2)} \mathbf{Q} + \left(\mathbf{B}_x^{(3)} - \mathbf{A}_x^{(4)}\right)\partial_x \mathbf{Q}\,, \\ \partial_x^{(5)}(\partial_t \mathbf{Q}) &= -\mathbf{A}\partial_x^{(6)} \mathbf{Q} + (\mathbf{B} - 5\mathbf{A}_x)\partial_x^{(5)} \mathbf{Q} + \left(4\mathbf{B}_x - 10\mathbf{A}_x^{(2)}\right)\partial_x^{(4)} \mathbf{Q} \\ &\quad + \left(6\mathbf{B}_x^{(2)} - 10\mathbf{A}_x^{(3)}\right)\partial_x^{(3)} \mathbf{Q} + \left(4\mathbf{B}_x^{(3)} - 5\mathbf{A}_x^{(4)}\right)\partial_x^{(2)} \mathbf{Q} \\ &\quad + \left(\mathbf{B}_x^{(4)} - \mathbf{A}_x^{(5)}\right)\partial_x \mathbf{Q}\,,\end{aligned} \tag{18}$$

Table 1
Coefficients $a_{l,k}$ in expression (19).

$a_{l,(l-4)}$	$a_{l,(l-3)}$	$a_{(l,l-2)}$	$a_{(l,l-1)}$	$a_{(l,l)}$	$a_{(l,l+1)}$	l
0	0	0	0	1	1	1
0	0	0	1	2	1	2
0	0	1	3	3	1	3
0	1	4	6	4	1	4
1	5	10	10	5	1	5

Table 2
Coefficients $b_{l,k}$ in expression (19).

$b_{l,(l-4)}$	$b_{l,(l-3)}$	$b_{(l,l-2)}$	$b_{(l,l-1)}$	$b_{(l,l)}$	$b_{(l,l+1)}$	l
0	0	0	0	0	1	1
0	0	0	0	1	1	2
0	0	0	1	2	1	3
0	0	1	3	3	1	4
0	1	4	6	4	1	5

so by inspection we observe that these derivatives, can be arranged as

$$\partial_x^{(l)}(\partial_t \mathbf{Q}) = \sum_{k=1}^{l+1} \left(b_{l,k}\mathbf{B}_x^{(l+1-k)} - a_{l,k}\mathbf{A}_x^{(l+2-k)}\right)\partial_x^{(k)} \mathbf{Q}\,. \tag{19}$$

Notice that, $\partial_x^{(l)}$ stands by the l-th spatial derivative, with the convention $\partial_x^{(0)} \mathbf{M} = \mathbf{M}$ for any function $\mathbf{M}$, which may be a scalar, vector or matrix function. This notation is also extended to temporal derivatives.

Table 1, shows the coefficients $a_{l,k}$. Similarly, Table 2 shows the coefficients $b_{l,k}$. We observe that they follow the structure of the Pascal triangle, in the combinatorial theory. In fact, the structure is given by the following.

Lemma 3.1.

$$\partial_x^{(l)}(\partial_t \mathbf{Q}) = \sum_{k=1}^{l+1} \mathbf{D}(l+1, k)\partial_x^{(k)} \mathbf{Q}\,, \tag{20}$$

where

$$\mathbf{D}(l+1, k) = \left(\binom{l-1}{l-k}\mathbf{B}_x^{(l-k)} - \binom{l}{l+1-k}\mathbf{A}_x^{(l+1-k)}\right) \tag{21}$$

and

$$\binom{l}{-1} = 0\,, \tag{22}$$

for all integer l.

Proof. Let us prove it by induction.

- We already know that this is true for $l = 1$.
- Let us assume it is true for l, that is

$$\partial_x^{(l)}(\partial_t \mathbf{Q}) = \sum_{k=1}^{l+1} \left(\binom{l-1}{l-k}\mathbf{B}_x^{(l-k)} - \binom{l}{l+1-k}\mathbf{A}_x^{(l+1-k)}\right)\partial_x^{(k)} \mathbf{Q}\,. \tag{23}$$

- Let us prove it is true for $l+1$. Indeed

$$\begin{aligned}\partial_x^{(l+1)}(\partial_t \mathbf{Q}) &= \sum_{k=1}^{l+1} \left(\binom{l-1}{l-k}\mathbf{B}_x^{(l+1-k)} - \binom{l}{l+1-k}\mathbf{A}_x^{(l+2-k)}\right)\partial_x^{(k)} \mathbf{Q} \\ &\quad + \sum_{k=1}^{l+1} \left(\binom{l-1}{l-k}\mathbf{B}_x^{(l+1-k)} - \binom{l}{l+1-k}\mathbf{A}_x^{(l+1-k)}\right)\partial_x^{(k+1)} \mathbf{Q} \\ &= \left(\binom{l-1}{l-1}\mathbf{B}_x^{(l+1)} - \binom{l}{l}\mathbf{A}_x^{(l+2)}\right)\partial_x \mathbf{Q}\end{aligned}$$

$$+\sum_{k=2}^{l+1}\left[\binom{l-1}{l+1-k}+\binom{l-1}{l-k}\right]\mathbf{B}_x^{(l+1-k)}\partial_x^{(k)}\mathbf{Q}$$
$$-\sum_{k=2}^{l+1}\left[\binom{l}{l+2-k}+\binom{l}{l+1-k}\right]\mathbf{A}_x^{(l+2-k)}\partial_x^{(k)}\mathbf{Q}$$
$$+\left(\binom{l-1}{-1}\mathbf{B}_x^{(-1)}-\binom{l}{0}\mathbf{A}_x^{(0)}\right)\partial_x^{(l+2)}\mathbf{Q}. \tag{24}$$

By considering the properties of the combinatorial factors

$$\binom{l-1}{l+1-k}+\binom{l-1}{l-k}=\binom{l}{l+1-k},$$
$$\binom{l-1}{l-1}=\binom{l}{l}=\binom{l}{0}=\binom{l-1}{0}=1 \tag{25}$$

and the assumption

$$\binom{m}{-1}=0\,, \tag{26}$$

for all m, after grouping terms we obtain

$$\partial_x^{(l+1)}(\partial_t\mathbf{Q})=\sum_{k=1}^{l+2}\left(\binom{l}{l+1-k}\mathbf{B}_x^{(l+1-k)}-\binom{l+1}{l+2-k}\mathbf{A}_x^{(l+2-k)}\right)\partial_x^{(k)}\mathbf{Q}. \tag{27}$$

This completes the proof.

□

Proposition 3.2. *The high-order time derivatives have the following recursive form*

$$\partial_t^{(k)}\mathbf{Q}=\sum_{l=1}^{k}\mathbf{C}(k,l)\partial_x^{(l)}\mathbf{Q}+\partial_t^{(k-2)}(\mathbf{B}\partial_t\mathbf{Q})\,, \tag{28}$$

where

$$\mathbf{C}(k,l)=\begin{cases}\mathbf{C}(k-1,k-1)\mathbf{D}(k,k), & l=k\,,\\ \mathbf{C}(k-1,l)_t+\sum_{m=l-1}^{k-1}\mathbf{C}(k-1,m)\mathbf{D}(m+1,l), & l<k\,,\end{cases} \tag{29}$$

here, the matrix $\mathbf{D}$ *is given by (21). We impose* $\mathbf{C}(k,0)=\mathbf{0}\ \forall k>0$, $\mathbf{C}(1,1)=-\mathbf{A}$ *and* $\partial_t^{(-1)}(\mathbf{B}\partial_t\mathbf{Q})=\mathbf{S}(\mathbf{Q})$.

Proof. Let us prove this proposition by induction.

- The result is true for $k=2$. In fact, we know that

  $$\partial_t\mathbf{Q}=-\mathbf{A}\partial_x\mathbf{Q}+\mathbf{S}(\mathbf{Q})\,. \tag{30}$$

  Since, $\partial_t^{(2)}\mathbf{Q}=\partial_t(\partial_t\mathbf{Q})$ from (30) we obtain

  $$\partial_t^{(2)}\mathbf{Q}=\mathbf{A}^2\partial_x^{(2)}\mathbf{Q}+(-\mathbf{A}_t-\mathbf{A}(\mathbf{B}-\mathbf{A}_x))\partial_x\mathbf{Q}+\mathbf{B}\partial_t\mathbf{Q}\,. \tag{31}$$

  Here, we have used the chain rule

  $$\partial_t(\mathbf{S}(\mathbf{Q}))=\mathbf{B}\partial_t\mathbf{Q}\,. \tag{32}$$

  Therefore, from the expressions $\mathbf{C}(1,1)=-\mathbf{A}$, $\mathbf{D}(2,2)=-\mathbf{A}$, $\mathbf{D}(2,1)=\mathbf{B}-\mathbf{A}_x$ and by identifying terms, the induction hypothesis is valid for $k=2$.
- We assume the induction hypothesis is valid for $k=n$ and thus

  $$\partial_t^{(k)}\mathbf{Q}=\sum_{l=1}^{k}\mathbf{C}(k,l)\partial_x^{(l)}\mathbf{Q}+\partial_t^{(k-2)}(\mathbf{B}\partial_t\mathbf{Q})\,, \tag{33}$$

  for all $k\le n$.
- Let us prove this is valid for $k=n+1$. In fact

  $$\begin{aligned}&\partial_t^{(n+1)}\mathbf{Q}\\ &=\sum_{l=1}^{n}\partial_t(\mathbf{C}(n,l)\partial_x^{(l)}\mathbf{Q})+\partial_t^{(n-1)}(\mathbf{B}\partial_t\mathbf{Q})\\ &=\sum_{l=1}^{n}\mathbf{C}(n,l)_t\partial_x^{(l)}\mathbf{Q}+\sum_{l=1}^{n}\mathbf{C}(n,l)\partial_x^{(l)}(\partial_t\mathbf{Q})+\partial_t^{(n-1)}(\mathbf{B}\partial_t\mathbf{Q})\\ &=\sum_{l=1}^{n}\mathbf{C}(n,l)_t\partial_x^{(l)}\mathbf{Q}+\sum_{l=1}^{n}\mathbf{C}(n,l)\sum_{m=1}^{l+1}\mathbf{D}(l+1,m)\partial_x^{(m)}\mathbf{Q}\\ &+\partial_t^{(n-1)}(\mathbf{B}\partial_t\mathbf{Q})=\sum_{l=1}^{n}\mathbf{C}(n,l)_t\partial_x^{(l)}\mathbf{Q}\\ &+\sum_{m=1}^{n}\mathbf{C}(n,m)\sum_{l=1}^{m+1}\mathbf{D}(m+1,l)\partial_x^{(l)}\mathbf{Q}+\partial_t^{(n-1)}(\mathbf{B}\partial_t\mathbf{Q})\\ &=\sum_{l=1}^{n}\Big[\mathbf{C}(n,l)_t+\sum_{m=l-1}^{n}\mathbf{C}(n,m)\mathbf{D}(m+1,l)\Big]\partial_x^{(l)}\mathbf{Q}\\ &+\mathbf{C}(n,n)\mathbf{D}(n+1,n+1)\Big]\partial_x^{(n+1)}\mathbf{Q}+\partial_t^{(n-1)}(\mathbf{B}\partial_t\mathbf{Q})\,,\end{aligned} \tag{34}$$

  with $\mathbf{C}(n,0)=\mathbf{0}$. So by collecting terms and defining

  $$\mathbf{C}(n+1,l)=\begin{cases}\mathbf{C}(n,n)\mathbf{D}(n+1,n+1) & ,l=n+1\,,\\ \mathbf{C}(n,l)_t+\sum_{m=l-1}^{n}\mathbf{C}(n,m)\mathbf{D}(m+1,l) & ,l<n+1\,,\end{cases} \tag{35}$$

  we can write (34) as

  $$\partial_t^{(n+1)}\mathbf{Q}=\sum_{l=1}^{n+1}\mathbf{C}(n+1,l)\partial_x^{(l)}\mathbf{Q}+\partial_t^{(n-1)}(\mathbf{B}\partial_t\mathbf{Q})\,, \tag{36}$$

  this proves the sought result.

Notice that the condition $\partial_t^{(-1)}(\mathbf{B}\partial_t\mathbf{Q})=\partial_t^{(-1)}(\partial_t\mathbf{S}(\mathbf{Q}))=\partial_t^{(0)}\mathbf{S}(\mathbf{Q})=\mathbf{S}(\mathbf{Q})$ is natural, which also justifies the expression (30), it also corresponds to $k=1$ in the formula (28). □

Notice that the expression (28) is only possible from the simplifications proposed in this work. This is not possible, in general, for the conventional Cauchy-Kowalewskaya procedure. Notice that (28) expresses the time derivatives in terms of spatial derivatives of $\mathbf{Q}$, space and time derivatives of both $\mathbf{A}$ and $\mathbf{B}$.

Corollary 3.3. *The expression (28) can be written as*

$$\partial_t^{(k)}\mathbf{Q}=\mathbf{M}_k+\mathbf{B}\partial_t^{(k-1)}\mathbf{Q}\,, \tag{37}$$

where

$$\mathbf{M}_k=\sum_{l=1}^{k}\mathbf{C}(k,l)\partial_x^{(l)}\mathbf{Q}+\sum_{l=1}^{k-2}\binom{k-2}{l-1}\mathbf{B}_t^{(k-1-l)}\partial_t^{(l)}\mathbf{Q}\,. \tag{38}$$

Proof. This result follows from the manipulation of (28) in Proposition 3.2. Particularly the term $\partial_t^{(k-2)}(\mathbf{B}\partial_t\mathbf{Q})$ can be expressed, by using the matrix-vector differentiation rules, as

$$\partial_t^{(k-2)}(\mathbf{B}\partial_t\mathbf{Q})=\sum_{l=1}^{k-1}\binom{k-2}{l-1}\mathbf{B}_t^{(k-1-l)}\partial_t^{(l)}\mathbf{Q}\,. \tag{39}$$

By collecting terms and isolating for $l=k-1$, we obtain

$$\partial_t^{(k)}\mathbf{Q}=\sum_{l=1}^{k}\mathbf{C}(k,l)\partial_x^{(l)}\mathbf{Q}+\sum_{l=1}^{k-2}\binom{k-2}{l-1}\mathbf{B}_t^{(k-1-l)}\partial_t^{(l)}\mathbf{Q}+\mathbf{B}\partial_t^{(k-1)}\mathbf{Q} \tag{40}$$

and thus the result holds. □

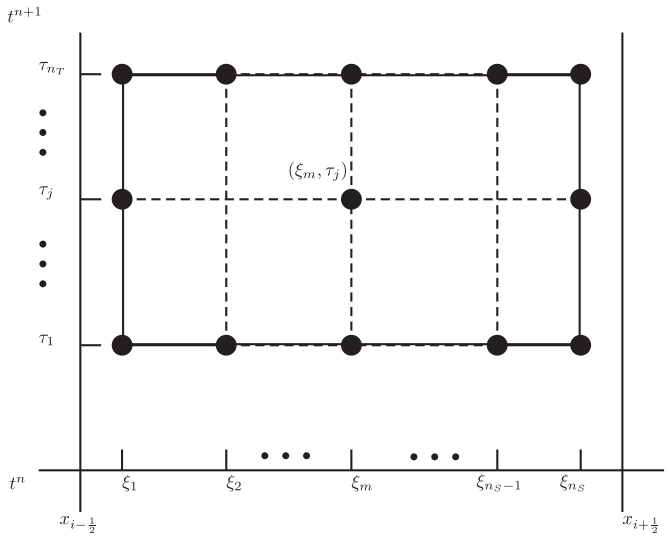

Fig. 1. Sketch of the space-time node distribution.

Proposition 3.4.

$$\partial_t^{(k)}\mathbf{Q} = \sum_{r=2}^{k} \mathbf{M}_r + \mathbf{B}^{k-1}\mathbf{S}(\mathbf{Q}) \,, \tag{41}$$

where $\mathbf{M}_r$ *are those in (38).*

Proof. This is a consequence of the Corollary 3.3. □

As will be seen in the next sections, the previous results provide the closed form for approximations to Cauchy-Kowalewskaya functionals, $\mathbf{G}^{(k)}$, which will be important to design fixed-point iteration procedures. In B.1, we provide operational details for generating the matrix $\mathbf{C}(k, l)$ involved into the simplified Cauchy-Kowaleskaya procedure introduced in this section.

3.3. The predictor step based on a modified implicit Taylor series expansion and the simplified Cauchy-Kowalewskaya procedure

Notice that the predictor $\mathbf{Q}_i$, within I_i^n, is required for evaluating integrals in (4). On the other hand, the evaluation of these integrals is carried out by means of quadrature rules in space and time. So, for the temporal integration we use the Gaussian rule, which involves τ_j, $j = 1, ..., n_T$ Gaussian points. Whereas, for the spatial integration, we use the Newton-Cotes rule, which involves ξ_m, $m = 1, ..., n_S$ equidistant quadrature points, with $n_S = M + 1$ where $M + 1$ is the accuracy. See B.2 for further details about the set up of these quadrature points through reference elements. This set of quadrature points allows us to build space-time nodal points (ξ_m, τ_j) within the space-time cell I_i^n, as illustrated in the Fig. 1. Here, ξ_1 and ξ_{n_S} coincide with the cell interfaces. So, from the previous comment it is evident that for flux and source evaluations, we only need the information of $\mathbf{Q}_i$ at (ξ_m, τ_j).

To obtain approximation of the predictor at every space-time node (ξ_m, τ_j), we propose the following iterative strategy.

1. Provide a starting guess for $\mathbf{Q}_i(\xi_m, \tau_j)$, $m = 1, ..., n_S$ and $j = 1, ..., n_T$.
   This is done by using the formula
   $$\mathbf{Q}_i(\xi_m, \tau_j) = [\mathbf{I} - \tau_j \mathbf{B}(\mathbf{W}(\xi_m))]^{-1}(\mathbf{W}_i(\xi_m) - \tau_j \mathbf{A}(\mathbf{W}(\xi_m))\partial_x \mathbf{Q}_i(\xi_m, \tau_j)) \,, \tag{42}$$
   which corresponds to the second order accurate expression in [39]. Here, $\mathbf{W}_i(\xi)$ represents the reconstruction polynomial obtained within the space-time cell I_i^n. Any reconstruction procedure can be implemented, however, in this work we use the Weighted Essentially Non-Oscillatory (WENO) reconstruction method described in [11,12], where the central, left-sided and right-sided stencils are involved, with respective weights $\omega_C = 10^5$, $\omega_L = \omega_R = 1$ and the coefficients $\varepsilon = 10^{-14}$, $r = 9$.
2. Compute the approximation of high-order derivatives in time and space of the state and Jacobian matrices as well. For this purpose, we use the following approach.
   Let $\mathbf{M}$ be a function, which may represent the state function $\mathbf{Q}$ and the Jacobian matrices $\mathbf{A}$ and $\mathbf{B}$ as well.
   - Then, for obtaining $\mathbf{M}_x^{(l)}(\xi, \tau_j)$, we first interpolate the function $\mathbf{M}$ on the nodes (ξ_m, τ_j) with j fix and varying $m = 1, ..., n_S$. So an interpolation function $\tilde{\mathbf{M}}(\xi, \tau_j)$ is obtained. Then, we are able to provide approximations of spatial derivatives of $\mathbf{M}(\xi, \tau_j)$ for any order l by using the spatial derivatives of $\tilde{\mathbf{M}}(\xi, \tau_j)$.
   - Similarly, to obtain $\mathbf{M}_t^{(l)}(\xi_m, \tau)$, we first interpolate the function $\mathbf{M}$ on the nodes (ξ_m, τ_j) with m fix and varying $j = 1, ..., n_T$. So an interpolation function $\tilde{\mathbf{M}}(\xi_m, \tau)$ is obtained. Then, we are able to provide approximations of temporal derivatives of $\mathbf{M}(\xi_m, \tau)$ for any order l by using the temporal derivatives of $\tilde{\mathbf{M}}(\xi_m, \tau)$.

   In B.2, is shown the form of these interpolation polynomials for the orders of accuracy considered in this paper.
3. Update $\mathbf{Q}_i$ at every (ξ_m, τ_j) by using
   $$\mathbf{Q}_i(\xi_m, \tau_j) = \mathbf{W}_i(\xi_m) - \sum_{k=1}^{M} \frac{(-\tau_j)^k}{k!} \tilde{\mathbf{G}}^{(k)}(\xi_m, \tau_j) \,, \tag{43}$$
   where $\tilde{\mathbf{G}}^{(k)} = \tilde{\mathbf{G}}^{(k)}(\mathbf{Q}_i, ..., \partial_x^{(k)}\mathbf{Q}_i, \mathbf{A}_t^{(l)}, .., \mathbf{A}_x^{(l)}, .., \mathbf{B}_t^{(l)}, ..., \mathbf{B}_x^{(l)}, ...)$ is given by (28). The derivatives of the state function and matrices are evaluated at (ξ_m, τ_j) and computed in the previous step. The Eq. (43) corresponds to the implicit Taylor series expansion in [39] with the difference that $\tilde{\mathbf{G}}^{(k)}$ is a simplification of the conventional Cauchy-Kowalewskaya functional.
   For solving (43), we use the following nested Picard iteration procedure.
   $$\mathbf{Q}_i^{s+1} = \mathbf{W}_i(\xi_m) - \sum_{k=1}^{M} \frac{(-\tau_j)^k}{k!} \sum_{l=2}^{k} \mathbf{M}_l - \sum_{k=1}^{M} \frac{(-\tau_j)^k}{k!} \mathbf{B}^{k-1}(\mathbf{Q}_i^s)\mathbf{S}\left(\mathbf{Q}_i^{s+1}\right) \,, \tag{44}$$
   where s is an iteration index and $\mathbf{M}_l$ comes from the Proposition 3.4. We have omitted the arguments of $\mathbf{Q}_i(\xi_m, \tau_j)$.
   Notice that for solving (44), we build an algebraic system, which has the form
   $$\mathcal{H}(\mathbf{Y}) = \mathbf{Y} - \mathbf{W}_i(\xi_m) + \sum_{k=1}^{M} \frac{(-\tau_j)^k}{k!} \sum_{l=2}^{k} \mathbf{M}_l + \sum_{k=1}^{M} \frac{(-\tau_j)^k}{k!} \mathbf{B}^{k-1}\left(\mathbf{Q}_i^s(\xi_m, \tau_j)\right)\mathbf{S}(\mathbf{Y}) \,, \tag{45}$$
   then by solving for $\mathbf{Y}$ we get $\mathbf{Q}_i^{s+1}$ as $\mathbf{Y} = \mathbf{Q}_i^{s+1}$. This is carried out in one step, that is $\mathbf{Q}_i^{s+1} = \mathbf{Q}_i^s - \delta$, where δ is the solution to $\mathcal{J}(\mathbf{Q}_i^s)\delta = \mathcal{H}(\mathbf{Q}_i^s)$, where
   $$\mathcal{J}(\mathbf{Y}) = \mathbf{I} + \sum_{k=1}^{M} \frac{(-\tau_j)^k}{k!} \mathbf{B}^{k-1}(\mathbf{Q}_i^s(\xi_m, \tau_j))\mathbf{B}(\mathbf{Y}) \,, \tag{46}$$
   is the Jacobian matrix of $\mathcal{H}(\mathbf{Y})$ with respect to $\mathbf{Y}$. Once the $\mathbf{Q}_i^{s+1}$ has been updated then we set $s \leftarrow s + 1$ and update

$\mathbf{Q}_i^{s+1}$. The update is carried out M times, where $M+1$ corresponds to the order of accuracy. Notice that algebraic equations with a similar structure to $\mathbf{Y}$ has to be solved for any order of accuracy. So, the size of $\mathbf{Y}$ does not depend on the accuracy, indeed, this corresponds to the state $\mathbf{Q}$ and so, $\mathbf{Y}$ and $\mathcal{H}$ have size m and consequently $\mathbf{J}$ is a matrix of $m \times m$.

The global loop, it is the steps 2 and 3 listed above, are repeated M times. From experiments, not shown here, the result does not vary in terms of accuracy if a stop criterion, based on the tolerance for the relative error between subsequent approximations, is implemented.

Once $\mathbf{Q}_i$ is computed for each cell I_i^n, the numerical flux and source terms can be easily evaluated. In B.3, is shown the form in which the integrals in (4) are evaluated.

This completes the description of the proposed strategy for obtaining the predictor within the computational cell I_i^n by using the implicit GRP approach.

4. Numerical results

In this section we shall consider several tests aimed at assessing the accuracy and performance of the present scheme. The time step, Δt, will be computed by using the well-known CFL condition

$$\Delta t = C_{cfl}\frac{\Delta x}{\lambda_{abs}}\ , \tag{47}$$

where $\lambda_{abs} = \max_i(\max_j(|\lambda_j(\mathbf{Q}_i^n)|))$, here $\lambda_j,\ j = 1, ..., m$ are the eigenvalues of the Jacobian matrix of $\mathbf{F}$ evaluated at $\mathbf{Q}_i^n$, the data at each cell I_i^n and the maximum is taken over all cells I_i^n. On the other hand, to assess empirically the convergence rate, we are going to use the norm

$$||\mathbf{Q}-\mathbf{Q}^e||_p^p = \sum_{i=1}^{N}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} |\mathbf{W}_i(x) - \mathbf{Q}^e(x, t_{out})|^p dt\ , \tag{48}$$

where $\mathbf{Q}^e(x,\ t)$ is the exact solution, $\mathbf{W}_i(x)$ is the reconstruction polynomial within the interval $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ obtained at the output time of the global simulation. In this paper we are going to use (48) with $p = 1,\ p = 2$ and the maximum norm given by

$$||\mathbf{Q}-\mathbf{Q}^e|| = \max_i\{\max_{x\in[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]} |\mathbf{W}_i(x) - \mathbf{Q}^e(x, t_{out})|\}\ . \tag{49}$$

Here, t_{out} stands by the output time of simulations.

4.1. A linear system of hyperbolic balance laws

Here, we consider the linear system in [29], given by

$$\begin{aligned}\partial_t\mathbf{Q}(x,t) + \mathbf{A}\partial_x(\mathbf{Q}(x,t)) = \mathbf{B}\mathbf{Q}(x,t)\ , x\in[0,1]\ ,\\ \mathbf{Q}(x,0) = \begin{bmatrix}\sin(2\pi x)\\ \cos(2\pi x)\end{bmatrix}\ ,\end{aligned} \tag{50}$$

where

$$\mathbf{A} = \begin{bmatrix}0 & \lambda\\ \lambda & 0\end{bmatrix}, \mathbf{B} = \begin{bmatrix}\beta & 0\\ 0 & \beta\end{bmatrix}. \tag{51}$$

The problem is endowed with periodic boundary conditions. This system has the exact solution

$$\mathbf{Q}^e(x,t) = \frac{e^{\beta t}}{2}\begin{bmatrix}\Phi(x,t)+\Psi(x,t)\\ \Phi(x,t)-\Psi(x,t)\end{bmatrix}, \tag{52}$$

where

$$\begin{aligned}\Phi(x,t) = \sin(2\pi(x-\lambda t)) + \cos(2\pi(x-\lambda t))\ ,\\ \Psi(x,t) = \sin(2\pi(x+\lambda t)) - \cos(2\pi(x+\lambda t))\ .\end{aligned} \tag{53}$$

Here we consider $\lambda = 1$ and $\beta = -1$. This is a simple test aimed at evaluating the accuracy of the present scheme. As can be seen in Table 3, the expected theoretical orders of accuracy are achieved.

4.2. A system of non-linear hyperbolic balance laws

Here we assess the present methods, applied to the non-linear system

$$\begin{aligned}\partial_t\mathbf{Q} + \partial_x\mathbf{F}(\mathbf{Q}) = \mathbf{S}(\mathbf{Q})\ ,\\ \mathbf{Q}(x,0) = \begin{bmatrix}\sin(2\pi x)\\ \cos(2\pi x)\end{bmatrix}\ ,\end{aligned} \tag{54}$$

where $\mathbf{F}(\mathbf{Q})$ and $\mathbf{S}(\mathbf{Q})$ are given by

$$\mathbf{F}(\mathbf{Q}) = \begin{bmatrix}\frac{1}{9}\left(\frac{5}{2}u^2 + v^2 - uv\right)\\ \frac{1}{9}\left(4uv - u^2 + \frac{1}{2}v^2\right)\end{bmatrix}, \quad \mathbf{S}(\mathbf{Q}) = \begin{bmatrix}\beta\left(\frac{2u-v}{3}\right)^2\\ -\beta\left(\frac{2u-v}{3}\right)^2\end{bmatrix}, \tag{55}$$

where $\beta \le 0$ is a constant value, see [39]. The exact solution is given by

$$\begin{aligned}u(x,t) = w_1(x,t) + w_2(x,t)\ ,\\ v(x,t) = 2w_1(x,t) - w_2(x,t)\ ,\end{aligned} \tag{56}$$

where w_1 and w_2 are the solutions to

$$\begin{aligned}\partial_t w_1 + w_1\partial_x(w_1) = 0\ ,\\ \partial_t w_2 + w_2\partial_x(w_2) = \beta w_2^2\ ,\end{aligned} \tag{57}$$

where the initial condition for each equation is

$$\begin{aligned}w_1(x,0) = \frac{\sin(2\pi x) + cos(2\pi x)}{3}\ ,\\ w_2(x,0) = \frac{2sin(2\pi x) - cos(2\pi x)}{3}\ .\end{aligned}$$

Notice that system (57) requires the solution of the Burgers equation with a non linear source term, this solution is reported in [39], which is computed numerically. Table 4, shows the empirical orders of accuracy and the CPU times. Comparing with CPU times of the implicit Taylor series expansion and conventional Cauchy-Kowalewskaya procedure in [39], we observe that the present scheme depicts important improvements in the performance. An improvement of one order of magnitude compared with the strategies in [39], is obtained. See Appendix A. On the other hand, since the exact solution in [39] is also an approximation, we observe a penalization in the convergence rate as the accuracy increases. It is more evident in the fifth order case.

4.3. The LeVeque and Yee test

Here, we apply our schemes to the well-known and challenging scalar test problem proposed by LeVeque and Yee [23], given by

$$\partial_t q(x,t) + \partial_x q(x,t) = \beta q(x,t)(q(x,t)-1)(q(x,t)-\tfrac{1}{2})\ . \tag{58}$$

We solve this PDE on the computational domain [0,1] with transmissive boundary conditions and the initial condition given by

$$q(x,0) = \begin{cases}1\ , x < 0.3\ ,\\ 0\ , x > 0.3\ .\end{cases} \tag{59}$$

The solution on the characteristic curves satisfies the ordinary differential equation $\frac{d(x(t),t)}{dt} = \beta q(x(t),t)(q(x(t),t)-1)(q(x(t),t)-\frac{1}{2})$, which has two stable solutions $q \equiv 0$ and $q \equiv 1$ and one unstable solution in $q \equiv \frac{1}{2}$ where any solution tries to away from this. Similarly, any solution associated to characteristic curves necessarily must converge to one of the two stable solutions. On the other hand, a numerical scheme which is not able to solve stiff source terms, may blow up or introduce an excessive numerical diffusion and so the numerical solution, following characteristic curves, converges to the wrong stable solution. This penalizes the right propagation. Fig. 2 shows the comparison between

Table 3
Linear system. Output time $t_{out} = 1$ with $C_{cfl} = 0.9,\ \beta = -1,\ \lambda = 1$.

Theoretical order : 2							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
8	-	$2.45e-02$	-	$1.33e-02$	-	$1.74e-02$	0.0064
16	2.35	$4.81e-03$	2.38	$2.55e-03$	2.55	$2.97e-03$	0.0114
32	1.67	$1.52e-03$	2.85	$3.54e-04$	2.34	$5.84e-04$	0.0287
64	2.95	$1.96e-04$	3.71	$2.71e-05$	3.47	$5.29e-05$	0.0853
128	1.49	$6.96e-05$	1.83	$7.62e-06$	1.91	$1.41e-05$	0.3395
Theoretical order : 3							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
8	-	$1.52e-02$	-	$1.07e-02$	-	$1.15e-02$	0.0116
16	2.85	$2.11e-03$	2.97	$1.36e-03$	2.94	$1.50e-03$	0.0352
32	3.01	$2.62e-04$	3.03	$1.66e-04$	3.02	$1.85e-04$	0.1096
64	3.05	$3.17e-05$	3.04	$2.02e-05$	3.05	$2.24e-05$	0.4506
128	3.01	$3.93e-06$	3.01	$2.50e-06$	3.01	$2.78e-06$	2.0173
Theoretical order : 4							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
8	-	$7.69e-03$	-	$4.80e-03$	-	$5.26e-03$	0.0358
16	2.92	$1.02e-03$	3.15	$5.42e-04$	3.10	$6.14e-04$	0.1280
32	3.64	$8.15e-05$	3.67	$4.26e-05$	3.67	$4.82e-05$	0.4703
64	3.87	$5.56e-06$	3.87	$2.92e-06$	3.87	$3.30e-06$	1.7537
128	3.95	$3.59e-07$	3.95	$1.89e-07$	3.95	$2.14e-07$	6.9353
Theoretical order : 5							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
8	-	$1.49e-03$	-	$6.42e-04$	-	$7.65e-04$	0.1270
16	4.85	$5.16e-05$	4.88	$2.18e-05$	4.89	$2.59e-05$	0.3801
32	4.96	$1.66e-06$	4.96	$6.99e-07$	4.96	$8.29e-07$	1.4746
64	4.99	$5.23e-08$	4.99	$2.19e-08$	4.99	$2.60e-08$	5.7874
128	5.00	$1.64e-09$	4.99	$6.88e-010$	5.00	$8.16e-010$	22.4536

Table 4
Non-linear system. Output time $t_{out} = 0.1$ with $C_{cfl} = 0.9,\ \beta = -1$.

Theoretical order : 2							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
32	-	$1.95e-02$	-	$5.45e-03$	-	$7.81e-03$	0.0061
64	1.53	$6.75e-03$	2.12	$1.25e-03$	1.94	$2.04e-03$	0.0201
128	1.75	$2.01e-03$	2.30	$2.54e-04$	2.18	$4.49e-04$	0.0490
256	1.80	$5.79e-04$	2.18	$5.62e-05$	2.15	$1.01e-04$	0.1703
512	1.38	$2.23e-04$	2.13	$1.29e-05$	2.06	$2.44e-05$	0.7851
Theoretical order : 3							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
32	-	$2.00e-02$	-	$2.88e-03$	-	$5.17e-03$	0.0139
64	2.47	$3.63e-03$	2.69	$4.45e-04$	2.57	$8.67e-04$	0.0668
128	2.74	$5.42e-04$	2.92	$5.90e-05$	2.85	$1.20e-04$	0.1935
256	2.84	$7.59e-05$	2.95	$7.62e-06$	2.93	$1.57e-05$	0.7559
512	2.95	$9.83e-06$	2.99	$9.62e-07$	2.98	$2.00e-06$	2.9953
Theoretical order : 4							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
32	-	$2.44e-02$	-	$3.33e-03$	-	$6.09e-03$	0.0609
64	3.07	$2.90e-03$	3.53	$2.89e-04$	3.24	$6.46e-04$	0.2186
128	3.88	$1.97e-04$	4.14	$1.64e-05$	4.00	$4.03e-05$	0.7516
256	4.28	$1.01e-05$	4.40	$7.80e-07$	4.34	$1.99e-06$	3.0089
512	4.18	$5.58e-07$	4.33	$3.87e-08$	4.33	$9.89e-08$	11.5548
Theoretical order : 5							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
32	-	$8.80e-03$	-	$8.56e-04$	-	$1.82e-03$	0.1879
64	3.48	$7.91e-04$	4.07	$5.09e-05$	3.82	$1.29e-04$	0.6927
128	4.46	$3.60e-05$	4.66	$2.02e-06$	4.58	$5.37e-06$	2.4518
256	4.81	$1.29e-06$	4.72	$7.65e-08$	4.82	$1.90e-07$	9.6558
512	4.46	$5.87e-08$	3.97	$4.86e-09$	4.41	$8.90e-09$	38.51853

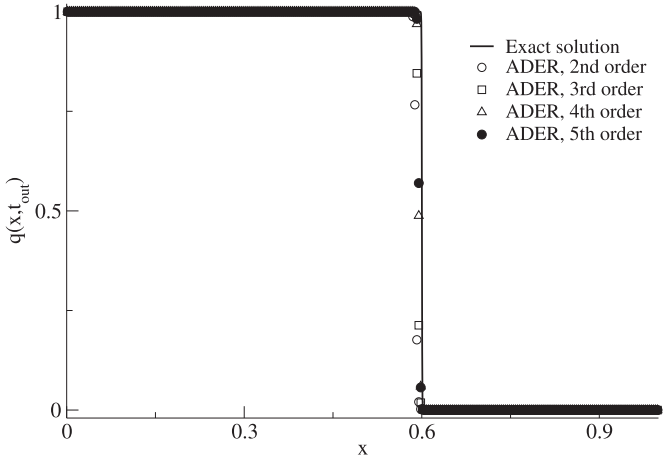

Fig. 2. Leveque and Yee test. We have used 300 cells, $C_{cfl} = 0.2$, $t_{out} = 0.3$, $\beta = -1000$.

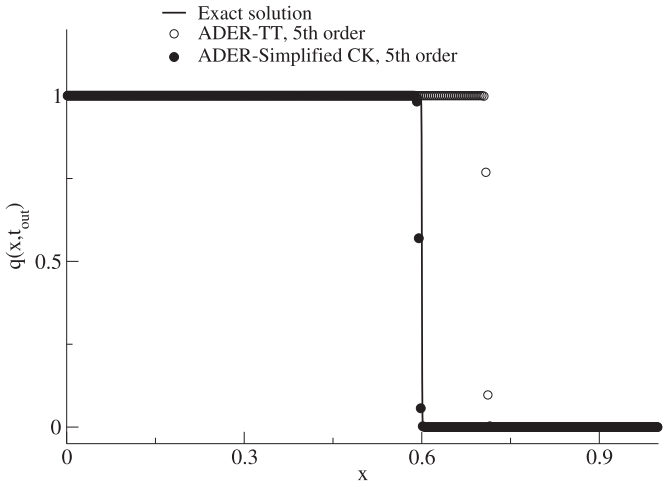

Fig. 3. Leveque and Yee test. Proposed scheme against ADER-TT with the conventional Cauchy-Kowalewskaya procedure. We have used 300 cells, $C_{cfl} = 0.2$, $t_{out} = 0.3$, $\beta = -1000$.

the exact solution and the numerical approximations provided by the present scheme of second, third, fourth and fifth orders of accuracy. The figure shows a good agreement for $\beta = -10000$ at $t_{out} = 0.3$, which correspond to the stiff regime. We have used 300 cells. For this particular test, we use $C_{cfl} = 0.2$ which provides good results, notice this is not the case for schemes using explicit GRP solvers, as can be seen in the Fig. 3, where the comparison between ADER schemes of fifth order of accuracy using the proposed GRP solver and the conventional TT solver, is shown. Despite this is expected because ADER-TT is not designed for dealing with stiff source terms, it helps us to show the gaining of using the local implicit Taylor series expansion formulation. This test illustrates the ability of the present scheme for solving hyperbolic balance laws with stiff source terms. Despite the *CFL* coefficient in this test is less than that used in [39] we remark that this is still in the range of functional values.

4.4. The Euler equations

Now let us consider the Euler equations, given by

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, \quad \mathbf{F}(\mathbf{Q}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix}, \tag{60}$$

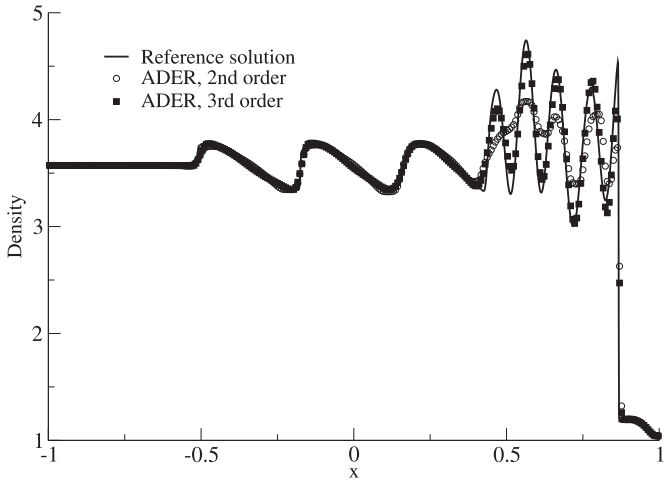

Fig. 4. The Shu-Osher test. Otput time $t_{out} = 0.47$, 300 cells and $CFL = 0.5$.

where the pressure p is related with the conserved variables through the equation

$$p = (\gamma - 1)(E - \frac{\rho u^2}{2}) , \tag{61}$$

for an ideal gas $\gamma = 1.4$. Notice that, the choice of the initial condition given by the functions

$$\begin{aligned} &\rho(x,0) = 1 + 0.2\sin(2\pi x) , \\ &u(x,0) = 1, \\ &p(x,0) = 2, \end{aligned} \tag{62}$$

provides the exact solution for the system (60), which corresponds to the set of functions

$$\begin{aligned} &\rho(x,t) = 1 + 0.2\sin(2\pi (x-t)) , \\ &u(x,t) = 1 , \\ &p(x,t) = 2 . \end{aligned} \tag{63}$$

Notice that, the variables ρ, u, p correspond to the non-conservative variables, the corresponding translation to conserved variables needs to be done. This test has a complex eigenstructure, which is a challenge for numerical methods. Table 5, shows the results of the empirical convergence rate assessment for the density variable ρ, at $t_{out} = 1$ and $C_{cfl} = 0.9$, we observe that the scheme achieves the expected theoretical orders of accuracy.

4.5. The Shu and Osher test

Here, we consider the test problem, first time proposed by Shu and Osher in [32], which is given by (60) and the initial condition given, in terms of non-conserved variables $\mathbf{W} = [\rho, u, p]^T$, as

$$\mathbf{W}(x,0) = \begin{cases} (3.8571, 2.6294, 10.333) & , x < -0.8 , \\ (1 + \sin(5\pi x), 0, 1) & , x \geq -0.8 . \end{cases} \tag{64}$$

The problem is solved on $[-1, 1]$ up to the output time t_{out}, see [18] for further details. Fig. 4 shows a comparison between a reference solution and numerical approximations. The reference solution has been obtained with the scheme of third order using 2000 cells. The numerical results correspond to second and third orders of accuracy for which we have used 300 cells, $C_{cfl} = 0.5$ and $t_{out} = 0.47$. This test illustrates the ability of the present scheme for solving complex fluids, a good agreement is observed for the scheme of second and third order of accuracy on the smooth region, whereas, the third order scheme present a better performance in both the smooth region and the high frequency region as well.

Table 5
Euler equations.Output time $t_{out} = 1$ with $C_{cfl} = 0.9$.

Theoretical order : 2							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
8	-	$1.51e-01$	-	$1.05e-01$	-	$1.14e-01$	0.0119
16	1.22	$6.45e-02$	1.76	$3.09e-02$	1.59	$3.79e-02$	0.0401
32	1.43	$2.40e-02$	1.62	$1.01e-02$	1.65	$1.21e-02$	0.1569
64	1.49	$8.50e-03$	1.90	$2.71e-03$	1.79	$3.49e-03$	0.4242
128	1.53	$2.95e-03$	2.07	$6.46e-04$	1.85	$9.69e-04$	1.6852
Theoretical order : 3							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
8	-	$8.33e-02$	-	$5.18e-02$	-	$5.83e-02$	0.0356
16	2.55	$1.43e-02$	2.58	$8.68e-03$	2.59	$9.68e-03$	0.1292
32	2.90	$1.91e-03$	2.93	$1.14e-03$	2.93	$1.27e-03$	0.5785
64	2.98	$2.42e-04$	2.99	$1.44e-04$	2.99	$1.60e-04$	2.0123
128	3.00	$3.03e-05$	3.00	$1.80e-05$	3.00	$2.00e-05$	7.3028
Theoretical order : 4							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
8	-	$7.49e-02$	-	$4.75e-02$	-	$5.30e-02$	0.1776
16	4.17	$4.17e-03$	4.24	$2.51e-03$	4.25	$2.79e-03$	0.5581
32	4.40	$1.97e-04$	4.49	$1.12e-04$	4.49	$1.25e-04$	2.0833
64	4.27	$1.02e-05$	4.24	$5.91e-06$	4.24	$6.59e-06$	7.4370
128	4.12	$5.88e-07$	4.08	$3.50e-07$	4.08	$3.90e-07$	30.9521
Theoretical order : 5							
Mesh	L_∞ - ord	L_∞- err	L_1 - ord	L_1 - err	L_2 - ord	L_2 - err	CPU
8	-	$1.24e-02$	-	$7.62e-03$	-	$8.47e-03$	0.3804
16	4.75	$4.59e-04$	4.80	$2.73e-04$	4.80	$3.04e-04$	1.6927
32	4.95	$1.48e-05$	4.96	$8.76e-06$	4.96	$9.76e-06$	6.4005
64	4.99	$4.66e-07$	4.99	$2.75e-07$	4.99	$3.07e-07$	23.9378
128	5.00	$1.46e-08$	5.00	$8.62e-09$	5.00	$9.59e-09$	88.51398

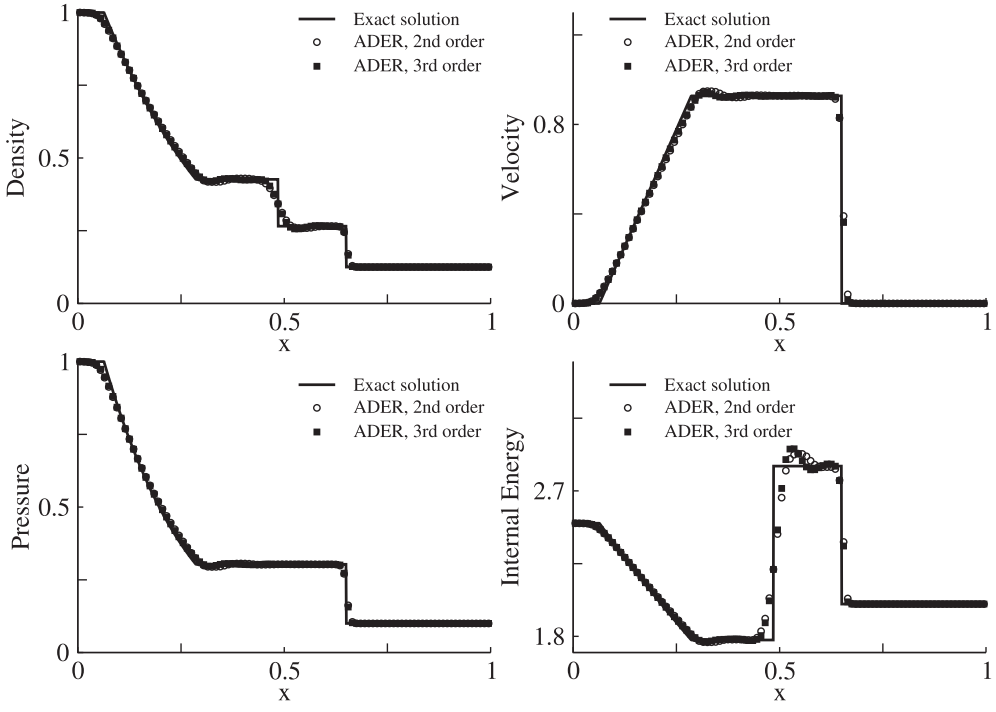

Fig. 5. The Sod test. Otput time $t_{out} = 0.2$, 100 cells and $CFL = 0.7$.

4.6. The Sod test problem

Here, we consider the test problem, first time proposed by Sod in [33], which is given by (60) and the initial condition given, in terms of non-conserved variables $\mathbf{W} = [\rho, u, p]^T$, as

$$\mathbf{W}(x,0) = \begin{cases} (1,0,1) & ,x < 0.3\,, \\ (1.125,0,0.1) & ,x \geq 0.3\,. \end{cases} \tag{65}$$

The problem is solved on $[-1, 1]$ up to the output time $t_{out} = 0.2$. Fig. 5 shows a comparison between the exact solution and numerical approximations. The numerical results correspond to second and third orders of accuracy for which we have used 100 cells and $C_{cfl} = 0.7$. In this test a rarefaction wave travels to the left and a shock wave travels on the right and an intermediate contact wave moves on the right also. The shock wave is solved in three cells and the contact wave for the density is solved in three cells. The rarefaction is lightly rounded on the left and right corners. Despite, the internal energy depicts an overshot after the contact wave, we observe that the constant state connecting the contact and shock waves is well solved. This is the performance observed in first order schemes able to capture discontinuous wave propagation [37].

5. Conclusions

In this paper, a simplified Cauchy-Kowalewskaya procedure has been proposed. The strategy uses not only the spatial derivatives of the data but also the derivatives of the Jacobian matrices in both space and time. The simplification allows us to propose a recursive formula which requires the ability of obtaining time and space derivatives of the state as well as matrices. This is achieved by using interpolations within a suitable arrangement of nodal points, which allows us to extract the information for flux and source term evaluations in a straightforward manner. This method is implemented in the context of GRP solvers based on implicit Taylor series expansions. The solver in [39] uses the same elements, that is, implicit Taylor series expansions and Cauchy-Kowalewskaya procedure. However, in the present approach the Taylor series expansion is used only once and the evolution of the space derivatives via Taylor series is not required. Despite, in both approaches, that is in [39] and the present one, the solution of an algebraic equation is required, that in [39] increases the number of unknowns as the accuracy increases, whereas, the number of unknowns in the present approach remains constant when the accuracy increases. We have implemented the GRP solver in the context of ADER methods and several tests reported in the literature have been solved. An empirical convergence rate assessment has been done for some of them. We have proved that the complexity of the simplified Cauchy-Kowalewskaya procedure presented here grows up linearly with the order of accuracy and we have observed from empirical comparisons that the performance of the present scheme is at least one order of magnitude cheaper than schemes in [39]. Furthermore, the expected theoretical orders of accuracy have been achieved up to fifth order of accuracy. We have solved the Osher-Shu and the Sod test, where the right propagation of waves has been captured by the proposed scheme describing very well strong-discontinuities as shock and contact waves. The extension to hyperbolic systems in 2D and 3D is the subject of ongoing research.

Declaration of Competing Interest

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

The authors declare that they have no financial interests/personal relationships which may be considered as potential competing interests.

CRediT authorship contribution statement

Gino I. Montecinos: Formal analysis, Writing - original draft, Writing - review & editing. **Dinshaw S. Balsara:** Conceptualization, Methodology.

Acknowledgements

G.M thanks to the *National chilean Fund for Scientific and Technological Development*, FONDECYT, in the frame of the research project for Initiation in Research, number 11180926.

Appendix A. A comparison between the complexity of floating point operations count between the conventional and simplified Cauchy-Kowalewskaya procedure

Since the code is not optimized, we do not carry out a floating point operations count of the enter program. Instead, we compare the classical and the simplified procedure in terms of the time complexity provided by the number of operations involved in the computation of time derivatives. This will give us information about the improvements of the present scheme.

For computing the time derivatives we use the expressions (37) and (38) where the matrices $\mathbf{C}(k, l)$ in (29) are the key. In practical approximations we encourage to compute the matrices in (29) and then store them for later uses in (37). Also, it is recommended to store the time derivatives and use them in the computation of derivatives of subsequent orders. We argue that storing and memory access is more efficient than computing long expressions at any time. The same argument is used for analyzing the performance of the conventional Cacuchy-Kowalewskaya procedure.

If one follows the suggested storage and access strategy for computing time derivatives, the number of operations are additive. So, if we denote by $T_{t,k}(m)$ the number of operations to get the time derivative k via (37) and $T_C(m)$ the number of operations to get the matrices C via (29), then the total number of operations to generate approximations of time derivatives is $T_{t,k}(m) + T_C(m)$.

The operations $T_{t,k}(m)$ in the case of time derivatives of order k is $O(km^2)$, whereas (29) in the case of the first time derivative requires $T_C(m) = O(m^2)$ operations. For $k > 1$, $\mathbf{C}(k, l)$ needs $T_C(m) = 3m^3$ in the case of $k = l$ and $m^2 + 3(k-l)m^3$ in the case $l < k$, so we can say that for high order, $T_C(m) = O(km^3)$. Notice that in (29) it is also required the matrix $\mathbf{D}$, which has been already computed. From (21) the number of operations involved in the computation of $\mathbf{D}$ is $O(m^2)$. Therefore, after collecting all the operation orders, the total number of operations for obtaining the approximations in the simplified Cauchy-Kowalewskaya procedure is $O(km^3)$.

Here, the number of operations for obtaining spatial and temporal gradients of matrices has been neglected because they are carried out by interpolations, where no matrix multiplication is involved, only additions are required for such task where the number of operations is of order $O(m^2)$.

Now, let us analyze the number of operations involved in the computation via the classical Cauchy-Kowalewskaya procedure. Let us begin with the first-order time derivative. In a component-wise, it has the form

$$\partial_t \mathbf{Q}_i = -\sum_{j=1}^{m} \mathbf{A}_{i,j}(\mathbf{Q}) \frac{\partial \mathbf{Q}_j}{\partial x} + \mathbf{S}_i \,, \tag{A.1}$$

so, we can see that the computation of this derivative involves $m + 2m^2$ operations, so the asymptotic order is $O(m^2)$.

Now, let us analyze the number of operations for the second-order time derivative. In a component-wise, it has the form

$$\partial_t^{(2)} \mathbf{Q}_i = -\sum_{j=1}^{m} \Big(\sum_{k=1}^{m} \frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k} \frac{\partial \mathbf{Q}_k}{\partial t} \frac{\partial \mathbf{Q}_j}{\partial x} + \mathbf{A}_{i,j}(\mathbf{Q}) \frac{\partial^2 \mathbf{Q}_j}{\partial x \partial t} \Big) + \sum_{j=1}^{m} \mathbf{B}_{i,j}(\mathbf{Q}) \frac{\partial \mathbf{Q}_j}{\partial t} \,, \tag{A.2}$$

where

$$\partial_x(\partial_t \mathbf{Q}_j) = -\sum_{l=1}^{m}\Big(\sum_{p=1}^{m}\frac{\partial \mathbf{A}_{j,l}(\mathbf{Q})}{\partial \mathbf{Q}_p}\frac{\partial \mathbf{Q}_p}{\partial x}\frac{\partial \mathbf{Q}_l}{\partial x} + \mathbf{A}_{j,l}(\mathbf{Q})\frac{\partial^2 \mathbf{Q}_l}{\partial x^2}\Big) + \sum_{l=1}^{m}\mathbf{B}_{j,l}(\mathbf{Q})\frac{\partial \mathbf{Q}_l}{\partial x}\,. \tag{A.3}$$

Here, we have several alternatives to obtain the second time derivative (A.2). We identify at least two options. The first one, is to compute and store the mixed space and time derivative (A.3), which involves $T_{xt}(m) = 4m^2 + 3m^3$ operations, and then use the stored values. In such a way, no extra operations are required so the number of operation for evaluating (A.2) is $4m^2 + 3m^3$. However, we must consider the operation for $T_{xt}(m)$ and $T_t(m)$, since the expressions have been precomputed already the total number of operations are additives. In this case the total iterations are $m + 10m^2 + 6m^3$. That is $O(m^3)$.

The second option, is to replace the mixed space-time derivative (A.3) into the full expression (A.2), obtaining

$$\begin{aligned}\partial_t^{(2)}\mathbf{Q}_i = &-\sum_{j=1}^{m}\sum_{k=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k}\frac{\partial \mathbf{Q}_k}{\partial t}\frac{\partial \mathbf{Q}_j}{\partial x}\\ &+\sum_{j=1}^{m}\sum_{l=1}^{m}\sum_{p=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\frac{\partial \mathbf{A}_{j,l}(\mathbf{Q})}{\partial \mathbf{Q}_p}\frac{\partial \mathbf{Q}_p}{\partial x}\frac{\partial \mathbf{Q}_l}{\partial x}\\ &+\sum_{j=1}^{m}\sum_{l=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\mathbf{A}_{j,l}(\mathbf{Q})\frac{\partial^2 \mathbf{Q}_l}{\partial x^2}\\ &-\sum_{j=1}^{m}\sum_{l=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\mathbf{B}_{j,l}(\mathbf{Q})\frac{\partial \mathbf{Q}_l}{\partial x}\\ &+\sum_{j=1}^{m}\mathbf{B}_{i,j}(\mathbf{Q})\frac{\partial \mathbf{Q}_j}{\partial t}\end{aligned} \tag{A.4}$$

and then the number of operations becomes $3m^2 + 12m^3 + 5m^4$, so the asymptotic order in this case is $O(m^4)$.

Let us obtain the order of operations involved into the computation of the third-order time derivative. This is obtained by differentiating in time (A.4), after some manipulations it yields

$$\begin{aligned}\partial_t^{(3)}\mathbf{Q}_i = -\sum_{j=1}^{m}\Bigg[&\sum_{k=1}^{m}\sum_{l=1}^{m}\frac{\partial^2 \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k \partial \mathbf{Q}_l}\frac{\partial \mathbf{Q}_l}{\partial t}\frac{\partial \mathbf{Q}_k}{\partial t}\frac{\partial \mathbf{Q}_j}{\partial x}\\ &+\sum_{k=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k}\frac{\partial^2 \mathbf{Q}_k}{\partial t^2}\frac{\partial \mathbf{Q}_j}{\partial x} + 2\sum_{k=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k}\frac{\partial \mathbf{Q}_k}{\partial t}\frac{\partial^2 \mathbf{Q}_j}{\partial x \partial t}\\ &+\sum_{k=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\frac{\partial^3 \mathbf{Q}_j}{\partial x \partial t^2}\Bigg] + \sum_{j=1}^{m}\sum_{k=1}^{m}\frac{\partial \mathbf{B}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k}\frac{\partial \mathbf{Q}_k}{\partial t}\frac{\partial \mathbf{Q}_j}{\partial t}\\ &+\sum_{j=1}^{m}\mathbf{B}_{i,j}(\mathbf{Q})\frac{\partial^2 \mathbf{Q}_j}{\partial t^2}\,,\end{aligned} \tag{A.5}$$

with

$$\begin{aligned}\partial_x(\partial_t^{(2)}\mathbf{Q})_i = -\sum_{j=1}^{m}\Bigg[&\sum_{k=1}^{m}\sum_{l=1}^{m}\frac{\partial^2 \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k \partial \mathbf{Q}_l}\frac{\partial \mathbf{Q}_l}{\partial x}\frac{\partial \mathbf{Q}_k}{\partial t}\frac{\partial \mathbf{Q}_j}{\partial x}\\ &+\sum_{k=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k}\frac{\partial^2 \mathbf{Q}_k}{\partial x \partial t}\frac{\partial \mathbf{Q}_j}{\partial x} + 2\sum_{k=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k}\frac{\partial \mathbf{Q}_k}{\partial t}\frac{\partial^2 \mathbf{Q}_j}{\partial x^2}\\ &+\sum_{k=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\frac{\partial^3 \mathbf{Q}_j}{\partial x^2 \partial t}\Bigg] + \sum_{j=1}^{m}\sum_{k=1}^{m}\frac{\partial \mathbf{B}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k}\frac{\partial \mathbf{Q}_k}{\partial x}\frac{\partial \mathbf{Q}_j}{\partial t}\\ &+\sum_{j=1}^{m}\mathbf{B}_{i,j}(\mathbf{Q})\frac{\partial^2 \mathbf{Q}_j}{\partial x \partial t}\end{aligned} \tag{A.6}$$

and

$$\begin{aligned}\partial_x^{(2)}(\partial_t \mathbf{Q}_j) = -\sum_{l=1}^{m}\Big(&\sum_{p=1}^{m}\sum_{q=1}^{m}\frac{\partial^2 \mathbf{A}_{j,l}(\mathbf{Q})}{\partial \mathbf{Q}_p \partial \mathbf{Q}_q}\frac{\partial \mathbf{Q}_q}{\partial x}\frac{\partial \mathbf{Q}_p}{\partial x}\frac{\partial \mathbf{Q}_l}{\partial x}\\ &+\sum_{p=1}^{m}\frac{\partial \mathbf{A}_{j,l}(\mathbf{Q})}{\partial \mathbf{Q}_p}\frac{\partial^2 \mathbf{Q}_p}{\partial x^2}\frac{\partial \mathbf{Q}_l}{\partial x} + \sum_{p=1}^{m}\frac{\partial \mathbf{A}_{j,l}(\mathbf{Q})}{\partial \mathbf{Q}_p}\frac{\partial \mathbf{Q}_p}{\partial x}\frac{\partial^2 \mathbf{Q}_l}{\partial x^2}\\ &+\sum_{p=1}^{m}\frac{\partial \mathbf{A}_{j,l}(\mathbf{Q})}{\partial \mathbf{Q}_p}\frac{\partial \mathbf{Q}_p}{\partial x}\frac{\partial^2 \mathbf{Q}_l}{\partial x^2} + \mathbf{A}_{j,l}(\mathbf{Q})\frac{\partial^3 \mathbf{Q}_l}{\partial x^3}\Big)\\ &+\sum_{l=1}^{m}\sum_{p=1}^{m}\frac{\partial \mathbf{B}_{j,l}(\mathbf{Q})}{\partial \mathbf{Q}_p}\frac{\partial \mathbf{Q}_p}{\partial x}\frac{\partial \mathbf{Q}_l}{\partial x} + \sum_{l=1}^{m}\mathbf{B}_{j,l}(\mathbf{Q})\frac{\partial^2 \mathbf{Q}_l}{\partial x^2}\,.\end{aligned} \tag{A.7}$$

Notice that, if terms $\frac{\partial \mathbf{Q}}{\partial t}$, $\frac{\partial^3 \mathbf{Q}}{\partial x^2 \partial t}$, $\frac{\partial^2 \mathbf{Q}}{\partial x \partial t}$, $\frac{\partial^2 \mathbf{Q}}{\partial t^2}$ and $\frac{\partial^3 \mathbf{Q}}{\partial x \partial t^2}$ are already precomputed and just called in (A.5), then the number of operations is $O(m^4)$.

However, if we introduce (A.3) and (A.7) into (A.5), we obtain

$$\begin{aligned}\partial_t^{(3)}\mathbf{Q}_i = &\\ &-\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m}\frac{\partial^2 \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k \partial \mathbf{Q}_l}\frac{\partial \mathbf{Q}_l}{\partial t}\frac{\partial \mathbf{Q}_k}{\partial t}\frac{\partial \mathbf{Q}_j}{\partial x}\\ &-\sum_{j=1}^{m}\sum_{k=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k}\frac{\partial^2 \mathbf{Q}_k}{\partial t^2}\frac{\partial \mathbf{Q}_j}{\partial x}\\ &-\sum_{j=1}^{m}\sum_{k=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k}\frac{\partial \mathbf{Q}_k}{\partial t}\frac{\partial^2 \mathbf{Q}_j}{\partial x \partial t}\\ &+\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m}\sum_{p=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_p}\frac{\partial \mathbf{Q}_p}{\partial t}\frac{\partial \mathbf{A}_{j,k}(\mathbf{Q})}{\partial \mathbf{Q}_l}\frac{\partial \mathbf{Q}_l}{\partial x}\frac{\partial \mathbf{Q}_k}{\partial x}\\ &+\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m}\sum_{p=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\frac{\partial^2 \mathbf{A}_{j,k}(\mathbf{Q})}{\partial \mathbf{Q}_l \partial \mathbf{Q}_p}\frac{\partial \mathbf{Q}_p}{\partial t}\frac{\partial \mathbf{Q}_l}{\partial x}\frac{\partial \mathbf{Q}_k}{\partial x}\\ &+\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\frac{\partial \mathbf{A}_{j,k}(\mathbf{Q})}{\partial \mathbf{Q}_l}\frac{\partial^2 \mathbf{Q}_l}{\partial x \partial t}\frac{\partial \mathbf{Q}_k}{\partial x}\\ &+\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\frac{\partial \mathbf{A}_{j,l}(\mathbf{Q})}{\partial \mathbf{Q}_l}\frac{\partial \mathbf{Q}_l}{\partial x}\frac{\partial^2 \mathbf{Q}_k}{\partial x \partial t}\\ &+\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_l}\frac{\partial \mathbf{Q}_l}{\partial t}\mathbf{A}_{j,k}(\mathbf{Q})\frac{\partial^2 \mathbf{Q}_k}{\partial x^2}\\ &+\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\frac{\partial \mathbf{A}_{j,k}(\mathbf{Q})}{\partial \mathbf{Q}_l}\frac{\partial \mathbf{Q}_l}{\partial t}\frac{\partial^2 \mathbf{Q}_k}{\partial x^2}\\ &+\sum_{j=1}^{m}\sum_{k=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\mathbf{A}_{j,k}(\mathbf{Q})\frac{\partial^3 \mathbf{Q}_k}{\partial x^2 \partial t}\\ &-\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m}\frac{\partial \mathbf{A}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_l}\frac{\partial \mathbf{Q}_l}{\partial t}\mathbf{B}_{j,k}(\mathbf{Q})\frac{\partial \mathbf{Q}_k}{\partial x}\\ &-\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\frac{\partial \mathbf{B}_{j,k}(\mathbf{Q})}{\partial \mathbf{Q}_l}\frac{\partial \mathbf{Q}_l}{\partial t}\frac{\partial \mathbf{Q}_k}{\partial x}\\ &-\sum_{j=1}^{m}\sum_{k=1}^{m}\mathbf{A}_{i,j}(\mathbf{Q})\mathbf{B}_{j,k}(\mathbf{Q})\frac{\partial^2 \mathbf{Q}_k}{\partial x \partial t}\\ &+\sum_{j=1}^{m}\sum_{k=1}^{m}\frac{\partial \mathbf{B}_{i,j}(\mathbf{Q})}{\partial \mathbf{Q}_k}\frac{\partial \mathbf{Q}_k}{\partial t}\frac{\partial \mathbf{Q}_j}{\partial t}\\ &+\sum_{j=1}^{m}\mathbf{B}_{i,j}(\mathbf{Q})\frac{\partial^2 \mathbf{Q}_j}{\partial t^2}\,,\end{aligned} \tag{A.8}$$

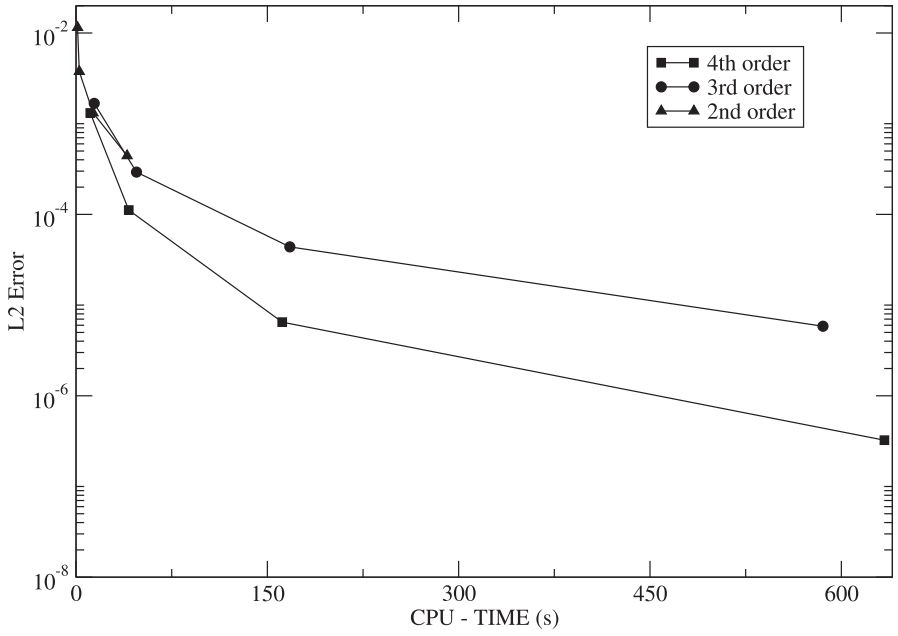

Fig. A1. Error vs CPU time for CITA: Error for the non-linear systems for orders from second to fourth order of accuracy.

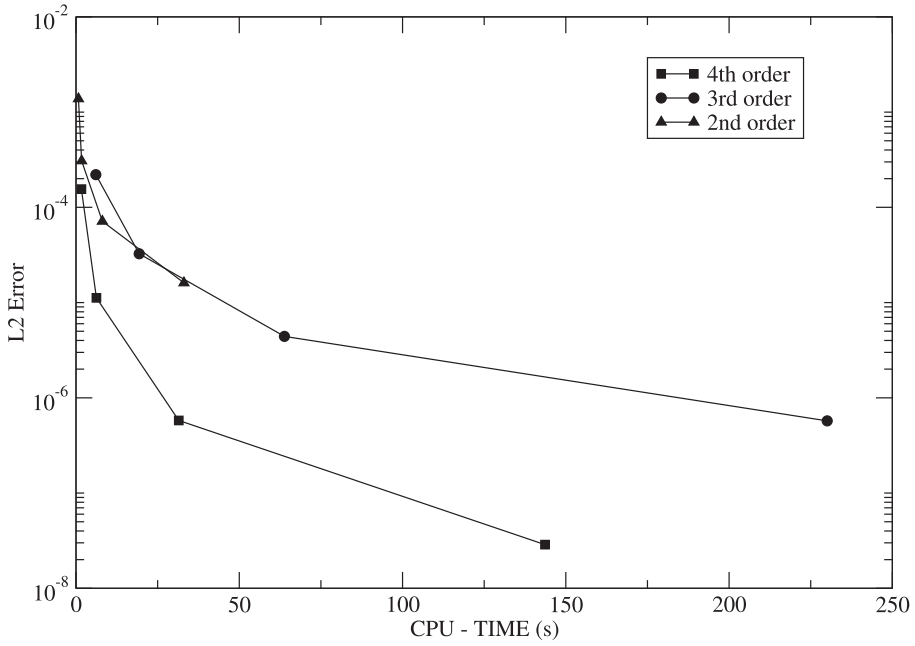

Fig. A2. Error vs CPU time for RITA: Error for the non-linear systems for orders from second to fourth order of accuracy.

which can be seen that it is $O(m^5)$ (here $\frac{\partial \mathbf{Q}}{\partial t}$, $\frac{\partial^3 \mathbf{Q}}{\partial x^2 \partial t}$, $\frac{\partial^2 \mathbf{Q}}{\partial x \partial t}$ and $\frac{\partial^2 \mathbf{Q}}{\partial t^2}$ are also computed previously and just used in (A.8)). Notice that the number operation increases by a factor of m if one of the mixed time and space derivatives commented above are explicitly computed in (A.8).

Here we are interested in $m > 1$, the system case. In the scalar case, $m = 1$, the number of operations in both the simplified and conventional Cauchy-Kowalewskaya procedures have similar asymptotic behavior. So, we have analyzed two strategies for computing the conventional Cauchy-Kowalewskaya in terms of the number of unknown attempting to reproduce an optimized coding strategy. We have observed that the present simplified version scales linearly with the order of accuracy whereas the conventional procedure scales exponentially with the order of accuracy. This shows an improvement in terms of the number of operations which is also observed in simulations, as we will illustrate later.

Despite the code structure of the proposed scheme and those in [39] are different, the GRP solvers should be one of the most time consuming processes in ADER implementations. So, any difference in the performance of GRP solvers should be evidenced into the global CPU time of the ADER schemes. So, in order to illustrate the improvement of the proposed GRP, enbedded into ADER schemes, with respect to the local implicit GRP solver presented in [39], we compare the L_1-error vs CPU time involved by solving the non-

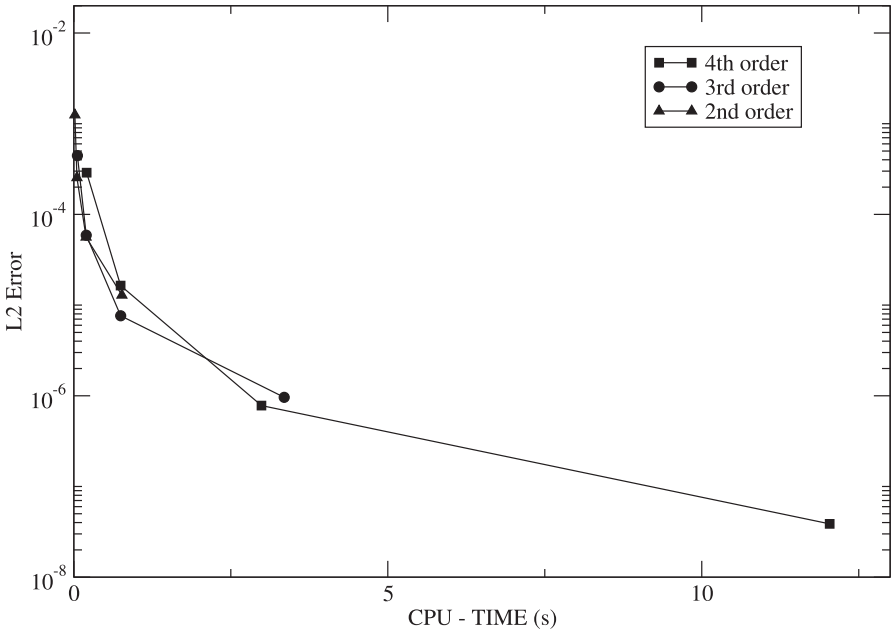

Fig. A3. Error vs CPU time for simplified CK: Error for the non-linear systems for orders from second to fourth order of accuracy.

linear system of Section 4.2, which has been solved in [39] too. We do not plot the fifth order because the accuracy for this particular test is sub-optimal.

Fig. A.6 shows the L_1-error vs CPU times for the ADER-HEOC-CITA approach in [39]. Fig. A.7 shows the L_1-error vs CPU times for the ADER-HEOC-RITA approach in [39]. Fig. A.8 shows the L_1-error vs CPU times for the simplified Cauchy-Kowalewskaya functional. From these figures, we observe two important issues. i) The present scheme is one order of magnitude cheaper than implicit formulation of GRP in the context of ADER reported in [39]. ii) The complexity $O(km^3)$, where k is the order of accuracy and $m = 2$, the number of unknowns, tells us that for $k < 5$. The CPU times should remain within similar ranges for all these orders of accuracy, which is observed in the Fig. A.8. Similarly from the complexity analysis, we observe that conventional Cauchy-Kowalewskaya functional has a complexity which grows up exponentially with the order of accuracy, which is also evidenced in Figs. A.7 andA.6.

Appendix B. Operational details

In this appendix, pseudo codes of the recursive formula for obtaining time derivatives are reported. Furthermore, the set up of quadrature points through reference elements and how these are used to obtain polynomials for approximating the spatial and temporal derivatives of high-order, are provided. The numerical flux and the source term evaluation strategy is also reported.

*B1. Pseudo codes for computing the matrix coefficients **D** and **C***

Algorithms 1 and 2 shows the pseudo codes *MATRIX_D* and *MATRIX_C*, respectively which provide the matrix D and C corresponding to the formulas (21) and (29), respectively.

Algorithm 1 MATRIX_D. Here **DxA** and **DxB** contains all the spatial derivatives of **A** and **B**, respectively, at each space time nodal point.

procedure MATRIX_D(l, k, i, j, **A**, **B**, **DxA**, **DxB**)
  if $(l - k - 1 < 0)$ **then**
    DB $\leftarrow$ **0**
  else if $l - k - 1 = 0$ **then**
    DB $\leftarrow$ **B**$(:, :, i, j)$
  else
    DB $\leftarrow$ **DxB**$(:, :, i, j, l - k - 1)$
  if $(l - k = 0)$ **then**
    DA $\leftarrow$ **A**$(:, :, i, j)$
  else
    DA $\leftarrow$ **DxA**$(:, :, i, j, l - k)$
  $comA \leftarrow \frac{(l-1)!}{(k-1)!\cdot(l-k)!}$
  $comB \leftarrow \frac{(l-2)!}{(k-1)!\cdot(l-k-1)!}$
  D $\leftarrow comB *$ **DB** $- comA *$ **DA**
  return D

B2. Approximation of space and time derivatives

The interpolation polynomials obtained here can be straightforward generalized to scalar, vector and matrix functions. So, we provide the polynomials for a generic function $f(\xi)$. To carry out interpolations, we have used the following strategy.

For interpolation in space, we first cast an interval $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ into $[-\frac{1}{2}, \frac{1}{2}]$ by the change of variable $x = x_{i-\frac{1}{2}} + (\xi - \frac{1}{2})\Delta x$. Second, we consider $\xi_j = -\frac{1}{2} + \frac{(j-1)}{M}$ with $j = 1, ..., M+1$. So, the polynomial interpolation in space for $M+1$ order has the form $P(x) = \sum_{k=0}^{M} a_{M+1,k} \cdot \xi^k$. The coefficients $a_{M+1,k}$ are given in the Table B.6 for $M = 1, 2, 3, 4$, we have used the convention $f_j = f(\xi_j)$. Because of the change of variable, each space derivative of order l needs to be scaled by Δx^{-l}.

To obtain interpolation in time, we cast an interval $[t^n, t^{n+1}]$ into the reference element [0, 1] by the change of variable $t = t^n + \tau\,\Delta t$. Then we construct an interpolation polynomial from the Gaussian points, τ_j in [0, 1] with $j = 1, ..., n_{GP}$. Interpolation in time are only needed for accuracy higher than 3. In such a case. The interpolation polynomial has the form $T(\tau) = \sum_{k=0}^{M-1} b_{M+1,k} \cdot \tau^k$. The coefficients $b_{M+1,k}$ are shown in the Table B.7 for $M = 2, 3, 4$. We have used the convention $f_j = f(\tau_j)$. Because of the change of variable, each time derivative of order l needs to be scaled by Δt^{-l}.

Algorithm 2 MATRIX_C. Here **DxA** and **DxB** contains all the spatial derivatives of **A** and **B**, respectively, at each space time nodal point. $NTime = Accuracy - 1$ is the number of temporal nodes, $NSpace = Accuracy$ is the number of spatial nodes. The subroutine uses the function *MATRIX_TIME_GRADIENT* which computes the time derivative approximation of **C** which is carried out in terms of interpolation. *Accuracy* is the order of accuracy of the scheme.

procedure MATRIX_C(l, k, i, j, **A**, **B**, $\mathbf{D_xA}$, $\mathbf{D_xB}$)
  for $j \leftarrow 1, NTime$ **do**
    for $i \leftarrow 1, NSpace$ **do**
      $\mathbf{C}(:,:,i,j,1,1) \leftarrow -\mathbf{A}(:,:,i,j)$
  for $k1 \leftarrow 2, Accuracy - 1$ **do**
    for $j \leftarrow 1, NTime$ **do**
      for $i \leftarrow 1, NSpace$ **do**
        Dmat $\leftarrow MATRIX\_D(k1, k1, i, j,$ **A**, **B**, **DxA**, **DxB**)
        $\mathbf{C}(:,:,i,j,k1,k1) \leftarrow \mathbf{C}(:,:,i,j,k1-1,k1-1) *$ **Dmat**
    for $l1 \leftarrow 1, k1 - 1$ **do**
      DtC$(:,:,:,:,:,k1-1,l1) \leftarrow MATRIX\_TIME\_GRADIENT(\mathbf{C}(:,:,:,:,k1-1,l1))$
      for $j \leftarrow 1, NTime$ **do**
        for $i \leftarrow 1, NSpace$ **do**
          $\mathbf{C}(:,:,i,j,k1,l1) \leftarrow$ **DtCM**$(:,:,i,j,1,k1-1,l1)$
          for $m \leftarrow l1 - 1, k1 - 1$ **do**
            if $(m > 0)$ **then**
              Dmat $\leftarrow MATRIX\_D(m + 1, l1, i, j,$ **A**, **B**, **DxA**, **DxB**)
              $\mathbf{C}(:,:,i,j,k1,l1) \leftarrow \mathbf{C}(:,:,i,j,k1,l1) + \mathbf{C}(:,:,i,j,k1-1,m) *$ **Dmat**
  return C

Table B1
Coefficients $a_{M+1,k}$ for interpolation in space, $P(x) = \sum_{k=0}^{M} a_{M+1,k} \cdot \xi^k$.

Second order ($M = 1$).
$a_{2,0} = f_2$
$a_{2,1} = (f_2 - f_1)$ Third order ($M = 2$).
$a_{3,0} = f_2$
$a_{3,0} = (f_3 - f_1)$
$a_{3,0} = 2(f_3 - 2f_2 + f_1)$
Fourth order ($M = 3$).
$a_{4,0} = -(f_4 - 9f_3 - 9f_2 + f_1)/16$
$a_{4,1} = +((-f_4 + 27f_3 - 27f_2 + f_1))/8$
$a_{4,2} = ((9f_4 - 9f_3 - 9f_2 + 9f_1)\xi^2)/4$
$a_{4,3} = -((-9f_4 + 27f_3 - 27f_2 + 9f_1)\xi^3)/2$
Fifth order ($M = 4$).
$a_{5,0} = f_3$
$a_{5,1} = +((-f_5 + 8f_4 - 8f_2 + f_1))/3$
$a_{5,2} = -((2f_5 - 32f_4 + 60f_3 - 32f_2 + 2f_1))/3$
$a_{5,3} = -((-16f_5 + 32f_4 - 32f_2 + 16f_1))/3$
$a_{5,4} = ((32f_5 - 128f_4 + 192f_3 - 128f_2 + 32f_1))/3$

Table B3
Numerical source term. Here, ω_j are the corresponding Gaussian weights.

Second order ($M = 1$).
$\mathbf{S}_i = \frac{1}{2}\sum_{j=1}^{n_{GP}} \omega_j(\mathbf{S}(\mathbf{Q}_i(\xi_2,\tau_j) + \mathbf{S}(\mathbf{Q}_i(\xi_1,\tau_j))$
Third order ($M = 2$).
$\mathbf{S}_i = \frac{1}{6}\sum_{j=1}^{n_{GP}} \omega_j(\mathbf{S}(\mathbf{Q}_i(\xi_3,\tau_j) + 4\mathbf{S}(\mathbf{Q}_i(\xi_2,\tau_j) + \mathbf{S}(\mathbf{Q}_i(\xi_1,\tau_j))$
Fourth order ($M = 3$).
$\mathbf{S}_i =$
$\frac{1}{8}\sum_{j=1}^{n_{GP}} \omega_j(\mathbf{S}(\mathbf{Q}_i(\xi_4,\tau_j) + 3\mathbf{S}(\mathbf{Q}_i(\xi_3,\tau_j) + 3\mathbf{S}(\mathbf{Q}_i(\xi_2,\tau_j)) + \mathbf{S}(\mathbf{Q}_i(\xi_1,\tau_j))$
Fifth order ($M = 4$).
$\mathbf{S}_i = \frac{1}{90}\sum_{j=1}^{n_{GP}} \omega_j(7\mathbf{S}(\mathbf{Q}_i(\xi_5,\tau_j) + 32\mathbf{S}(\mathbf{Q}_i(\xi_4,\tau_j) + 12\mathbf{S}(\mathbf{Q}_i(\xi_3,\tau_j) +$
$32\mathbf{S}(\mathbf{Q}_i(\xi_2,\tau_j)) + 7\mathbf{S}(\mathbf{Q}_i(\xi_1,\tau_j))$

B3. Evaluation of the numerical flux and source term

Regarding the evaluation of (4). The numerical flux can be easily evaluated as $\mathbf{F}_{i+\frac{1}{2}} = \sum_{j=1}^{n_{GP}} \omega_j \mathbf{F}_h(\mathbf{Q}_i(\xi_{M+1},\tau_j), \mathbf{Q}_{i+1}(\xi_1,\tau_j))$, where ω_j corresponds to the jth Gaussian weight in [0,1]. Whereas, the source terms, can be easily obtained from the interpolation in space and quadrature points in time. Table B.8 shows the form in which the source is computed for several orders of accuracy.

References

[1] Balsara DS. Multidimensional HLLE Riemann solver: application to euler and magnetohydrodynamic flows. J Comput Phys 2010;229(6):1970–93. doi:10.1016/j.jcp.2009.11.018.
[2] Balsara DS. A two-dimensional HLLC Riemann solver for conservation laws: application to euler and magnetohydrodynamic flows. J Comput Phys 2012;231(22):7476–503. doi:10.1016/j.jcp.2011.12.025.
[3] Balsara DS, Dumbser M, Abgrall R. Multidimensional HLLC Riemann solver for unstructured meshes - with application to Euler and MHD flows. J Comput Phys 2014;261:172–208. doi:10.1016/j.jcp.2013.12.029.
[4] Balsara DS, Li J, Montecinos GI. An efficient, second order accurate, universal generalized riemann problem solver based on the hlli riemann solver. J Comput Phys 2018;375:1238–69. doi:10.1016/j.jcp.2018.09.018.
[5] Ben-Artzi M, Falcovitz J. A second order Godunov–type scheme for compressible fluid dynamics. J Comput Phys 1984;55(1):1–32.
[6] Ben-Artzi M, Li J. Hyperbolic balance laws: Riemann invariants and the generalized riemann problem. Numerische Mathematik 2007;106(3):369–425. doi:10.1007/s00211-007-0069-y.
[7] Bourgeade A, Le Floch P, Raviart PA. An asymptotic expansion for the solution of the generalized riemann problem. part 2 : application to the equations of gas dynamics. Annales de l'institut Henri PoincarÃ©(C) Analyse non linÃ©aire 1989;6(6):437–80.
[8] Castro CE, Toro EF. Solvers for the high–order Riemann problem for hyperbolic balance laws. J Comput Phys 2008;227:2481–513.
[9] Dumbser M, Balsara D, Toro EF, Munz CD. A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes. J Comput Phys 2008;227:8209–53.
[10] Dumbser M, Castro MJ, Parés C, Toro EF. ADER schemes on unstructured meshes for nonconservative hyperbolic systems: applications to geophysical flows. Comput Fluid 2009;38(9):1731–48.
[11] Dumbser M, Enaux C, Toro EF. Finite volume schemes of very high order of accuracy for stiff hyperbolic balance laws. J Comput Phys 2008;227(8):3971–4001.
[12] Dumbser M, Käser M. Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems. J Comput Phys 2007;221(2):693–723.

Table B2
Coefficients $b_{M+1,k}$ for interpolation in time, $T(\tau) = \sum_{k=0}^{M-1} b_{M+1,k} \cdot \tau^k$.

Third order ($M = 2$).
$b_{3,0} = (-\sqrt{3}f_2 + f_2 + \frac{(\sqrt{3}+1)f_1)}{2}$
$b_{3,1} = (\sqrt{3}f_2 - \sqrt{3}f_1))$
Fourth order ($M = 3$).
$b_{4,0} = (-\sqrt{15}f_3 + 5f_3 - 4f_2 + (\sqrt{15}+5)f_1)/6$
$b_{4,1} = -\frac{(-\sqrt{15}f_3 + 10f_3 - 20f_2 + (\sqrt{15}+10)f_1)}{3}$
$b_{4,2} = \frac{(10f_3 - 20f_2 + 10f_1)}{3}$
Fifth order ($M = 4$).
$b_{5,0} = -0.1139171962819898f_4 + 0.4007615203116506f_3 - 0.8136324494869276f_2 + 1.526788125457266f_1$
$b_{5,1} = 2.15592710364526f_4 - 7.41707042146264f_3 + 13.80716692568958f_2 - 8.546023607872199f_1$
$b_{5,2} = -7.935761849944949f_4 + 24.99812585921913f_3 - 31.38822236344606f_2 + 14.32585835417188f_1$
$b_{5,3} = 7.420540068038946f_4 \cdot -18.79544940755506f_3 + 18.79544940755506f_2 - 7.420540068038946f_1$

[13] Dumbser M, Munz CD. ADER Discontinuous Galerkin Schemes for Aeroacoustics. Compte Rendus Mécanique 2005;333:683–7.
[14] Dumbser M, Munz CD. Building blocks for arbitrary high order discontinuous Galerkin schemes. J Sci Comput 2006;27:215–30.
[15] Dumbser M, Zanotti O. Very high order PNPM schemes on unstructured meshes for the resistive relativistic MHD equations. J Comput Phys 2009;228(18):6991–7006. doi:10.1016/j.jcp.2009.06.009.
[16] Dumbser M, Balsara DS. A new efficient formulation of the HLLEM Riemann solver for general conservative and non-conservative hyperbolic systems. J Comput Phys 2016;304:275–319. doi:10.1016/j.jcp.2015.10.014.
[17] Goetz CR, Balsara DS, Dumbser M. A family of hll-type solvers for the generalized riemann problem. Comput Fluids 2018;169:201–12. doi:10.1016/j.compfluid.2017.10.028. Recent progress in nonlinear numerical methods for time-dependent flow & transport problems
[18] Goetz CR, Dumbser M. A novel solver for the generalized Riemann problem based on a simplified leFloch–Raviart expansion and a local space–time discontinuous Galerkin formulation. J Sci Comput 2016;69(2):805–40. doi:10.1007/s10915-016-0218-5.
[19] Goetz CR, Iske A. Approximate solutions of generalized Riemann problems: the Toro-Titarev solver and the LeFloch-Raviart expansion. In: Proceeding of Numerical Methods for Hyperbolic Equations: Theory and Applications.Santiago Compostela, Spain, J2011; 2012. p. 267–75. ISBN 9780203562338.
[20] Harten A, Osher S. Uniformly High-Order accurate nonoscillatory schemes. I. SIAM J Numer Anal 1987;24(2):279–309.
[21] Käser M, Dumbser M. An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes – I. The two-dimensional isotropic case with external source terms. Geophys J Int 2006;166(2):855–77. doi:10.1111/j.1365-246X.2006.03051.x.
[22] Le Floch PG, Raviart PA. An asymptotic expansion for the solution of the generalized Riemann problem. Part I : general theory. Annales de l'institut Henri PoincarÃ©(C) Analyse non linÃ©aire 1988;5(2):179–207.
[23] LeVeque RJ, Yee HC. A study of numerical methods for hyperbolic conservation laws with stiff source terms. J Comput Phys 1990;86:187–210.
[24] Men'shov IS. Increasing the order of approximation of Godunov's scheme using solutions of the generalized Riemann problem. USSR Comput Math Math Phys 1990;30(5):54–65. doi:10.1016/0041-5553(90)90161-K.
[25] Men'shov IS. The generalized problem of breakup of an arbitrary discontinuity. J Appl Math Mech 1991;55(1):74–8. doi:10.1016/0021-8928(91)90064-2.
[26] Millington RC, Titarev VA, Toro EF. ADER: Arbitrary-Order Non-Oscillatory Advection Schemes. In: Freisthler H, Warnecke G, editors. Hyperbolic Problems: Theory, Numerics, Applications. ISNM International Series of Numerical Mathematics, 141. Birkhuser Basel; 2001. p. 723–32. ISBN 978-3-0348-9538-5. doi:10.1007/978-3-0348-8372-6_26.
[27] Montecinos G, Castro CE, Dumbser M, Toro EF. Comparison of solvers for the generalized Riemann problem for hyperbolic systems with source terms. J Comput Phys 2012;231:6472–94.
[28] Montecinos GI, Balsara DS. A cell-centered polynomial basis for efficient Galerkin predictors in the context of ader finite volume schemes. the one-dimensional case. Comput Fluids 2017;156:220–38. doi:10.1016/j.compfluid.2017.07.011. Ninth International Conference on Computational Fluid Dynamics (ICCFD9).
[29] Montecinos GI, Toro EF. Reformulations for general advection - diffusion - reaction equations and locally implicit ADER schemes. J Comput Phys 2014;275:415–42.
[30] Schwartzkopff T, Dumbser M, Munz CD. Fast high order ADER schemes for linear hyperbolic equations. J Comput Phys 2004;197:532–9.
[31] Schwartzkopff T, Munz CD, Toro EF. ADER: High–order approach for linear hyperbolic systems in 2D. J Sci Comput 2002;17:231–40.
[32] Shu CW, Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. J Comput Phys 1988;77(2):439–71. doi:10.1016/0021-9991(88)90177-5.
[33] Sod GA. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. J Comput Phys 1978;27(1):1–31. doi:10.1016/0021-9991(78)90023-2.
[34] Takakura Y, Toro EF. Arbitrarily accurate non-oscillatory schemes for nonlinear scalar conservation laws with source terms ii. In: Armfield SW, Morgan P, Srinivas K, editors. Computational Fluid Dynamics 2002. Berlin, Heidelberg: Springer Berlin Heidelberg; 2003. p. 247–52. ISBN 978-3-642-59334-5.
[35] Titarev VA, Toro EF. ADER: arbitrary high order Godunov approach. J Sci Comput 2002;17:609–18.
[36] Toro EF. Primitive, conservative and adaptive schemes for hyperbolic conservation laws. Dordrecht: Springer Netherlands; 1998. p. 323–85. ISBN 978-94-015-9137-9.
[37] Toro EF. Riemann solvers and numerical methods for fluid dynamics: a practical introduction. third. Springer-Verlag; 2009. ISBN 9783642064388. ISBN 978-3-540-25202-3
[38] Toro EF, Millington RC, Nejad LAM. Towards very high–order Godunov schemes. In: Godunov Methods: Theory and Applications. Edited Review, E. F. Toro (Editor). Kluwer Academic/Plenum Publishers; 2001. p. 905–37.
[39] Toro EF, Montecinos GI. Implicit, semi-analytical solution of the generalized Riemann problem for stiff hyperbolic balance laws. J Comput Phys 2015;303:146–72. doi:10.1016/j.jcp.2015.09.039.
[40] Toro EF, Titarev VA. ADER: Towards arbitrary–order non–oscillatory schemes for advection–diffusion–reaction. In: Proc. 8th National Conference on Computational Fluid Dynamics, E-Land, Taiwan, August 18–20, 2001; 2001. p. 8–23.
[41] Toro EF, Titarev VA. Solution of the generalised Riemann problem for advection–reaction equations. Proc R Soc Lond A 2002;458:271–81.
[42] Vanzo D, Siviglia A, Toro EF. Pollutant transport by shallow water equations on unstructured meshes: hyperbolization of the model and numerical solution via a novel flux splitting scheme. J Comput Phys 2016;321:1–20. doi:10.1016/j.jcp.2016.05.023.