Computers and Fluids 156 (2017) 220–238

Contents lists available at ScienceDirect

Computers and Fluids

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

A cell-centered polynomial basis for efficient Galerkin predictors in the context of ADER finite volume schemes. The one-dimensional case

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

[a] Center for Mathematical Modeling, CMM, Universidad de Chile, Santiago, Chile
[b] Department of Physics, University of Notre Dame, USA

ARTICLE INFO

Article history:
Received 29 April 2017
Revised 10 July 2017
Accepted 12 July 2017
Available online 13 July 2017

Keywords:
Finite volume schemes
ADER schemes
Generalized Riemann problems
Discontinuous Galerkin approach
Stiff source terms

ABSTRACT

In this paper, a family of high-order space-time polynomials in the context of continuous and discontinuous Galerkin methods is proposed. The resulting Galerkin schemes are used as a building block in ADER methods for solving one-dimensional hyperbolic balance laws, which can handle stiff source terms. The space-time polynomial basis is constructed as the tensor product of spatial and temporal polynomials. Temporal polynomials are constructed as conventional Lagrange polynomials on a set of temporal nodes, which is formed by Gaussian quadrature points of suitable order. To build the spatial polynomials we propose a set of spatial nodes, the number of these nodes are about a half of those required by conventional Lagrange polynomials.

Then, polynomials and their first derivatives are imposed to be nodal on the set of spatial nodes, it generates two family of degrees of freedom associated with polynomials and their derivatives. The procedure generates even numbers of polynomials. The degrees of freedom of the space-time polynomial solution, resulting from Galerkin approaches are obtained from a system of algebraic equations, which are coupled only in the flux and gradients of fluxes. It allows us to construct an efficient nested-type iteration procedure involving only the source terms and the gradients of source terms, where the set of degrees of freedom are decoupled. Only a few number of iterations are required to get the expected accuracy. Several test cases are solved to evidence the ability of the present scheme for solving hyperbolic balance laws. Expected theoretical orders of accuracy are obtained up to the fourth order in both space and time, using generous CFL numbers.

1. Introduction

Arbitray accuracy **DER**ivative Riemann problem (ADER) methods were first time put forward by Toro and collaborators [21,27–29]. These methods are based on two main steps. First, a reconstruction procedure, here a high order reconstruction is required, the Essentially Non-Oscillatory ENO procedure introduced by Harten et al. [16] and the Weighted Essentially Non-Oscillatory WENO approach, presented by Osher and Shu [26] can be chosen. Of particular interest for us, is the WENO strategy reported in [8,10,12]. The reconstruction process provides a piecewise polynomial representation of the solution at a given time level. Second, reconstruction polynomials are used to construct local initial value problems. The flux and source terms are computed from the solution of these local initial value problems. So, for the flux evaluation, the solution of a Generalized Riemann Problem (GRP) is used. The GRP is a local initial value problem defined at a cell interface, where the Partial Differential Equation (PDE) is the governing equation and the initial condition is given by the reconstructed polynomials at each side of the cell interface, so it is a piece-wise polynomial function. On the other hand, the solution of a local initial value problem within a cell, formed by the governing equation and the reconstructed polynomial, is used for the source evaluation. The same strategy implemented for solving GRP can be used to get the solution of this initial value problem.

For solving GRP's we may identify two strategies, well described by Castro and Toro [5]; the Toro-Titarev (TT) solver, which was the original strategy proposed in [29] and the other one is that following the idea proposed by Harten et al. [17], reformulated for high order implementations in [5] and redefined by the authors as HEOC solver. In the original work of Toro and Titarev [21,27,29], the GRP solutions are proposed in terms of explicit Taylor series expansions in time around the interface position, where the high-order time derivatives are obtained by using the Cauchy–Kowalewsky procedure [18], also referred as the Lax–Wendroff procedure [19], in this procedure the governing equation

* Corresponding author.
E-mail addresses: gmontecinos@dim.uchile.cl, gmont000@gmail.com (G.I. Montecinos).

http://dx.doi.org/10.1016/j.compfluid.2017.07.011
0045-7930/

is systematically used to express time derivatives in terms of purely space derivatives. Therefore for each time derivative, the Cauchy–Kowalewsky procedure generates a functional which requires spatial derivatives. The leading term of the Taylor expansion is obtained as the solution at the interface position of a local classical Riemann problems and the high derivatives required for the Cauchy–Kowalewsky procedure, are obtained from the solution of linear classical Riemann problems, which are obtained from the linearization of the governing equation around the computed leading term. On the other hand, in the HEOC strategy, the GRP's are solved in two stages; (i) compute independently predictors at both sides of the interface (ii) predictors at prescribed local times aside the interface positions, interact via the solution of a classical Rieman problem. The HEOC solver as proposed in [5], also provides the GRP solution via Taylor series expansions in time, but they are used to obtain extrapolated values of the solution at both sides of the interface position. So, the Cauchy–Kowalewsky procedure is also required but now the functionals are not filled with spatial derivatives coming from linearized Riemann problems, in this approach derivatives are obtained from the derivatives of reconstructed polynomials. For general non linear hyperbolic systems, schemes based on the Taylor expansion and Cauchy–Kowalewsky procedure can be cumbersome. Furthermore, if source terms are present, then the explicit Taylor expansions may fail to provide accurate solutions. A successful alternative was proposed by Dumbser et al. [8] and Dumbser et al. [6], where a local space-time Galerkin scheme was used to obtain a predictor inside a cell, see [22] for a comparison of GRP solver based on Cauchy–Kowalewsky procedures and the discontinuous Galerking approach. This kind of unified finite volume methods and Galerkin methods have shown to be a successful alternative for solving non-linear hyperbolic problems, see the schemes in [4,7,9,11,13–15,24] to mention but a few.

In the Galerkin framework, the PDE is solved in a weak integral form, from which a space-time polynomial approximation of the solution is obtained for each cell. The Galerkin framework generates algebraic non-linear systems for the degrees of freedoms, which normally are solved by using fixed point iteration procedures. Discontinuous as well as continuous Galerkin formulations can been proposed. It depends on the form the hyperbolic system, the presence of the source terms may introduce jumps between the solution at local time $\tau = 0$ and the solution at $\tau = 0^+$ so the discontinuous approach is able to represent this situation by introducing the reconstructed polynomial as the initial condition but in a weak sense. On the other hand, if the jump between solutions at $\tau = 0$ and $\tau = 0^+$ is negligible, then the initial condition can be included in a strong form, which is, the degrees of freedom corresponding to the local time level $\tau = 0$ are those of the reconstructed polynomial. Discontinuous Galerkin schemes are able to solve stiff source terms but they are more expensive than the continuous approach, because all the degrees of freedom need to be obtained.

Although these methods are locally implicit, the fixed point iteration procedures and matrix inversion carried out at each computational cell, make these schemes to be expensive. In this sense Balsara et al. [3] have recently proposed a scheme, which may improve these issues. In [3], a second order strategy for solving the Maxwell equation has been proposed, the method is designed to work in three dimensions by using the ADER-DG approach, in this work the authors have chosen a family of space-time polynomial, which is the tensor product of temporal and spatial polynomials. To build spatial polynomials the authors have proposed the following strategy; by imposing the condition that at the node the polynomial is nodal, that means it takes the value one at the node, and its three gradients are zero, a polynomial of degree one, is obtained. Similarly, by imposing that polynomials are zero at the node and their three gradients are nodal for each one of components, then three additional polynomials of degree one, are obtained. This procedure generates a set of four space polynomials in three dimensions. To build temporal polynomials, the Lagrange interpolation is carried out on temporal nodes $\tau_1 = 1/2$ and $\tau_2 = 1$. Notice that $\tau_1 = 1/2$ corresponds to the Gaussian quadrature point for the quadrature rule of second order. Although all the degrees of freedom need to be computed, only those associated with $\tau_1 = 1/2$ are involved for flux and source evaluations. Degrees of freedom associated with this choice of space-time polynomials are obtained by solving a system of algebraic equations, the Jacobian of the system depicts a blockwise structure which is used to design a very efficient inversion procedure.

The present work is devoted to the development of a family of high order space-time polynomials, aimed to be implemented in both Continuous Galerkin (CG) and Discontinuous Galerkin (DG) approaches, used as the predictor step in the context of ADER methods. The aim is to build very efficient schemes able to solve one-dimensional hyperbolic balance laws with stiff source terms. In this work, we extend the strategy in [3] to build high order space-time polynomials as tensor product of spatial and temporal polynomials.

Here, the spatial polynomials are obtained from the nodal imposition on polynomials as well as on their first derivatives, it is in some sense similar to the conventional Lagrange interpolation on a set of nodes. In conventional Lagrange interpolation, we need the same nodes as the accuracy of the scheme, then the degrees of freedom of Lagrange polynomials are set by imposing polynomials to be nodal. In our approach, the number of nodes are about a half of those required by the conventional Lagrange polynomial, we choose these nodes to be symmetric with respect to the barycentre of a local reference element. Then, we not only impose polynomials to be nodal but also the first derivative of polynomials. So the number of equations is twice the number of nodes, so the procedure is restricted to generate polynomials of even orders. Although for odd accuracy, this procedure generates polynomials having more degrees of freedom than those for convention Lagrange interpolation, the number of nodes is less than those required for conventional Lagrange interpolation and so the computational cost is compensated.

On the other hand, temporal polynomials are built in a Lagrange sense, on a set of suitable nodes. So, for constructing a numerical scheme of order N, we obtain a Lagrange polynomial of order $N-1$, by using $N-1$ Gaussian quadrature points in the interval [0, 1] plus the node 0 in the case of continuous Galerkin formulation or we use $N-1$ Gaussian quadrature points plus the node 1 in the case of discontinuous Galerkin scheme. The use of the proposed set of space-time polynomials in the context of Galerkin schemes yields a relationship among the degrees of freedom, flux and source terms, which is able to be separated into two sets of coefficients which are weakly coupled, they are only coupled in the flux and derivatives of fluxes. This is suitable for a nested-type fixed-point iteration procedures. Furthermore, the structure of the operators resulting from the Galerkin formulations, allows us to build a very efficient inversion procedure to obtain the sought degrees of freedom, this is suggested for solving hyperbolic balance laws with very stiff source terms. This makes a substantial difference between the proposed set of polynomials and the conventional Lagrange polynomials. For both stiff and non-stiff source term, the iteration procedures only require a few number of iterations to get the accuracy.

Several test cases are solved to evidence the ability of the present scheme for solving hyperbolic balance laws. Expected theoretical orders of accuracy are obtained up to the fourth order in both space and time, using generous CFL numbers.

This paper is organized as follows, in Section 2 the ADER type method implemented in this work is shown. In Section 3 the Galerkin schemes using the present set of polynomials are discussed, the construction of spatial and temporal polynomials is detailed. In Section 4 numerical results are presented and conclusions are contained in Section 5.

2. The ADER method

In this section we describe the ADER scheme, for solving conservative hyperbolic balance laws in the form

$$\partial_t \mathbf{Q}(x,t) + \partial_x \mathbf{F}(\mathbf{Q}(x,t)) = \mathbf{S}(\mathbf{Q}(x,t)), \qquad \mathbf{Q}(x,0) = \mathbf{H}(x), \tag{1}$$

where $\mathbf{Q}(x,t) \in \mathbb{R}^m$ is the vector of states, $\mathbf{H}(x)$ is the initial condition, $\mathbf{F}(\mathbf{Q}(x,t))$ is the flux function of the state and $\mathbf{S}(\mathbf{Q}(x,t))$ is the source term. Here, we use the conventional one-step finite volume 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 we use

$$\mathbf{Q}_i^0 = \frac{1}{\Delta x}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{H}(x)dx,$$

$$\mathbf{F}_{i+\frac{1}{2}} = \frac{1}{\Delta t}\int_0^{\Delta t} \mathbf{F}_h(\mathbf{Q}_i(x_{i+\frac{1}{2}},\tau), \mathbf{Q}_{i+1}(x_{i+\frac{1}{2}},\tau))d\tau,$$

$$\mathbf{S}_i = \frac{1}{\Delta t \Delta x}\int_0^{\Delta t}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{S}(\mathbf{Q}_i(x,\tau))dx d\tau. \tag{3}$$

Here $\mathbf{F}_h(\mathbf{Q}_L, \mathbf{Q}_R)$ represents some conventional numerical flux function (classical Riemann solver), which depends only on two states $\mathbf{Q}_L$ and $\mathbf{Q}_R$, that means, at a given τ we use the solution at the local interface $x = 0$ of the problem

$$\partial_t \mathbf{Q}(x,t) + \partial_x \mathbf{F}(\mathbf{Q}(x,t)) = \mathbf{S}(\mathbf{Q}(x,t)), \qquad \mathbf{Q}(x,0) = \begin{cases} \mathbf{Q}_L \equiv \mathbf{Q}_i(x_{i+\frac{1}{2}},\tau), x < 0, \\ \mathbf{Q}_R \equiv \mathbf{Q}_{i+1}(x_{i+\frac{1}{2}},\tau), x > 0, \end{cases} \tag{4}$$

where $\mathbf{Q}_i(x,t)$ is the local predictor inside the cell $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [t^n, t^{n+1}]$, with $\Delta x = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ and $\Delta t = t^{n+1} - t^n$.

This GRP solver follows the strategy in [5,22], named the HEOC solver, which is a high-order reformulation of the approach of Harten. et al. [17]. However, in these papers, predictors $\mathbf{Q}_i(x,t)$ are obtained from the use of Taylor series expansion and the Cauchy–Kowalewsky procedure. The present approach avoids the use of Taylor expansions and the Cauchy–Kowalewsky procedure by using a space-time local weak formulation of the governing equation. Therefore, this approach follows the HEOC approach but the predictor step is closed to the strategy reported in [1–3,6,8,14], to mention but a few.

In the following section, the proposed strategy for the predictor step, is presented.

3. Galerking method for the predictor step, continuous as well as discontinuous approach

In this section, we describe the way in which a polynomial representation of the solution, within the computational cell $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [t^n, t^{n+1}]$, is obtained. The first step is to transform the problem into the local reference system $\xi, \tau \in [0, 1]$ via the change of variables

$$x = x(\xi) = x_{i-\frac{1}{2}} + \xi \Delta x, \quad t = t(\tau) = t^n + \tau \Delta t, \tag{5}$$

thus, the PDE becomes

$$\partial_\tau \tilde{\mathbf{Q}}(\xi,\tau) + \partial_\xi \mathbf{F}^*(\tilde{\mathbf{Q}}(\xi,\tau)) = \mathbf{S}^*(\tilde{\mathbf{Q}}(\xi,\tau)), \qquad \tilde{\mathbf{Q}}(\xi,0) = \mathbf{H}(x(\xi)), \tag{6}$$

where

$$\tilde{\mathbf{Q}}(\xi,\tau) := \mathbf{Q}(x(\xi), t(\tau)),$$
$$\mathbf{F}^*(\tilde{\mathbf{Q}}(\xi,\tau)) := \frac{\Delta t}{\Delta x}\mathbf{F}(\mathbf{Q}(x(\xi), t(\tau))),$$
$$\mathbf{S}^*(\tilde{\mathbf{Q}}(\xi,\tau)) := \Delta t \mathbf{S}(\mathbf{Q}(x(\xi), t(\tau))). \tag{7}$$

We solve system (6) in a polynomial space $V := \{\phi_k(\xi, \tau)\}$, that means we look for a solution $\tilde{\mathbf{Q}}(\xi,\eta) := \sum_k \hat{Q}_k \phi_k(\xi,\tau)$, satisfying the variational problem

$$< \partial_\tau \tilde{\mathbf{Q}}, \phi_k > + < \partial_\xi \mathbf{F}^*(\tilde{\mathbf{Q}}), \phi_k > = < \mathbf{S}^*(\tilde{\mathbf{Q}}), \phi_k >, \tag{8}$$

where we have introduced

$$< f, g > := \int_0^1 \int_{-\frac{1}{2}}^{\frac{1}{2}} f(\xi,\tau) g(\xi,\tau) d\xi d\tau,$$
$$[f, g]_\tau := \int_{-\frac{1}{2}}^{\frac{1}{2}} f(\xi,\tau) g(\xi,\tau) d\xi. \tag{9}$$

Integrating by parts, we obtain

$$< \partial_\tau \tilde{\mathbf{Q}}, \phi_k > = [\tilde{\mathbf{Q}}, \phi_k]_1 - [\tilde{\mathbf{Q}}, \phi_k]_0 - < \tilde{\mathbf{Q}}, \partial_t \phi_k > . \tag{10}$$

After this stage, we can choose between the Discontinuous and Continuous Galerkin approach. The choice is based on the fact that the source term generates a variation between solution at local initial $\tau = 0$ and the solution at $\tau = 0^+$. If this variation is large then a suitable approach is the Discontinuous Galerkin formulation, where the initial condition is incorporated in a weak sense, we replace $[\tilde{\mathbf{Q}}, \phi_k]_0$ by

$$[\tilde{\mathbf{Q}}, \partial_t \phi_k]_1 - < \tilde{\mathbf{Q}}, \partial_t \phi_k > + < \partial_\xi \mathbf{F}^*(\tilde{\mathbf{Q}}), \phi_k > = < \mathbf{S}^*(\tilde{\mathbf{Q}}), \phi_k > + \{\tilde{\mathbf{W}}, \phi_k\}, \tag{11}$$

with

$$\{\tilde{\mathbf{W}}, \phi_k\} = \int_{-\frac{1}{2}}^{\frac{1}{2}} \tilde{\mathbf{W}}(\xi), \phi_k(\xi, 0) d\xi, \tag{12}$$

where $\tilde{\mathbf{W}}(\xi)$, is the reconstruction polynomial. Notice that in this case $\tilde{\mathbf{W}}(\xi) \neq \tilde{\mathbf{Q}}(\xi, 0)$. In Sections 3.2 and 3.3, solution strategies for solving (11) are detailed.

On the other hand, if the variation does not matter, then a continuous Galerkin scheme is implemented, here the initial condition is incorporated in a strong fashion, that means at local level $\tau = 0$ the solution polynomial is exactly the reconstructed polynomial, $\tilde{\mathbf{W}}(\xi) = \tilde{\mathbf{Q}}(\xi, 0)$ then, (8) becomes

$$[\tilde{\mathbf{Q}}, \partial_t \phi_k]_1 - < \tilde{\mathbf{Q}}, \partial_t \phi_k > + < \partial_\xi \mathbf{F}^*(\tilde{\mathbf{Q}}), \phi_k > = < \mathbf{S}^*(\tilde{\mathbf{Q}}), \phi_k > + [\tilde{\mathbf{Q}}, \phi_k]_0. \tag{13}$$

In Section 3.4 the solution strategy for solving (13) is detailed. Notice that in the continuous approach some degrees of freedom have already been set from the reconstructed polynomials, thus the continuous approach is more efficient than the discontinuous approach.

3.1. A class of cell-centered polynomial basis

This section contains one of the most relevant part of this paper. An early second order version of this innovative choice of polynomial basis functions has been described in [3] for the numerical solution of Maxwell equations in material media. However, for the sake of completeness, here let us describe the strategy for the one-dimensional case. Let us define n_N nodes

$\{\xi_1, \xi_2, \ldots, \xi_{n_N}\} \subset [-\frac{1}{2}, \frac{1}{2}]$. We are going to construct a set of $2n_N =: n_{DOF}$ polynomials divided into two groups $\{P_i(\xi)\}_{i=1}^{n_N}$ and $\{P_i(\xi)\}_{i=n_N+1}^{2n_N}$, where each one of these polynomials has the form

$$P_i(\xi) = \sum_{j=1}^{n_{DOF}} a_j \xi^j. \tag{14}$$

These polynomials, satisfy

$$P_i(\xi_j) = \delta_{i,j},\ P_{i+n_N}(\xi_j) = 0, i, j = 1, \ldots, n_N, \tag{15}$$

that means, the polynomials of the first group $\{P_i(\xi)\}_{i=1}^{n_N}$, are nodal on $\{\xi_1, \xi_2, \ldots, \xi_{n_N}\}$ and

$$P_i'(\xi_j) = 0,\ P_{i+n_N}'(\xi_j) = \delta_{i,j},\ i, j = 1, \ldots, n_N, \tag{16}$$

that means, the first derivative of polynomials of the second group $\{P_i(\xi)\}_{i=n_N+1}^{2n_N}$, are nodal on $\{\xi_1, \xi_2, \ldots, \xi_{n_N}\}$. Notice that from (15) to (16) we obtain the n_{DOF} degrees of freedom a_j for each one of these polynomials.

In order to define the space-time polynomial, V, spanned by space-time polynomials of the form

$$\phi_k(\xi, \tau) = P_i(\xi) T_j(\tau), \tag{17}$$

we need the temporal polynomials, $T_j(\tau)$, which are Lagrange polynomials in time, so for a given order of accuracy N, they are constructed using a set of $n_{GP} = N - 1$ quadrature points $\{\chi_1, \ldots, \chi_{n_{GP}}\}$ plus an additional point which will depend on the type of Galerkin approach, continuous or discontinuous.

3.2. Discontinuous Galerking approach

To build a local space-time discontinuous Galerkin scheme for (11), we set the following nodes in time

$$\begin{aligned} \tau_1 &= \chi_1, \\ \tau_2 &= \chi_2, \\ &\vdots \\ \tau_{n_{GP}} &= \chi_{n_{GP}}, \\ \tau_{n_{GP}+1} &= 1. \end{aligned} \tag{18}$$

Then, we build a set of Lagrange polynomials

$$T_k(\tau) = \Pi_{l=1, l\neq k}^{n_{GP}+1} \frac{(\tau - \tau_l)}{(\tau_k - \tau_l)}. \tag{19}$$

We define the space-time basis spanned by polynomials

$$\phi_{r(i,j)}(\xi, \tau) = P_i(\xi) T_j(\tau), \tag{20}$$

where r is a mono-index defined by $r = r(i, j) = i + (j - 1)n_{DOF}$ with $i = 1, \ldots, n_{DOF}$ and $j = 1, \ldots, n_{GP} + 1$, so $r = 1, \ldots, n_{DOF}(n_{GP} + 1)$.

As polynomials $P_i(\xi)$ satisfy the constraints (15) and (16), coefficients $\hat{Q}_r$ are divided into two sets of coefficients denoted by $\{\hat{\mathbf{V}}^0_{\alpha(i,j)}\}$ and $\{\hat{\mathbf{V}}^x_{\alpha(i,j)}\}$, associated with polynomials satisfying (15) and (16), respectively. Here $\alpha(i, j) = i + (j - 1)n_N$ is a mono-index function, with $i = 1, \ldots, n_N$ and $j = 1, \ldots, n_{GP} + 1$. So

$$\tilde{\mathbf{Q}}(\xi, \tau) = \sum_{j=1}^{n_{GP}+1} \left\{ \sum_{i=1}^{n_N} \hat{\mathbf{V}}^0_{\alpha(i,j)} \phi_{r(i,j)}(\xi, \tau) + \sum_{i=1}^{n_N} \hat{\mathbf{V}}^x_{\alpha(i,j)} \phi_{r(i+n_N,j)}(\xi, \tau) \right\}. \tag{21}$$

Notice that, we can project the flux and the source term into V as

$$\begin{aligned} \tilde{\mathbf{F}}^*(\xi, \tau) &= \sum_{j=1}^{n_{GP}+1} \left\{ \sum_{i=1}^{n_N} \hat{\mathbf{F}}^0_{\alpha(i,j)} \phi_{r(i,j)}(\xi, \tau) + \sum_{i=1}^{n_N} \hat{\mathbf{F}}^x_{\alpha(i,j)} \phi_{r(i+n_N,j)}(\xi, \tau) \right\}, \\ \tilde{\mathbf{S}}^*(\xi, \tau) &= \sum_{j=1}^{n_{GP}+1} \left\{ \sum_{i=1}^{n_N} \hat{\mathbf{S}}^0_{\alpha(i,j)} \phi_{r(i,j)}(\xi, \tau) + \sum_{i=1}^{n_N} \hat{\mathbf{S}}^x_{\alpha(i,j)} \phi_{r(i+n_N,j)}(\xi, \tau) \right\}, \end{aligned} \tag{22}$$

where coefficients $\hat{\mathbf{F}}^0_{\alpha(i,j)}$, $\hat{\mathbf{F}}^x_{\alpha(i,j)}$, $\hat{\mathbf{S}}^0_{\alpha(i,j)}$ and $\hat{\mathbf{S}}^x_{\alpha(i,j)}$ are determined as follows. Equating

$$\begin{aligned} \mathbf{F}^*(\tilde{\mathbf{Q}}(\xi, \tau)) &= \tilde{\mathbf{F}}^*(\xi, \tau), \\ \mathbf{S}^*(\tilde{\mathbf{Q}}(\xi, \tau)) &= \tilde{\mathbf{S}}^*(\xi, \tau), \end{aligned} \tag{23}$$

and evaluating at nodes $\chi_{\alpha(i,j)} = (\xi_i, \tau_j)$, from (15) we obtain

$$\begin{aligned} \hat{\mathbf{F}}^0_{\alpha(i,j)} &:= \mathbf{F}^*(\hat{\mathbf{V}}^0_{\alpha(i,j)}), \\ \hat{\mathbf{S}}^0_{\alpha(i,j)} &:= \mathbf{S}^*(\hat{\mathbf{V}}^0_{\alpha(i,j)}), \end{aligned} \tag{24}$$

similarly, by differentiating (22) we obtain

$$\begin{aligned} \mathbf{A}^*(\tilde{\mathbf{Q}}(\xi, \tau)) \partial_\xi \tilde{\mathbf{Q}}(\xi, \tau) &= \partial_\xi \tilde{\mathbf{F}}^*(\xi, \tau), \\ \mathbf{B}^*(\tilde{\mathbf{Q}}(\xi, \tau)) \partial_\xi \tilde{\mathbf{Q}}(\xi, \tau) &= \partial_\xi \tilde{\mathbf{S}}^*(\xi, \tau), \end{aligned} \tag{25}$$

where $\mathbf{A}^*(\tilde{\mathbf{Q}})$ and $\mathbf{B}^*(\tilde{\mathbf{Q}})$ are the Jacobian matrices of $\mathbf{F}^*(\tilde{\mathbf{Q}})$ and $\mathbf{S}^*(\tilde{\mathbf{Q}})$, respectively. So by evaluating at nodes $\chi_{\alpha(i,j)} = (\xi_i, \tau_j)$, from (16) we have

$$\begin{aligned} \hat{\mathbf{F}}^x_{\alpha(i,j)} &:= \mathbf{A}^*(\hat{\mathbf{V}}^0_{\alpha(i,j)}) \hat{\mathbf{V}}^x_{\alpha(i,j)}, \\ \hat{\mathbf{S}}^x_{\alpha(i,j)} &:= \mathbf{B}^*(\hat{\mathbf{V}}^0_{\alpha(i,j)}) \hat{\mathbf{V}}^x_{\alpha(i,j)}. \end{aligned} \tag{26}$$

The projection of the flux into V, in terms of coefficients $\hat{\mathbf{F}}^0_{\alpha(i,j)}$ $\hat{\mathbf{F}}^x_{\alpha(i,j)}$ is not as simple as in the case of conventional nodal space-time polynomials, but it only involves the degrees of freedoms associated with the space-time node $\chi_{\alpha(i,j)} = (\xi_i, \tau_j)$, that is $\hat{\mathbf{V}}^0_{\alpha(i,j)}$ and $\hat{\mathbf{V}}^x_{\alpha(i,j)}$. This observation also applies for the projection of the source term into V.

From the reconstruction procedure we have a polynomial $\mathbf{W}(x)$ defined within the cell $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$. However, we need to incorporate the information from reconstruction procedure in terms of the polynomial $\{P_i(\xi)\}_{i=1}^{n_{DOF}}$ so look for a polynomial of the form

$$\tilde{\mathbf{W}}(\xi) = \sum_{i=1}^{n_N} \hat{\mathbf{W}}^0_i P_i(\xi) + \sum_{i=1}^{n_N} \hat{\mathbf{W}}^x_i P_{i+n_N}(\xi), \tag{27}$$

coefficients $\hat{\mathbf{W}}^0_i$ and $\hat{\mathbf{W}}^x_i$ are found by equating the reconstruction polynomial $\mathbf{W}(x)$ and evaluate at ξ_i, so from (15) we have

$$\hat{\mathbf{W}}^0_i = \mathbf{W}(x(\xi_i)). \tag{28}$$

and if we differentiate $\mathbf{W}(x(\xi))$ and equate to $\tilde{\mathbf{W}}(\xi)$ from (16) we obtain

$$\hat{\mathbf{W}}^x_i = \partial_\xi \tilde{\mathbf{W}}(\xi_i) = \frac{d\mathbf{W}(x(\xi_i))}{d\xi} \frac{1}{\Delta x}. \tag{29}$$

Now, we can set (11) in a matrix form

$$\mathbf{K1}\mathcal{Q} + \mathbf{K}^\xi \mathcal{F} = \mathbf{M}\mathcal{S} + \mathbf{M}\mathbf{w}^0 \mathcal{W}, \tag{30}$$

where

$$\begin{aligned} \mathbf{K1}_{k,l} &= [\phi_k, \phi_l]_1 - < \phi_k, \partial_\tau \phi_l >, \\ \mathbf{K}^\xi_{k,l} &= < \partial_\xi \phi_k, \phi_l >, \\ \mathbf{M}_{k,l} &= < \phi_k, \phi_l > \end{aligned} \tag{31}$$

and

$$\mathcal{Q}_{r(i,j)} = \hat{\mathbf{V}}^0_{\alpha(i,j)},$$

$$\begin{aligned}\mathcal{Q}_{r(i+n_N,j)} &= \hat{\mathbf{V}}^x_{\alpha(i,j)},\\ \mathcal{W}_i &= \hat{\mathbf{W}}^0_i,\\ \mathcal{W}_{i+n_N} &= \hat{\mathbf{W}}^x_i,\end{aligned} \tag{32}$$

flux and source coefficients become

$$\begin{aligned}\mathcal{F}_{r(i,j)} &= \hat{\mathbf{F}}^0_{\alpha(i,j)},\\ \mathcal{F}_{r(i+n_N,j)} &= \hat{\mathbf{F}}^x_{\alpha(i,j)},\\ \mathcal{S}_{r(i,j)} &= \hat{\mathbf{S}}^0_{\alpha(i,j)},\\ \mathcal{S}_{r(i+n_N,j)} &= \hat{\mathbf{S}}^x_{\alpha(i,j)},\end{aligned} \tag{33}$$

for $i = 1\ldots, n_N$ and $j = 1, \ldots, n_{GP}+1$. Information from the reconstruction is given by

$$\mathbf{Mw}^0_{k,l} = \int_{-\frac{1}{2}}^{\frac{1}{2}} \phi_k(\xi, 0) P_l(\xi) d\xi.$$

Notice that by mean of (68) we can get coefficients $\hat{\mathbf{V}}^0_{\alpha(i,j)}$ and $\hat{\mathbf{V}}^x_{\alpha(i,j)}$ from $\mathcal{Q}$. The solution can be solved with the fixed-point procedure

$$\mathbf{K1}\,\mathcal{Q}^{p+1} + \mathbf{K}^\xi \mathcal{F}^p = \mathbf{M}\mathcal{S}^p + \mathbf{Mw}^0 \mathcal{W}, \tag{34}$$

where p stands by the iteration index. If so, then the procedure is very simple, at each iteration step it only requires the right multiplication of the last equation by the inverse matrix $(\mathbf{K1})^{-1}$. This local predictor has a simple structure, which allows us to split the problem into decoupled systems, suitable for iterative procedures.

Proposition 1. *For each order of accuracy N, there exist constant values $a_{j,k}$ such that*

$$\begin{aligned}\hat{\mathbf{V}}^0_{i+(j-1)n_N} &= \hat{\mathbf{W}}^0_i + \sum_{k=1}^{N} a_{j,k} \hat{\mathbf{S}}^0_{i+(k-1)n_N} + \mathbf{RF}^0_{i+(j-1)n_N}(\mathbf{F}^0, \mathbf{F}^x),\\ \hat{\mathbf{V}}^x_{i+(j-1)n_N} &= \hat{\mathbf{W}}^x_i + \sum_{k=1}^{N} a_{j,k} \hat{\mathbf{S}}^x_{i+(k-1)n_N} + \mathbf{RF}^x_{i+(j-1)n_N}(\mathbf{F}^0, \mathbf{F}^x),\end{aligned} \tag{35}$$

where $\mathbf{RF}^0$ and $\mathbf{RF}^x$ are functionals which only depends on the flux, with $i = 1, \ldots, n_N,\ j = 1, \ldots, N$.

Proof. Due to the form of the polynomial basis, we can write every matrix in (34) as

$$\begin{aligned}\mathbf{M} &= \mathbf{M}^\tau \otimes \mathbf{M}^\xi,\\ \mathbf{Mw}^0 &= \mathbf{R}^{\tau,0} \otimes \mathbf{M}^\xi,\\ \mathbf{K1} &= (\mathbf{R}^{\tau,1} - \mathbf{K}^{\tau,1}) \otimes \mathbf{M}^\xi,\\ \mathbf{K}^\xi &= \mathbf{M}^\tau \otimes \mathbf{M}^{\xi,1},\end{aligned} \tag{36}$$

where product $\otimes$ for two arbitrary matrices $\mathbf{A}$ and $\mathbf{B}$ of size $m \times n$ and $p \times q$, respectively, is defined as $(\mathbf{A} \otimes \mathbf{B})_{k+(i-1)p,l+(j-1)q} = \mathbf{A}_{i,j}\mathbf{B}_{k,l}$. Local matrices $\mathbf{M}_\tau$, $\mathbf{Mw}_\tau$, $(\mathbf{R}_1 - \mathbf{K}^{\tau,1})$, $\mathbf{M}^\tau$, $\mathbf{M}^\xi$ and $\mathbf{M}^{\xi,1}$ are defined as

$$\begin{aligned}\mathbf{M}^\tau_{i,j} &= \int_0^1 T_i(\tau) T_j(\tau) d\tau,\\ \mathbf{R}^{\tau,1}_{i,j} &= T_i(1) T_j(1),\\ \mathbf{R}^{\tau,0}_{i} &= T_i(0),\\ \mathbf{K}^{\tau,1}_{i,j} &= \int_0^1 T_i'(\tau) T_j(\tau) d\tau,\\ \mathbf{M}^\xi_{i,j} &= \int_{-\frac{1}{2}}^{\frac{1}{2}} P_i(\xi) P_j(\xi) d\xi,\\ \mathbf{M}^{\xi,1}_{i,j} &= \int_{-\frac{1}{2}}^{\frac{1}{2}} P_i(\xi) P_j'(\xi) d\xi.\end{aligned} \tag{37}$$

Additionally, let us define $\mathbf{E} = \mathbf{R}^{\tau,1} - \mathbf{K}^{\tau,1}$. So, from

$$\begin{aligned}\mathcal{Q} = {} & (\mathbf{E}^{-1}\mathbf{R}^{\tau,0}) \otimes ((\mathbf{M}^\xi)^{-1}\mathbf{M}^\xi)\mathcal{W} + (\mathbf{E}^{-1}\mathbf{M}_\tau) \otimes (\mathbf{M}^\xi)^{-1}\mathbf{M}^\xi \mathcal{S}\\ & - (\mathbf{E}^{-1}\mathbf{M}_\tau) \otimes (\mathbf{M}^\xi)^{-1}\mathbf{M}^{\xi,1}\mathcal{F},\end{aligned} \tag{38}$$

or

$$\mathcal{Q} = (\mathbf{E}^{-1}\mathbf{R}^{\tau,0}) \otimes \mathbf{I}\mathcal{W} + (\mathbf{E}^{-1}\mathbf{M}_\tau) \otimes \mathbf{I}\mathcal{S} - (\mathbf{E}^{-1}\mathbf{M}_\tau) \otimes (\mathbf{M}^\xi)^{-1}\mathbf{M}^{\xi,1}\mathcal{F}, \tag{39}$$

where $\mathbf{I}$ is the $2n_N \times 2n_N$ identity matrix. Remember that, firsts n_{GP} nodes in (18) correspond to Gaussian quadrature points, with weights $\{\omega_k\}_{k=1}^{n_{GP}}$, so we can include an extra point with the corresponding weight $\omega_{n_{GP}+1} = 0$. Therefore, we can integrate exactly polynomials up to degree $2N-3$. So the first term in (39)

$$\mathbf{E}_{i,j} = T_i(1)T_j(1) - \int_0^1 T_i'(\tau) T_j(\tau) d\tau, \tag{40}$$

is exactly

$$\mathbf{E}_{i,j} = T_i(1)T_j(1) - \sum_{k=1}^{N=n_{GP}+1} T_i'(\tau_k) T_j(\tau_k) \omega_k, \tag{41}$$

on the other hand

$$\begin{aligned}\sum_j^N \mathbf{E}_{i,j} &= \sum_j^N (T_i(1)T_j(1) - \sum_{k=1}^{N=n_{GP}+1} T_i'(\tau_k) T_j(\tau_k)\omega_k)\\ &= T_i(1) - \sum_{j=1}^{N} T_i'(\tau_j)\omega_j) = T_i(1) - \int_0^1 T_i'(\tau) d\tau\\ &= T_i(1) - (T_i(1) - T_i(0)) = T_i(0).\end{aligned} \tag{42}$$

Notice that, if we define the vector $\mathbf{1} = [1, \ldots, 1]^T$, that means the vector full of $1's$. Then the last summation can be written in a matrix form as

$$\mathbf{E} \cdot \mathbf{1} = \mathbf{K}^{\tau,0}, \tag{43}$$

hence $\mathbf{1} = \mathbf{E}^{-1}\mathbf{K}^{\tau,0}$, so matrix associated with the first term in (39), corresponds to $\mathbf{1} \otimes \mathbf{I}$.

Associated with index $i = 1, \ldots, 2n_N,\ j = 1, \ldots, N$ and $k = 0, 1$ we can define the function $\nu(i, j, k) = i + kn_N + (j-1)2n_N$. It helps to evaluate global matrices as follows

$$\begin{aligned}\mathbf{1} \otimes \mathbf{I}_{k+(i-1)2n_N,l} &= \delta_{k,l},\\ ((\mathbf{E}^{-1}\mathbf{M}^\tau) \otimes \mathbf{I})_{k+(i-1)2_{n_N},l+(j-1)2n_N} &= (\mathbf{E}^{-1}\mathbf{M}^\tau)_{i,j}\delta_{k,l},\end{aligned} \tag{44}$$

where $k, l = 1, \ldots, 2n_N$. That means, the only non zero entries are

$$\begin{aligned}(\mathbf{1} \otimes \mathbf{I})_{k+(i-1)2n_N,k} &= 1,\\ ((\mathbf{E}^{-1}\mathbf{M}^\tau) \otimes \mathbf{I})_{k+(i-1)2n_N,k+(j-1)2n_N} &= (\mathbf{E}^{-1}\mathbf{M}^\tau)_{i,j}.\end{aligned} \tag{45}$$

So we already know that $\mathbf{K1}^{-1}\mathbf{Mw}^0\mathcal{W} = (\mathbf{E}^{-1}\mathbf{M}^\tau) \otimes \mathbf{I}\mathcal{W}$ in a component wise

$$\sum_l^{2n_N} (\mathbf{1} \otimes \mathbf{I})_{k+(i-1)2n_N,l} \mathcal{W}_l = \mathcal{W}_k, \tag{46}$$

and similarly $\mathbf{K1}^{-1}\mathbf{M}\mathcal{S} = (\mathbf{E}^{-1}\mathbf{M}^\tau \otimes \mathbf{I})\mathcal{S}$ in a component wise is

$$\begin{aligned}&\sum_{l=1}^{2n_N}\sum_{j=1}^{N} ((\mathbf{E}^{-1}\mathbf{M}^\tau) \otimes \mathbf{I})_{k+(i-1)2n_N,l+(j-1)2n_N} \mathcal{S}_{l+(j-1)2n_N}\\ &\quad = \sum_{j=1}^{N} a_{i,j}\mathcal{S}_{k+(j-1)2n_N},\end{aligned} \tag{47}$$

where $a_{i,j} = (\mathbf{E}^{-1}\mathbf{M}^\tau)_{i,j}$. Now, we may relate coefficients $\mathcal{Q}$, with coefficients $\mathbf{V}^0$ and $\mathbf{V}^x$ as follows

$$\mathcal{Q}_{r+sn_N+(j-1)2n_N} = \hat{\mathbf{V}}^0_{r+(j-1)n_N}(1-s) + \hat{\mathbf{V}}^x_{r+(j-1)n_N} s, \tag{48}$$

where $r = 1, \ldots, n_N,\ i = 1, \ldots, 2n_N,\ s = 0, 1,\ j = 1, \ldots, N$. Similarly we have

$$\begin{aligned}
\mathcal{S}_{r+sn_N+(j-1)2n_N} &= \hat{\mathbf{S}}^0_{r+(j-1)n_N}(1-s) + \hat{\mathbf{S}}^x_{r+(j-1)n_N}s,\\
\mathcal{F}_{r+sn_N+(j-1)2n_N} &= \hat{\mathbf{F}}^0_{r+(j-1)n_N}(1-s) + \hat{\mathbf{F}}^x_{r+(j-1)n_N}s,\\
\mathcal{W}_{r+sn_N} &= \hat{\mathbf{W}}^0_r(1-s) + \hat{\mathbf{W}}^x_r s.
\end{aligned} \tag{49}$$

From Eqs. (46) and (47), we have

$$\begin{aligned}
&\hat{\mathbf{V}}^0_{r+(j-1)n_N}(1-s) + \hat{\mathbf{V}}^x_{r+(j-1)n_N}s = \hat{\mathbf{W}}^0_r(1-s)\\
&+\hat{\mathbf{W}}^x_r s + \sum_{i=1} a_{j,i}(\hat{\mathbf{S}}^0_{r+(i-1)n_N}(1-s) + \hat{\mathbf{S}}^x_{r+(i-1)n_N}s)\\
&+\sum \mathbf{K1}^{-1}\mathbf{K}^{\xi}_{r+sn_N+(j-1)2n_N,n+\nu n_N+(i-1)2n_N}\\
&(\hat{\mathbf{F}}^0_{n+(i-1)n_N}(1-\nu) + \hat{\mathbf{F}}^x_{n+(i-1)n_N}\nu),
\end{aligned} \tag{50}$$

defining

$$\begin{aligned}
\mathbf{FH}^{\mathbf{0}}_{\mathbf{r}+sn_N+(\mathbf{j}-\mathbf{1})2n_N} &:= \sum \mathbf{K1}^{-1}\mathbf{K}^{\xi}_{r+sn_N+(j-1)2n_N,n+(i-1)2n_N}\hat{\mathbf{F}}^0_{n+(i-1)n_N},\\
\mathbf{FH}^{x}_{r+sn_N+(j-1)2n_N} &:= \sum \mathbf{K1}^{-1}\mathbf{K}^{\xi}_{r+sn_N+(j-1)2n_N,n+n_N+(i-1)2n_N}\hat{\mathbf{F}}^x_{n+(i-1)n_N},
\end{aligned} \tag{51}$$

we have

$$\begin{aligned}
\mathbf{RF}^{\mathbf{0}}_{\mathbf{r}+(\mathbf{j}-\mathbf{1})n_N} &:= \mathbf{FH}^0_{r+(j-1)2n_N} + \mathbf{FH}^x_{r+(j-1)2n_N},\\
\mathbf{RF}^x_{r+(j-1)n_N} &:= \mathbf{FH}^0_{r+n_N+(j-1)2n_N} + \mathbf{FH}^x_{r+n_N+(j-1)2n_N}.
\end{aligned} \tag{52}$$

So, can divide

$$\begin{aligned}
&\sum \mathbf{K1}^{-1}\mathbf{K}^{\xi}_{r+sn_N+(j-1)2n_N,n+\nu n_N+(i-1)2n_N}\mathcal{F}_{n+\nu n_N+(i-1)2n_N}\\
&\quad = \mathbf{RF}^0_{r+(j-1)n_N}(1-s) + \mathbf{RF}^x_{r+(j-1)n_N}s,
\end{aligned} \tag{53}$$

so, finally

$$\begin{aligned}
&\hat{\mathbf{V}}^0_{r+(j-1)n_N}(1-s) + \hat{\mathbf{V}}^x_{r+(j-1)n_N}s\\
&= (1-s)\left(\hat{\mathbf{W}}^0_r + \sum_{i=1} a_{j,i}\hat{\mathbf{S}}^0_{r+(i-1)n_N} + \mathbf{RF}^0_{r+(j-1)n_N}\right)\\
&+s\left(\hat{\mathbf{W}}^x_r + \sum_{i=1} a_{j,i}\hat{\mathbf{S}}^x_{r+(i-1)n_N}\right) + \mathbf{RF}^x_{r+(j-1)n_N}).
\end{aligned} \tag{54}$$

Hence separating for $s = 1$ and $s = 0$, we have

$$\begin{aligned}
\hat{\mathbf{V}}^0_{r+(j-1)n_N} &= \hat{\mathbf{W}}^0_r + \sum_{i=1} a_{j,i}\hat{\mathbf{S}}^0_{r+(i-1)n_N} + \mathbf{RF}^0_{r+(j-1)n_N},\\
\hat{\mathbf{V}}^x_{r+(j-1)n_N} &= \hat{\mathbf{W}}^x_r + \sum_{i=1} a_{j,i}\hat{\mathbf{S}}^x_{r+(i-1)n_N}) + \mathbf{RF}^x_{r+(j-1)n_N}.
\end{aligned} \tag{55}$$

□

In Appendix A, the equations for the degrees of freedom for second, third and fourth order of accuracy are shown, where the structure evidenced in the previous proposition can be seen.

If the source term is not stiff, the previous step is very fast and yields the accuracy in a reduced number of iterations. For a scheme of order N we have used N iterations. If the source term is stiff the iteration procedure may require a more sophisticated strategy to get convergence, in this approach it is done in a nested form. The following section shows the proposed strategy for very stiff cases, the structure of matrices allows us to derive an efficient inversion strategy.

3.3. Stiff source term

The DG version of the present formulation relates the degrees of freedom, source terms and fluxes as follows

$$\begin{aligned}
\hat{\mathbf{V}}^0_{i+(j-1)n_N} &= \hat{\mathbf{W}}^0_i + \sum_{k=1}^{N} a_{j,k}\mathbf{S}^*(\hat{\mathbf{V}}^0_{i+(k-1)n_N}) + \mathbf{RF}^0_{i+(j-1)n_N}(\mathbf{F}^0, \mathbf{F}^x),\\
\hat{\mathbf{V}}^x_{i+(j-1)n_N} &= \hat{\mathbf{W}}^x_i + \sum_{k=1}^{N} a_{j,k}\hat{\mathbf{S}}^x_{i+(k-1)n_N} + \mathbf{RF}^x_{i+(j-1)n_N}(\mathbf{F}^0, \mathbf{F}^x),
\end{aligned} \tag{56}$$

where

$$\hat{\mathbf{S}}^x_{i+(k-1)n_N} = \mathbf{B}^*(\hat{\mathbf{V}}^0_{i+(j-1)n_N})\hat{\mathbf{V}}^x_{i+(k-1)n_N},$$

$\mathbf{RF}^0$ and $\mathbf{RF}^x$ are functionals which only depend on the fluxes and their gradients, with $i = 1, \ldots, n_N,\ j = 1, \ldots, N$.

Notice that in (56) both of the equations are coupled in $\mathbf{RF}^0$ and $\mathbf{RF}^x$. However the first equation only contains evaluations of the source on nodes $\hat{\mathbf{V}}^0$ and in the second equation, terms $\hat{\mathbf{S}}^x$ are related with $\hat{\mathbf{V}}^0$ through the Jacobian of the source term. However, a suitable nested-type fixed-point iteration procedure can be constructed such that the iterations are carried out on decoupled systems for $\hat{\mathbf{V}}^0$ and $\hat{\mathbf{V}}^x$, in this sense we say that the system is weakly coupled. We propose the following Picard-type iteration procedure:

$$\left.\begin{aligned}
\hat{\mathbf{V}}^{0,r+1}_{i+(j-1)n_N} &= \hat{\mathbf{W}}^0_i + \textstyle\sum_{k=1}^{N} a_{j,k}\mathbf{S}^*(\hat{\mathbf{V}}^{0,r+1}_{i+(k-1)n_N}) + \mathbf{RF}^{0,r}_{i+(j-1)n_N},\\
\hat{\mathbf{V}}^{x,r+1}_{i+(j-1)n_N} &= \hat{\mathbf{W}}^x_i + \textstyle\sum_{k=1}^{N} a_{j,k}\mathbf{B}^*(\hat{V}^{0,r}_{i+(k-1)n_N})\hat{\mathbf{V}}^{x,r+1}_{i+(k-1)n_N}\\
&\quad +\mathbf{RF}^{x,r}_{i+(j-1)n_N},
\end{aligned}\right\} \tag{57}$$

here r stands by an iteration index.

(1) Given $\hat{\mathbf{V}}^{0,r}_{i+(k-1)n_N}$ and $\hat{\mathbf{V}}^{x,r}_{i+(k-1)n_N}$, find $\hat{\mathbf{V}}^{0,r+1}_{i+(k-1)n_N}$ and $\hat{\mathbf{V}}^{x,r+1}_{i+(k-1)n_N}$ as the roots of non-lineal algebraic system $\mathbf{H}^0$ and $\mathbf{H}^x$, defined in a component wise by

$$\left.\begin{aligned}
\mathbf{H}^0_{i+(j-1)n_N}(\hat{\mathbf{V}}^{0,r+1}, \hat{\mathbf{V}}^{x,r}) &:= \textstyle\sum_{k=1}^{N} a_{j,k}\mathbf{S}^*(\hat{\mathbf{V}}^{0,r+1}_{i+(k-1)n_N})\\
&\quad -\hat{\mathbf{V}}^{0,r+1}_{i+(j-1)n_N} + \hat{\mathbf{W}}^0_i + \mathbf{RF}^{0,r}_{i+(j-1)n_N},\\
\mathbf{H}^x_{i+(j-1)n_N}(\hat{\mathbf{V}}^{0,r}, \hat{\mathbf{V}}^{x,r+1}) &= \textstyle\sum_{k=1}^{N} a_{j,k}\mathbf{B}^*(\hat{\mathbf{V}}^{0,r}_{i+(k-1)n_N})\hat{\mathbf{V}}^{x,r+1}_{i+(k-1)n_N}\\
&\quad -\hat{\mathbf{V}}^{x,r+1}_{i+(j-1)n_N} + \hat{\mathbf{W}}^x_i + \mathbf{RF}^{x,r}_{i+(j-1)n_N}.
\end{aligned}\right\} \tag{58}$$

(2) Compute the Jacobian matrix associated with both systems, $\mathbf{H}^0$ and $\mathbf{H}^x$

$$\begin{aligned}
\partial\mathbf{H}^0_{i+(j-1)n_N}(\hat{\mathbf{V}}^{0,r+1}, \hat{\mathbf{V}}^{x,r})/\partial\hat{\mathbf{V}}^{0,r+1} &= -(\tilde{\mathbf{I}} - \tilde{\mathbf{B}}(\mathbf{V}^0)),\\
\partial\mathbf{H}^x_{i+(j-1)n_N}(\hat{\mathbf{V}}^{0,r}, \hat{\mathbf{V}}^{x,r+1})/\partial\hat{\mathbf{V}}^{x,r+1} &= -(\tilde{\mathbf{I}} - \tilde{\mathbf{B}}(\mathbf{V}^0)).
\end{aligned} \tag{59}$$

Notice that, both Jacobian matrices have the form, $-(\tilde{\mathbf{I}} - \tilde{\mathbf{B}}(\mathbf{V}^0))$, where

$$\tilde{\mathbf{I}} = \begin{bmatrix}
\mathbf{I} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0}\\
\mathbf{0} & \mathbf{I} & \ldots & \mathbf{0} & \mathbf{0}\\
\vdots & \vdots & \ddots & \vdots & \vdots\\
\mathbf{0} & \mathbf{0} & \ldots & \mathbf{I} & \mathbf{0}\\
\mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{I}
\end{bmatrix} \tag{60}$$

and

$$\tilde{\mathbf{B}}(\mathbf{V}^0) = \begin{bmatrix}
a_{1,1}\bar{\mathbf{B}}_1 & a_{1,2}\bar{\mathbf{B}}_2 & \ldots & a_{1,N-1}\bar{\mathbf{B}}_{N-1} & a_{1,N}\bar{\mathbf{B}}_N\\
a_{2,1}\bar{\mathbf{B}}_1 & a_{2,2}\bar{\mathbf{B}}_2 & \ldots & a_{2,N-1}\bar{\mathbf{B}}_{N-1} & a_{2,N}\bar{\mathbf{B}}_N\\
\vdots & \vdots & \ddots & \vdots & \vdots\\
a_{N-1,1}\bar{\mathbf{B}}_1 & a_{N-1,2}\bar{\mathbf{B}}_2 & \ldots & a_{N-1,N-1}\bar{\mathbf{B}}_{N-1} & a_{N-1,N}\bar{\mathbf{B}}_N\\
a_{N,1}\bar{\mathbf{B}}_1 & a_{N,2}\bar{\mathbf{B}}_2 & \ldots & a_{N,N-1}\bar{\mathbf{B}}_{N-1} & \mathbf{0}
\end{bmatrix}, \tag{61}$$

with $\bar{\mathbf{B}}_k := \mathbf{B}^*(\hat{\mathbf{V}}^0_{i+(k-1)n_N})$. We design an inversion strategy for $(\tilde{\mathbf{I}} - \tilde{\mathbf{B}}(\mathbf{V}^0))$, which allows us to obtain δ^0 and δ^x as the solution of the linear systems

$$(\tilde{\mathbf{I}} - \tilde{\mathbf{B}}(\mathbf{V}^0))\delta^0 = \mathbf{H}^0(\mathbf{V}^0)\ ,$$

$$(\tilde{\mathbf{I}} - \tilde{\mathbf{B}}(\mathbf{V}^0))\delta^x = \mathbf{H}^k(\mathbf{V}^x). \tag{62}$$

The strategy only requires the inversions of $N-1$ matrices of size $m \times m$. See Appendix B for further details.

(3) Update the solution $\mathbf{V}^{0,r+1} = \mathbf{V}^{0,r} + \delta^{0,r}$ and $\mathbf{V}^{x,r+1} = \mathbf{V}^{x,r} + \delta^{x,r}$ Compute $\mathbf{RF}^0$ and $\mathbf{RF}^x$ using updated $\mathbf{V}^{0,r+1}$ and $\mathbf{V}^{x,r+1}$, go to step 1).

For each order of accuracy, N, the procedure is done in $N+1$ iterations.

3.4. Continuous Galerking scheme

To build a continuous Galerkin scheme for (13), we set the following temporal nodes

$$\begin{aligned} \tau_1 &= 0, \\ \tau_2 &= \chi_1, \\ \tau_3 &= \chi_2, \\ &\vdots \\ \tau_{n_{GP}+1} &= \chi_{n_{GP}}, \end{aligned} \tag{63}$$

Then, we build a set of Lagrange polynomials by following (19). We define the space-time polynomials (20).

In the continuous Galerkin approach, we must have

$$\mathbf{W}(x(\xi)) = \tilde{\mathbf{Q}}(\xi, 0), \tag{64}$$

where $x(\xi) = x_{i-\frac{1}{2}} + (\xi + \frac{1}{2})\Delta x$. So, evaluating at $(\xi_i, 0)$ we set the n_N coefficients

$$\hat{\mathbf{V}}^0_{\alpha(i,1)} = \mathbf{W}(x(\xi_i)). \tag{65}$$

Now, if we differentiate $\mathbf{W}(x(\xi))$ and equate to $\tilde{\mathbf{Q}}(\xi_k, 0)$ we obtain

$$\hat{\mathbf{V}}^x_{\alpha(i,1)} = \partial_\xi \tilde{\mathbf{Q}}(\xi_i, 0) = \frac{d\mathbf{W}(x(\xi_i))}{d\xi}\frac{1}{\Delta x} \tag{66}$$

Now we can set the problem in a matrix form

$$\mathbf{K1}\mathcal{Q} + \mathbf{K}^\xi \mathcal{F} = \mathbf{M}\mathcal{S} + \mathbf{M0}, \tag{67}$$

where

$$\begin{aligned} \mathcal{Q}_{r(i,j)} &= \hat{\mathbf{V}}^0_{\alpha(i,j)}, \\ \mathcal{Q}_{r(i+n_N,j)} &= \hat{\mathbf{V}}^x_{\alpha(i,j)} \end{aligned} \tag{68}$$

and

$$\begin{aligned} \mathcal{F}_{r(i,j)} &= \hat{\mathbf{F}}^0_{\alpha(i,j)}, \\ \mathcal{F}_{r(i+n_N,j)} &= \hat{\mathbf{F}}^x_{\alpha(i,j)}, \\ \mathcal{S}_{r(i,j)} &= \hat{\mathbf{S}}^0_{\alpha(i,j)}, \\ \mathcal{S}_{r(i+n_N,j)} &= \hat{\mathbf{S}}^x_{\alpha(i,j)}, \end{aligned} \tag{69}$$

for $i = 1 \ldots, n_N$ and $j = 1, \ldots, n_{GP}+1$. $\mathbf{M0}_k = \int_{-\frac{1}{2}}^{\frac{1}{2}} \tilde{\mathbf{Q}}(\xi, 0)\phi_k(\xi, 0)d\xi$. and

$$\begin{aligned} \mathbf{K1}_{k,l} &= [\phi_k, \phi_l]_1 - < \phi_k, \partial_\tau \phi_l >, \\ \mathbf{K}^\xi_{k,l} &= < \partial_\xi \phi_k, \phi_l >, \\ \mathbf{M}_{k,l} &= < \phi_k, \phi_l > . \end{aligned} \tag{70}$$

Notice that by mean of (68) we can get coefficients $\hat{\mathbf{V}}^0_{\alpha(i,j)}$ and $\hat{\mathbf{V}}^x_{\alpha(i,j)}$ from $\mathcal{Q}$. Similarly to the discontinuous Galerkin scheme, the solution of the degrees of freedom yield

$$\hat{\mathbf{V}}^0_{i+(j-1)n_N} = \hat{\mathbf{V}}^0_i + \sum_{k=1}^{N} a_{j,k}\mathbf{S}^*(\hat{V}^0_{i+(k-1)n_N}) + \mathbf{RF}^0_{i+(j-1)n_N}(F^0, F^x),$$

$$\hat{\mathbf{V}}^x_{i+(j-1)n_N} = \hat{\mathbf{V}}^x_i + \sum_{k=1}^{N} a_{j,k}\hat{\mathbf{S}}^x_{i+(k-1)n_N} + \mathbf{RF}^x_{i+(j-1)n_N}(F^0, F^x), \tag{71}$$

for $j = 2, \ldots, N$ and $i = 1, \ldots, n_N$. Please note the similitude between the Eq. (35) with the Eq. (71). In this case, we suggest the simple iteration procedure

$$\left.\begin{aligned} \hat{\mathbf{V}}^{0,r+1}_{i+(j-1)n_N} &= \hat{\mathbf{V}}^0_i + \textstyle\sum_{k=1}^{N} a_{j,k}\mathbf{S}^*(\hat{\mathbf{V}}^{0,r}_{i+(k-1)n_N}) + \mathbf{RF}^{0,r}_{i+(j-1)n_N}, \\ \hat{\mathbf{V}}^{x,r+1}_{i+(j-1)n_N} &= \hat{\mathbf{V}}^x_i + \textstyle\sum_{k=1}^{N} a_{j,k}\mathbf{B}^*(\hat{\mathbf{V}}^{0,r}_{i+(k-1)n_N})\hat{\mathbf{V}}^{x,r}_{i+(k-1)n_N} + \mathbf{RF}^{x,r}_{i+(j-1)n_N}, \end{aligned}\right\} \tag{72}$$

for $i = 1, \ldots, n_N$ and $j = 2, \ldots, N$. Here r stands by an iteration index. Notice that, coefficients associated with the first time level, $j = 1$, have been already computed in (65) and (66), so the iteration is to find $(N-1)n_{DOF}$ coefficients by solving $(N-1)n_{DOF}$ equations. This procedure is carried out a finite number of times. For each order of accuracy, N, the procedure is done in N iterations.

4. Numerical results

In this section we solve several conventional hyperbolic balance laws to illustrate the applicability of the present scheme. We will use N_{cell} uniform meshes of size Δx, so given the time level t^n and the cell averages $\{\mathbf{Q}^n_i\}_{i=1}^{N_{cell}}$, the stable time step, Δt, will be obtained as

$$\Delta t = C_{CFL}\frac{\Delta x}{\lambda_{max}}, \tag{73}$$

where $\lambda_{\max} = max_{i=1,\ldots,N_{cells}}\{\bar{\lambda}(\mathbf{Q}^n_i)\}$, with $\bar{\lambda}(\mathbf{W}) = max_{j=1,\ldots,m}|\lambda_j(\mathbf{W})|$, where $\lambda_j(\mathbf{W})$ is the jth eigenvalue of the Jacobian matrix $\mathbf{A}(\mathbf{W})$ of the flux function $\mathbf{F}(\mathbf{W})$ with respect to $\mathbf{W}$. As we are using (73) the order of accuracy of the numerical solution should be $M = \min(n_{GP}+1, n_{DOF})$, we remark that in general we have $n_{GP}+1 \leq n_{DOF}$.

4.1. The linear advection equation

In this section we solve the well known linear advection equation

$$\begin{aligned} &\text{PDE}: \partial_t q(x,t) + \lambda\partial_x q = \beta q \; x \in [0,1], t \in [0, t_{out}], \\ &\text{IC}: \qquad\qquad q(x,0) = \sin(2\pi x), \end{aligned} \tag{74}$$

with λ and $\beta \leq 0$ constant values. This model equation has the exact solution $q(x,t) = \sin(2\pi(x-\lambda t))e^{\beta t}$.

We solve Eq. (74) using $\lambda = 1$, $\beta = -1$, $C_{CFL} = 0.9$. Notice that, the source term is not stiff with the choice of β. Tables 1 and 2, show the empirical convergence rate assessment for this test, where the predictors are computed via CG and DG, respectively. In the figure are tabulated the norms L_{inf}, L_1, L_2 and their respective convergence rates, the last column shows the CPU time for each order of accuracy. Expected theoretical convergence rates are achieved up to fourth order and both numerical schemes have comparable computational costs even at high order.

4.2. Burgers equation with quadratic source term

Let us consider the partial differential equation

$$\left.\begin{aligned} \partial_t q(x,t) + \partial_x\left(\tfrac{q(x,t)^2}{2}\right) &= \beta q(x,t)^2, \\ q(x,t) &= h_0(x). \end{aligned}\right\} \tag{75}$$

which has the exact solution in terms of the initial condition h_0 as

$$q(x,t) = \frac{h_0(y)}{1 - \beta t h_0(y)}, \tag{76}$$

Table 1
Linear advection-reaction equation. CG-predictor output time $t_{out} = 1$ with $C_{cfl} = 0.9$, $\lambda = 1$, $\beta = -1$.

Theoretical order : 2							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$1.48e-01$	0.00	$4.85e-02$	0.00	$7.59e-02$	0.0080
16	2.15	$3.33e-02$	2.67	$7.62e-03$	2.59	$1.26e-02$	0.0280
32	2.06	$7.97e-03$	2.10	$1.77e-03$	2.27	$2.62e-03$	0.0640
64	1.93	$2.09e-03$	2.03	$4.36e-04$	2.02	$6.44e-04$	0.1840
128	1.31	$8.42e-04$	2.04	$1.06e-04$	1.96	$1.65e-04$	0.2080
Theoretical order : 3							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$8.11e-03$	0.00	$4.91e-03$	0.00	$5.73e-03$	0.0120
16	2.93	$1.06e-03$	2.85	$6.82e-04$	2.92	$7.56e-04$	0.0360
32	3.01	$1.32e-04$	3.02	$8.40e-05$	3.02	$9.33e-05$	0.1440
64	3.09	$1.55e-05$	3.09	$9.87e-06$	3.09	$1.10e-05$	0.1640
128	3.00	$1.94e-06$	3.00	$1.24e-06$	3.00	$1.37e-06$	0.4520
Theoretical order : 4							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$1.50e-02$	0.00	$1.03e-02$	0.00	$1.12e-02$	0.0040
16	3.99	$9.40e-04$	4.10	$6.02e-04$	4.06	$6.68e-04$	0.0120
32	4.10	$5.47e-05$	4.10	$3.50e-05$	4.11	$3.88e-05$	0.0400
64	4.05	$3.31e-06$	4.06	$2.10e-06$	4.05	$2.34e-06$	0.1680
128	4.03	$2.03e-07$	4.02	$1.29e-07$	4.03	$1.44e-07$	0.5520

Table 2
Linear advection-reaction equation. DG-predictor, output time $t_{out} = 1$ with $C_{cfl} = 0.9$, $\lambda = 1$, $\beta = -1$.

Theoretical order : 2							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$3.86e-02$	0.00	$2.62e-02$	0.00	$2.72e-02$	0.0040
16	1.65	$1.23e-02$	2.30	$5.32e-03$	2.09	$6.40e-03$	0.0080
32	1.48	$4.42e-03$	1.85	$1.48e-03$	1.78	$1.87e-03$	0.0360
64	1.53	$1.53e-03$	1.97	$3.76e-04$	1.93	$4.90e-04$	0.2080
128	1.48	$5.48e-04$	1.93	$9.86e-05$	1.81	$1.40e-04$	0.2360
Theoretical order : 3							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$1.22e-02$	0.00	$8.10e-03$	0.00	$8.87e-03$	0.0040
16	2.91	$1.62e-03$	2.94	$1.06e-03$	2.93	$1.17e-03$	0.0320
32	2.99	$2.05e-04$	3.02	$1.30e-04$	3.01	$1.45e-04$	0.1160
64	3.04	$2.48e-05$	3.04	$1.58e-05$	3.04	$1.75e-05$	0.1720
128	3.00	$3.10e-06$	3.00	$1.97e-06$	3.00	$2.19e-06$	0.3440
Theoretical order : 4							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$1.46e-02$	0.00	$1.01e-02$	0.00	$1.10e-02$	0.0040
16	4.00	$9.15e-04$	4.11	$5.88e-04$	4.08	$6.51e-04$	0.0080
32	4.11	$5.31e-05$	4.11	$3.39e-05$	4.11	$3.76e-05$	0.0360
64	4.05	$3.20e-06$	4.06	$2.04e-06$	4.06	$2.26e-06$	0.1560
128	4.03	$1.96e-07$	4.03	$1.25e-07$	4.03	$1.39e-07$	0.5520

with y satisfying

$$x = y - \frac{ln(1 - \beta th_0(y))}{\beta}. \tag{77}$$

details about the derivation of this solution can be found in [23].

Tables 3 and 4 show the results for CG and DG predictors, respectively. Using the output time $t_{out} = 0.1$ with $C_{cfl} = 0.9$, $\beta = -1$. We note that in this case the DG predictor produces better results than CG predictors, specially for high order of accuracy (third and fourth order). There is no a large difference in the CPU time computations. However, we observe that DG predictor is slightly more expensive than CG predictor when the number of cells increases, which is expected because in the DG approach the number of degrees of freedom to be computed is larger than the number of degrees of freedom required for the CG approach.

4.3. A system of non-linear hyperbolic balance laws

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

$$\begin{aligned} &\partial_t \mathbf{Q} + \partial_x \mathbf{F}(\mathbf{Q}) = \mathbf{S}(\mathbf{Q}), \\ &\mathbf{Q} = [\sin(2\pi x), \cos(2\pi x)]^T, \end{aligned} \tag{78}$$

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

$$\left.\begin{aligned} \mathbf{F}(\mathbf{Q}) &= \begin{bmatrix} \frac{1}{9}\left(\frac{5}{2}u^2 + v^2 - uv\right) \\ \frac{1}{9}\left(4uv - u^2 + \frac{1}{2}v^2\right) \end{bmatrix}, \quad \mathbf{S}(\mathbf{Q}) = \begin{bmatrix} \beta\left(\frac{2u-v}{3}\right)^2 \\ -\beta\left(\frac{2u-v}{3}\right)^2 \end{bmatrix}, \end{aligned}\right\} \tag{79}$$

where $\beta \leq 0$ is a constant value. The exact solution is given by

$$\begin{aligned} u(x,t) &= w_1(x,t) + w_2(x,t), \\ v(x,t) &= 2w_1(x,t) - w_2(x,t), \end{aligned} \tag{80}$$

Table 3
Burgers equation with quadratic reaction term. CG-predictor output time $t_{out} = 0.1$ with $C_{cfl} = 0.9$, $\beta = -1$.

Theoretical order : 2							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$1.23e-02$	0.00	$3.16e-03$	0.00	$4.60e-03$	0.0160
64	0.70	$7.58e-03$	1.77	$9.29e-04$	1.56	$1.56e-03$	0.0480
128	1.66	$2.41e-03$	1.86	$2.55e-04$	1.78	$4.55e-04$	0.1320
256	1.97	$6.16e-04$	1.98	$6.49e-05$	1.94	$1.19e-04$	0.4440
512	1.98	$1.56e-04$	1.98	$1.65e-05$	1.96	$3.05e-05$	1.8120
Theoretical order : 3							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$1.28e-03$	0.00	$2.30e-04$	0.00	$3.81e-04$	0.0160
64	1.04	$6.21e-04$	2.35	$4.51e-05$	1.83	$1.08e-04$	0.0520
128	2.31	$1.25e-04$	2.82	$6.38e-06$	2.65	$1.71e-05$	0.1920
256	2.77	$1.83e-05$	2.85	$8.82e-07$	2.80	$2.46e-06$	0.7360
512	2.94	$2.39e-06$	2.94	$1.15e-07$	2.91	$3.28e-07$	2.7240
Theoretical order : 4							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$1.75e-03$	0.00	$1.89e-04$	0.00	$4.22e-04$	0.0160
64	2.76	$2.59e-04$	3.21	$2.04e-05$	2.85	$5.86e-05$	0.0600
128	3.57	$2.18e-05$	4.09	$1.20e-06$	4.00	$3.66e-06$	0.2320
256	4.19	$1.20e-06$	4.27	$6.21e-08$	4.15	$2.06e-07$	0.9200
512	3.97	$7.63e-08$	4.15	$3.49e-09$	4.14	$1.17e-08$	3.6880

Table 4
Burgers equation with quadratic reaction term. DG-predictor output time $t_{out} = 0.1$ with $C_{cfl} = 0.9$, $\beta = -1$.

Theoretical order : 2							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$1.70e-02$	0.00	$2.91e-03$	0.00	$4.57e-03$	0.0920
64	1.45	$6.21e-03$	1.78	$8.47e-04$	1.64	$1.46e-03$	0.0760
128	1.51	$2.18e-03$	1.89	$2.28e-04$	1.79	$4.23e-04$	0.1120
256	1.91	$5.80e-04$	1.99	$5.72e-05$	1.95	$1.10e-04$	0.4280
512	1.99	$1.46e-04$	1.98	$1.45e-05$	1.96	$2.82e-05$	1.7080
Theoretical order : 3							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$1.93e-03$	0.00	$2.25e-04$	0.00	$5.05e-04$	0.0160
64	1.47	$7.00e-04$	2.34	$4.43e-05$	2.04	$1.23e-04$	0.0560
128	2.40	$1.33e-04$	2.83	$6.23e-06$	2.66	$1.94e-05$	0.1960
256	2.82	$1.88e-05$	2.81	$8.91e-07$	2.80	$2.78e-06$	0.7320
512	2.92	$2.49e-06$	2.92	$1.18e-07$	2.91	$3.70e-07$	2.7520
Theoretical order : 4							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$1.68e-03$	0.00	$1.79e-04$	0.00	$4.06e-04$	0.0200
64	2.71	$2.57e-04$	3.19	$1.96e-05$	2.82	$5.75e-05$	0.0720
128	3.60	$2.12e-05$	4.10	$1.15e-06$	4.01	$3.58e-06$	0.2520
256	4.16	$1.19e-06$	4.28	$5.91e-08$	4.16	$2.00e-07$	0.9800
512	3.97	$7.61e-08$	4.16	$3.29e-09$	4.14	$1.13e-08$	3.7000

where w_1 and w_2 are the solutions to

$$\begin{aligned} &\partial_t w_1 + w_1 \partial_x(w_1) = 0, \\ &\partial_t w_2 + w_2 \partial_x(w_2) = \beta w_2^2, \end{aligned} \tag{81}$$

where the initial condition for each equation is

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

From Section 4.2, we are able to find the solution to the decoupled system (81) thus to (80).

Tables 5 and 6 show the results for CG and DG predictors, respectively. We observe that the computational costs for meshes in this test are comparable for both CG and DG approaches. However, CPU times for the DG predictor are slightly more expensive than those for the CG predictor when the number of cells increases, as expected. We observe that the expected theoretical orders of accuracy are achieved up to the fourth order.

4.4. The Euler equations

Now let us consider the Euler equations

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, \ \mathbf{F}(\mathbf{Q}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix}. \tag{82}$$

Here, the pressure p is related with the conserved variables through the equation for an ideal gas with $\gamma = 1.4$, which yields

$$p = (\gamma - 1)\left(E - \frac{\rho u^2}{2}\right). \tag{83}$$

The initial condition for this system in terms of non-conservative variables $[\rho, u, p]$, is given by

$$\begin{aligned} &\rho(x,0) = 1 + 0.2\sin(2\pi x), \\ &u(x,0) = 1, \\ &p(x,0) = 2, \end{aligned} \tag{84}$$

Table 5
Non-Linear system. CG-predictor output time $t_{out} = 0.1$ with $C_{cfl} = 0.9$, $\beta = -1$.

Theoretical order : 2							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$3.18e-02$	0.00	$8.51e-03$	0.00	$1.26e-02$	0.1680
64	1.45	$1.16e-02$	1.80	$2.44e-03$	1.63	$4.07e-03$	0.1640
128	1.79	$3.37e-03$	1.95	$6.33e-04$	1.90	$1.09e-03$	0.2880
256	1.97	$8.63e-04$	2.00	$1.58e-04$	1.98	$2.76e-04$	1.1000
512	1.98	$2.18e-04$	2.01	$3.93e-05$	1.99	$6.95e-05$	4.2920
Theoretical order : 3							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$6.75e-03$	0.00	$1.54e-03$	0.00	$2.47e-03$	0.0760
64	1.88	$1.83e-03$	2.54	$2.66e-04$	2.28	$5.10e-04$	0.0880
128	2.52	$3.18e-04$	2.71	$4.05e-05$	2.57	$8.58e-05$	0.3000
256	2.75	$4.73e-05$	2.89	$5.49e-06$	2.82	$1.21e-05$	1.1440
512	2.90	$6.33e-06$	2.95	$7.12e-07$	2.92	$1.60e-06$	4.5240
Theoretical order : 4							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$8.28e-03$	0.00	$1.52e-03$	0.00	$2.71e-03$	0.0240
64	2.74	$1.24e-03$	3.27	$1.57e-04$	2.98	$3.44e-04$	0.1000
128	3.43	$1.15e-04$	3.72	$1.19e-05$	3.64	$2.76e-05$	0.3600
256	4.09	$6.71e-06$	4.26	$6.25e-07$	4.22	$1.48e-06$	1.2440
512	4.26	$3.50e-07$	4.26	$3.26e-08$	4.31	$7.45e-08$	4.8000

Table 6
Non-Linear system. DG-predictor output time $t_{out} = 0.1$ with $C_{cfl} = 0.9$, $\beta = -1$.

Theoretical order : 2							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$3.52e-02$	0.00	$8.66e-03$	0.00	$1.30e-02$	0.0480
64	1.44	$1.30e-02$	1.75	$2.57e-03$	1.58	$4.36e-03$	0.1880
128	1.68	$4.07e-03$	1.94	$6.70e-04$	1.86	$1.20e-03$	0.2880
256	1.93	$1.07e-03$	1.98	$1.70e-04$	1.95	$3.10e-04$	1.0920
512	1.96	$2.75e-04$	2.00	$4.25e-05$	1.97	$7.92e-05$	4.2640
Theoretical order : 3							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$7.23e-03$	0.00	$1.43e-03$	0.00	$2.72e-03$	0.1640
64	2.12	$1.67e-03$	2.63	$2.30e-04$	2.39	$5.18e-04$	0.0840
128	2.53	$2.89e-04$	2.67	$3.62e-05$	2.60	$8.53e-05$	0.3080
256	2.74	$4.31e-05$	2.86	$4.97e-06$	2.83	$1.20e-05$	1.1720
512	2.90	$5.77e-06$	2.94	$6.46e-07$	2.93	$1.58e-06$	4.6240
Theoretical order : 4							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
32	0.00	$8.34e-03$	0.00	$1.55e-03$	0.00	$2.74e-03$	0.0280
64	2.75	$1.24e-03$	3.27	$1.60e-04$	2.98	$3.47e-04$	0.1000
128	3.43	$1.15e-04$	3.73	$1.20e-05$	3.65	$2.77e-05$	0.3760
256	4.10	$6.73e-06$	4.27	$6.23e-07$	4.23	$1.48e-06$	1.3440
512	4.26	$3.51e-07$	4.27	$3.23e-08$	4.32	$7.42e-08$	5.0120

additionally the system is endowed periodic boundary conditions. The exact solution is given by

$$\begin{aligned} \rho(x,t) &= 1 + 0.2\sin(2\pi(x-t)), \\ u(x,t) &= 1, \\ p(x,t) &= 2. \end{aligned} \tag{85}$$

Tables 7 and 8 show the convergence rate for the variable ρ of the Euler equations. We observe that both approaches, DG and CG for the predictor, provides very similar results in terms of accuracy and CPU times.

4.5. The LeVeque-Yee test

We apply the scheme described in Section 3.3 to solve the challenging LeVeque-Yee test [20]. The flux is $f(q) = \lambda q$ and the source term has the expression $s(q) = \beta q(q-1)(q-\frac{1}{2})$, with $\beta \leq 0$, constant. Here we take $\lambda = 1$. The computational domain is [0, 1] and boundary conditions are transmissive. The initial condition is

$$q(x,0) = \begin{cases} 1 & \text{if} \quad x < 0.3, \\ 0 & \text{if} \quad x > 0.3. \end{cases} \tag{86}$$

Fig. 1 shows a comparison between exact (line) and numerical (symbols) solutions of second, third and fourth order of accuracy and the zoom of the solution around $x = 0.6$ and $q = 1$ is depicted in the left superior part of the figure. In this test, we take $t_{out} = 0.3$, for $\beta = -1000$, 100 cells and $C_{cfl} = 0.99$. In this system, the initial condition moves to the right with velocity $v = 1$. We observe a very good agreement between the numerical and the exact solution with a very large CFL. This illustrates that this method councils accuracy and stability.

Table 7
Euler equations. CG-predictor output time $t_{out} = 1$ with $C_{cfl} = 0.9$.

Theoretical order : 2							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$1.20e-01$	0.00	$8.08e-02$	0.00	$8.80e-02$	0.0200
16	2.08	$2.84e-02$	2.14	$1.83e-02$	2.12	$2.02e-02$	0.0520
32	2.32	$5.70e-03$	2.25	$3.84e-03$	2.29	$4.13e-03$	0.2400
64	1.99	$1.44e-03$	2.12	$8.82e-04$	2.09	$9.70e-04$	0.2400
128	2.06	$3.45e-04$	2.02	$2.17e-04$	2.01	$2.41e-04$	0.8480
Theoretical order : 3							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$7.65e-02$	0.00	$5.18e-02$	0.00	$5.63e-02$	0.0160
16	2.46	$1.39e-02$	2.59	$8.63e-03$	2.56	$9.55e-03$	0.1000
32	2.87	$1.90e-03$	2.93	$1.13e-03$	2.92	$1.26e-03$	0.1920
64	2.97	$2.41e-04$	2.99	$1.43e-04$	2.99	$1.59e-04$	0.3360
128	2.99	$3.03e-05$	3.00	$1.79e-05$	3.00	$2.00e-05$	1.3520
Theoretical order : 4							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$6.91e-02$	0.00	$4.73e-02$	0.00	$5.13e-02$	0.0120
16	4.10	$4.03e-03$	4.24	$2.50e-03$	4.21	$2.76e-03$	0.0600
32	4.41	$1.90e-04$	4.48	$1.12e-04$	4.48	$1.24e-04$	0.1880
64	4.26	$9.90e-06$	4.24	$5.90e-06$	4.24	$6.57e-06$	0.5200
128	4.11	$5.73e-07$	4.08	$3.49e-07$	4.08	$3.89e-07$	2.1200

Table 8
Euler equations. DG-predictor output time $t_{out} = 1$ with $C_{cfl} = 0.9$.

Theoretical order : 2							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$1.20e-01$	0.00	$8.08e-02$	0.00	$8.80e-02$	0.0240
16	2.08	$2.84e-02$	2.14	$1.83e-02$	2.12	$2.02e-02$	0.0720
32	2.32	$5.70e-03$	2.25	$3.84e-03$	2.29	$4.13e-03$	0.2400
64	1.99	$1.44e-03$	2.12	$8.82e-04$	2.09	$9.70e-04$	0.2080
128	2.06	$3.45e-04$	2.02	$2.17e-04$	2.01	$2.41e-04$	0.7920
Theoretical order : 3							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$7.65e-02$	0.00	$5.18e-02$	0.00	$5.63e-02$	0.0240
16	2.46	$1.39e-02$	2.59	$8.63e-03$	2.56	$9.55e-03$	0.1280
32	2.87	$1.90e-03$	2.93	$1.13e-03$	2.92	$1.26e-03$	0.1720
64	2.97	$2.41e-04$	2.99	$1.43e-04$	2.99	$1.59e-04$	0.3720
128	2.99	$3.03e-05$	3.00	$1.79e-05$	3.00	$2.00e-05$	1.5760
Theoretical order : 4							
Mesh	L_∞ - err	L_∞- ord	L_1 - err	L_1 - ord	L_2 - err	L_2 - ord	CPU
8	0.00	$6.91e-02$	0.00	$4.73e-02$	0.00	$5.13e-02$	0.0080
16	4.10	$4.03e-03$	4.24	$2.50e-03$	4.21	$2.76e-03$	0.0400
32	4.41	$1.90e-04$	4.48	$1.12e-04$	4.48	$1.24e-04$	0.1520
64	4.26	$9.90e-06$	4.24	$5.90e-06$	4.24	$6.57e-06$	0.6640
128	4.11	$5.73e-07$	4.08	$3.49e-07$	4.08	$3.89e-07$	2.3920

5. Conclusions

Here, we have presented an ADER finite volume scheme, where GRP's have been solved by following an HEOC type approach where the predictor was obtained by using DG as well as CG schemes. The presented family of space-time polynomials implemented in the Galerkin framework resulted into an algebraic equation for the degrees of freedom which are weakly coupled, they are only coupled on the fluxes and gradient of fluxes. It made possible the implementation of nested-type fixed-point procedures, where an efficient matrix inversion procedure was designed from the structure of the Jacobian matrix of the algebraic equation. The projection of flux and source term on the polynomial space is not so simple as in the case of conventional space-time nodal basis, but the projection of these terms on a particular node only involves the manipulation of the degrees of freedoms associated with such node.

ADER schemes have been implemented to solve several test problems, for non-stiff cases, DG schemes, as well as CG schemes, have provided accurate numerical solutions, empirical convergence rate assessments have shown that the present schemes achieve up to fourth order of accuracy in space and time. The computational cost associated with both schemes are comparable and only some differences appear when the number of cells increases. We have observed that CPU times of the DG predictor are slightly more expensive than those for the CG predictor, which is expected because in the DG approach the number of degrees of freedom to be determined is larger than the number of degrees of freedom required for the CG approach. The Leveque-Yee test has been solved, in a very stiff regime with a very large CFL number and a good agreement was observed.

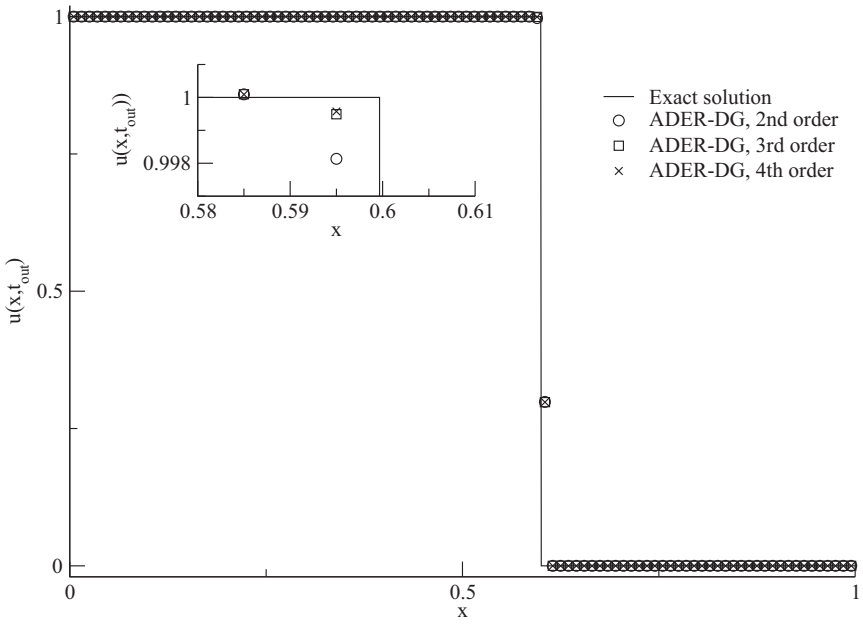

Fig. 1. Leveque and Yee. Parameters $\beta = -1000$, $c_{cfl} = 0.99$ $t_{out} = 0.3$, 100 cells.

Acknowledgments

G.M thanks FONDECYT in the frame of the research project FONDECYT Postdoctorado 2016, number 3160743.

Appendix A. Update of degrees of freedom. Formulae coming from the DG discretization

Here we present a *Fortran 90* code to update coefficients $\hat{V}^0_{i+(j-1)n_N}$ and $\hat{V}^x_{i+(j-1)n_N}$ once flux and source evaluations have been carried out. This approach is designated to be used in a Picard-type iteration loop.

In these subroutines we use indexes like $k = 1, \ldots, N * n_N$, which are related with $i = 1, \ldots, n_N$ and $j = 1, \ldots, N$ as $k = i + (j-1)n_N$. So using this index convention, coefficients $\hat{V}^0_{i+(j-1)n_N}$ and $\hat{V}^x_{i+(j-1)n_N}$ are stored in $V0(k, :)$ and $Vx(k, :)$, respectively. Formulae for second, third and fourth order of accuracy are presented.

The update of $\hat{V}^0_{i+(j-1)n_N}$ is obtained from the following subroutine:

```
Subroutine GetUpdatedDOF_V0 ( Accuracy, W0, F0, Fx, S0, V0)
!-------------------------------------------------------------------
!
Integer          , Intent ( in) :: Accuracy
Double Precision, Intent ( in) :: F0 ( :, :), Fx ( :, :), &
&  W0(:, :), S0 ( :, :)
!
Double Precision, Intent ( output) :: V0 ( :, :)
!
!-------------------------------------------------------------------
!  Input:  "Accuracy" is the order of accuracy to be implemented.
!
! "W0" the modes associated with the reconstruction polynomials. It
!  has dimension (Number of nodes).
!
! "F0, Fx, S0", have dimension ( Modes, Number of variables).
!
!   These are evaluated from V0 and Vx as follows:
!
!   F0 ( k, :) = F ( V0 ( k, :) ),  S0 ( k, :) = S ( V0 ( k, :) ),
!
!   where F ( Q) is the flux function and S ( Q) is the source function.
!
!   Fx ( k, :) = A ( V0 ( k, :) ) * Vx ( k, :),
!
!   with A(Q)the Jacobian matrix of F(Q) with respect to Q.
!
!  Output: "V0" updated coefficients, has
!   dimension ( Modes, Number of variables).
!
!-------------------------------------------------------------------
```

```
Select Case ( Accuracy)
 Case (2)
  !------------------------------------------------------------
  ! Degrees of freedom at 2nd order in space and time.
  !------------------------------------------------------------
  V0 ( 1, :) = -S0 ( 2, :)/6.0 + Fx ( 2, :) / 6.0 + W0 ( 1, :) &
  & + 2.0 * S0 ( 1 , : ) / 3.0 - 2.0 * Fx ( 1, :) / 3.0
  !
  V0 ( 2, :) = W0 ( 1, :) + S0 ( 1, :) - Fx ( 1, :)
  !------------------------------------------------------------

 Case ( 3)
  !------------------------------------------------------------
  ! Degrees of freedom at 4th order in space  and 3rd order in time.
  !------------------------------------------------------------
  V0(1,:) =((sqrt(3)-18)*Fx(1,: )+(18-sqrt(3))*S0(1,: )+60*W0(1,&
       &: )+(11*sqrt(3)-12)*Fx(3,:)+(12-11*sqrt(3))*S0(3,: )-2*sqrt(3&
       &)*Fx(5,: )+2*sqrt(3)*S0(5,: ))/60.0

  V0(2,:) =((sqrt(3)-18)*Fx(2,: )+(18-sqrt(3))*S0(2,: )+60*W0(2,&
       &: )+(11*sqrt(3)-12)*Fx(4,:)+(12-11*sqrt(3))*S0(4,: )-2*sqrt(3&
       &)*Fx(6,: )+2*sqrt(3)*S0(6,: ))/60.0

  V0(3,:) =-((11*sqrt(3)+12)*Fx(1,: )+((-11*sqrt(3))-12)*S0(1,: &
       &)-60*W0(1,: )+(sqrt(3)+18)*Fx(3,:)+((-sqrt(3))-18)*S0(3,: )-2&
       &*sqrt(3)*Fx(5,: )+2*sqrt(3)*S0(5,: ))/60.0

  V0(4,:) =-((11*sqrt(3)+12)*Fx(2,: )+((-11*sqrt(3))-12)*S0(2,: &
       &)-60*W0(2,: )+(sqrt(3)+18)*Fx(4,:)+((-sqrt(3))-18)*S0(4,: )-2&
       &*sqrt(3)*Fx(6,: )+2*sqrt(3)*S0(6,: ))/60.0


  V0(5,:) =((-Fx(1,: ))+S0(1,: )+2*W0(1,: )-Fx(3,: )+S0(3,: )&
       &)/2.0

  V0(6,:) =((-Fx(2,: ))+S0(2,: )+2*W0(2,: )-Fx(4,: )+S0(4,: )&
       &)/2.0
  !------------------------------------------------------------

 Case ( 4)
  !------------------------------------------------------------
  ! Degrees of freedom at 4th order in space  and 4th order in time.
  !------------------------------------------------------------
  V0(1,:) = -((205-2*3**(3.0/2.0)*sqrt(5))*Fx(1,:  )+(2*3**(3.0/2.0&
       &)*sqrt(5)-205)*S0(1,:  )-1260*W0(1,:  )+(256-28*3**(3.0/2.0)*sqrt&
       &(5))*Fx(3,:  )+(28*3**(3.0/2.0)*sqrt(5)-256)*S0(3,:  )+(205-4*3**&
       &(5.0/2.0)*sqrt(5))*Fx(5,:  )+(4*3**(5.0/2.0)*sqrt(5)-205)*S0(5,: &
       & )-36*Fx(7,:  )+36*S0(7,:  ))/1260.0

  V0(2,:) = -((205-2*3**(3.0/2.0)*sqrt(5))*Fx(2,:  )+(2*3**(3.0/2.0&
       &)*sqrt(5)-205)*S0(2,:  )-1260*W0(2,:  )+(256-28*3**(3.0/2.0)*sqrt&
       &(5))*Fx(4,:  )+(28*3**(3.0/2.0)*sqrt(5)-256)*S0(4,:  )+(205-4*3**&
       &(5.0/2.0)*sqrt(5))*Fx(6,:  )+(4*3**(5.0/2.0)*sqrt(5)-205)*S0(6,: &
       & )-36*Fx(8,:  )+36*S0(8,:  ))/1260.0

  V0(3,:) = (((-8*3**(3.0/2.0)*sqrt(5))-55)*Fx(1,:  )+(8*3**(3.0/2.&
       &0)*sqrt(5)+55)*S0(1,:  )+504*W0(1,:  )-124*Fx(3,:  )+124*S0(3,:  &
       &)+(8*3**(3.0/2.0)*sqrt(5)-55)*Fx(5,:  )+(55-8*3**(3.0/2.0)*sqrt(5&
       &))*S0(5,:  )-18*Fx(7,:  )+18*S0(7,:  ))/504.0

  V0(4,:) = (((-8*3**(3.0/2.0)*sqrt(5))-55)*Fx(2,:  )+(8*3**(3.0/2.&
       &0)*sqrt(5)+55)*S0(2,:  )+504*W0(2,:  )-124*Fx(4,:  )+124*S0(4,:  &
       &)+(8*3**(3.0/2.0)*sqrt(5)-55)*Fx(6,:  )+(55-8*3**(3.0/2.0)*sqrt(5&
       &))*S0(6,:  )-18*Fx(8,:  )+18*S0(8,:  ))/504.0

  V0(5,:) = -((4*3**(5.0/2.0)*sqrt(5)+205)*Fx(1,:  )+((-4*3**(5.0/2&
       &.0)*sqrt(5))-205)*S0(1,:  )-1260*W0(1,:  )+(28*3**(3.0/2.0)*sqrt(&
       &5)+256)*Fx(3,:  )+((-28*3**(3.0/2.0)*sqrt(5))-256)*S0(3,:  )+(2*3&
       &**(3.0/2.0)*sqrt(5)+205)*Fx(5,:  )+((-2*3**(3.0/2.0)*sqrt(5))-205&
       &)*S0(5,:  )-36*Fx(7,:  )+36*S0(7,:  ))/1260.0
```

```
V0(6,:) = -((4*3**(5.0/2.0)*sqrt(5)+205)*Fx(2,:  )+((-4*3**(5.0/2&
     &.0)*sqrt(5))-205)*S0(2,:  )-1260*W0(2,:  )+(28*3**(3.0/2.0)*sqrt(&
     &5)+256)*Fx(4,:  )+((-28*3**(3.0/2.0)*sqrt(5))-256)*S0(4,:  )+(2*3&
     &**(3.0/2.0)*sqrt(5)+205)*Fx(6,:  )+((-2*3**(3.0/2.0)*sqrt(5))-205&
     &)*S0(6,:  )-36*Fx(8,:  )+36*S0(8,:  ))/1260.0

V0(7,:) = ((-5*Fx(1,:  ))+5*S0(1,:  )+18*W0(1,:  )-8*Fx(3,:  )+8*&
     &S0(3,:  )-5*Fx(5,:  )+5*S0(5,:  ))/18.0

V0(8,:) = ((-5*Fx(2,:  ))+5*S0(2,:  )+18*W0(2,:  )-8*Fx(4,:  )+8*&
     &S0(4,:  )-5*Fx(6,:  )+5*S0(6,:  ))/18.0

  !-------------------------------------------------------------
 Case Default
  !
  Print*,"Not yet implemented. Accuracy = ", Accuracy
  Stop
  !
  !-------------------------------------------------------------
End Select
!-------------------------------------------------------------------
End Subroutine GetUpdatedDOF_V0
```

The update of $\hat{V}^x_{i+(j-1)n_N}$ is obtained from the following subroutine:

```
Subroutine GetUpdatedDOF_Vx ( Accuracy, Wx, F0, Fx, Sx, Vx)
!-------------------------------------------------------------------
!
Integer         , Intent ( in) :: Accuracy
Double Precision, Intent ( in) :: F0 ( :, :), Fx ( :, :), &
&  Wx ( :, :), Sx ( :, :)
!
Double Precision, Intent ( output) :: Vx ( :, :)
!
!-------------------------------------------------------------------
!  Input:  "Accuracy" is the order of accuracy to be implemented.
!
! "Wx" the modes associated with the reconstruction polynomials. It
!  has dimension (Number of nodes).
!
! "F0, Fx, S0, Sx", have dimension ( Modes, Number of variables).
!
!   These are evaluated from V0 and Vx as follows:
!
!   F0 ( k, :) = F ( V0 ( k, :) )
!
!   where F ( Q) is the flux function and
!
!   Fx ( k, :) = A ( V0 ( k, :) ) * Vx ( k, :),
!   Sx ( k, :) = B ( V0 ( k, :) ) * Vx ( k, :),
!
!   with A(Q) the Jacobian matrix of F(Q) and B(Q) the Jacobian matrix
!  of the source function S(Q) with respect to Q.
!
!  Output: "V0" updated coefficients, has
!   dimension ( Modes, Number of variables).
!-------------------------------------------------------------------
Select Case (Accuracy)
 Case ( 2)
  !-------------------------------------------------------------
  ! Algebraic equation for degrees of freedom at second order in
  ! space and time.
!-------------------------------------------------------------------
Vx(1,:) = - Sx ( 2, :) / 6.0 + Wx ( 1, :) + 2.0 * Sx ( 1, :)  / 3.0
Vx(2,:) = Wx ( 1, :) + Sx (1, :)
!-------------------------------------------------------------------
Case ( 3)
```

```
!-----------------------------------------------------------------
! Algebraic equation for for degrees of freedom at 4th order in
! space  and 3rd order in time.
!-----------------------------------------------------------------
Vx(1,:) = -((48*F0(6,:  )-8*Fx(6,:  )-48*F0(5,:  )-2*Sx(5,:  )-16&
     &*Fx(5,:  )+(32*3**(3.0/2.0)-264)*F0(4,:  )+(44-16*sqrt(3))*Fx(4,:&
     &  )+(264-32*3**(3.0/2.0))*F0(3,:  )+(11-4*sqrt(3))*Sx(3,:  )+(88-&
     &32*sqrt(3))*Fx(3,:  )+(16*3**(5.0/2.0)-24)*F0(2,:  )+(4-8*3**(3.0&
     &/2.0))*Fx(2,:  )+(24-16*3**(5.0/2.0))*F0(1,:  )-20*sqrt(3)*Wx(1,:&
     &  )+(1-2*3**(3.0/2.0))*Sx(1,:  )+(8-16*3**(3.0/2.0))*Fx(1,:  ))/s&
     &qrt(3))/20.0

Vx(2,:) = ((48*F0(6,:  )+2*Sx(6,:  )-16*Fx(6,:  )-48*F0(5,:  )-8*&
     &Fx(5,:  )+(32*3**(3.0/2.0)-264)*F0(4,:  )+(4*sqrt(3)-11)*Sx(4,:  &
     &)+(88-32*sqrt(3))*Fx(4,:  )+(264-32*3**(3.0/2.0))*F0(3,:  )+(44-1&
     &6*sqrt(3))*Fx(3,:  )+(16*3**(5.0/2.0)-24)*F0(2,:  )+20*sqrt(3)*Wx&
     &(2,:  )+(2*3**(3.0/2.0)-1)*Sx(2,:  )+(8-16*3**(3.0/2.0))*Fx(2,:  &
     &)+(24-16*3**(5.0/2.0))*F0(1,:  )+(4-8*3**(3.0/2.0))*Fx(1,:  ))/sq&
     &rt(3))/20.0

Vx(3,:) = ((48*F0(6,:  )-8*Fx(6,:  )-48*F0(5,:  )-2*Sx(5,:  )-16*&
     &Fx(5,:  )+((-24)-16*3**(5.0/2.0))*F0(4,:  )+(4+8*3**(3.0/2.0))*Fx&
     &(4,:  )+(24+16*3**(5.0/2.0))*F0(3,:  )+(1+2*3**(3.0/2.0))*Sx(3,: &
     & )+(8+16*3**(3.0/2.0))*Fx(3,:  )+((-264)-32*3**(3.0/2.0))*F0(2,: &
     & )+(44+16*sqrt(3))*Fx(2,:  )+(264+32*3**(3.0/2.0))*F0(1,:  )+20*s&
     &qrt(3)*Wx(1,:  )+(11+4*sqrt(3))*Sx(1,:  )+(88+32*sqrt(3))*Fx(1,: &
     & ))/sqrt(3))/20.0

Vx(4,:) = -((48*F0(6,:  )+2*Sx(6,:  )-16*Fx(6,:  )-48*F0(5,:  )-8&
     &*Fx(5,:  )+((-24)-16*3**(5.0/2.0))*F0(4,:  )+((-1)-2*3**(3.0/2.0)&
     &)*Sx(4,:  )+(8+16*3**(3.0/2.0))*Fx(4,:  )+(24+16*3**(5.0/2.0))*F0&
     &(3,:  )+(4+8*3**(3.0/2.0))*Fx(3,:  )+((-264)-32*3**(3.0/2.0))*F0(&
     &2,:  )-20*sqrt(3)*Wx(2,:  )+((-11)-4*sqrt(3))*Sx(2,:  )+(88+32*sq&
     &rt(3))*Fx(2,:  )+(264+32*3**(3.0/2.0))*F0(1,:  )+(44+16*sqrt(3))*&
     &Fx(1,:  ))/sqrt(3))/20.0

Vx(5,:) = -((-8*Fx(1,:  ))-Sx(1,:  )-2*Wx(1,:  )-24*F0(1,:  )-4*F&
     &x(2,:  )+24*F0(2,:  )-8*Fx(3,:  )-Sx(3,:  )-24*F0(3,:  )-4*Fx(4,:&
     &  )+24*F0(4,:  ))/2.0

Vx(6,:) = ((-4*Fx(1,:  ))-24*F0(1,:  )-8*Fx(2,:  )+Sx(2,:  )+2*Wx&
     &(2,:  )+24*F0(2,:  )-4*Fx(3,:  )-24*F0(3,:  )-8*Fx(4,:  )+Sx(4,: &
     & )+24*F0(4,:  ))/2.0
!-----------------------------------------------------------------
```

```
Case ( 4)
 !-------------------------------------------------------------
 ! Algebraic equation for for degrees of freedom at 4th order in
 ! space  and 4th order in time.
 !-------------------------------------------------------------

Vx(1,:) = (5**((-3.0)/2.0)*((328*5**(3.0/2.0)-80*3**(3.0/2.0))*Fx&
     &(1,:  )+(41*5**(3.0/2.0)-10*3**(3.0/2.0))*Sx(1,:  )+252*5**(3.0/2&
     &.0)*Wx(1,:  )+(984*5**(3.0/2.0)-80*3**(5.0/2.0))*F0(1,:  )+(164*5&
     &**(3.0/2.0)-40*3**(3.0/2.0))*Fx(2,:  )+(80*3**(5.0/2.0)-984*5**(3&
     &.0/2.0))*F0(2,:  )+(2048*sqrt(5)-1120*3**(3.0/2.0))*Fx(3,:  )+(25&
     &6*sqrt(5)-140*3**(3.0/2.0))*Sx(3,:  )+(6144*sqrt(5)-1120*3**(5.0/&
     &2.0))*F0(3,:  )+(1024*sqrt(5)-560*3**(3.0/2.0))*Fx(4,:  )+(1120*3&
     &**(5.0/2.0)-6144*sqrt(5))*F0(4,:  )+(328*5**(3.0/2.0)-160*3**(5.0&
```

```
     &/2.0))*Fx(5,:  )+(41*5**(3.0/2.0)-20*3**(5.0/2.0))*Sx(5,:  )+(984&
     &*5**(3.0/2.0)-160*3**(7.0/2.0))*F0(5,:  )+(164*5**(3.0/2.0)-80*3*&
     &*(5.0/2.0))*Fx(6,:  )+(160*3**(7.0/2.0)-984*5**(3.0/2.0))*F0(6,: &
     & )-288*sqrt(5)*Fx(7,:  )-36*sqrt(5)*Sx(7,:  )-864*sqrt(5)*F0(7,: &
     & )-144*sqrt(5)*Fx(8,:  )+864*sqrt(5)*F0(8,:  )))/252.0


Vx(2,:) = -(5**((-3.0)/2.0)*((164*5**(3.0/2.0)-40*3**(3.0/2.0))*F&
     &x(1,:  )+(984*5**(3.0/2.0)-80*3**(5.0/2.0))*F0(1,:  )+(328*5**(3.&
     &0/2.0)-80*3**(3.0/2.0))*Fx(2,:  )+(10*3**(3.0/2.0)-41*5**(3.0/2.0&
     &))*Sx(2,:  )-252*5**(3.0/2.0)*Wx(2,:  )+(80*3**(5.0/2.0)-984*5**(&
     &3.0/2.0))*F0(2,:  )+(1024*sqrt(5)-560*3**(3.0/2.0))*Fx(3,:  )+(61&
     &44*sqrt(5)-1120*3**(5.0/2.0))*F0(3,:  )+(2048*sqrt(5)-1120*3**(3.&
     &0/2.0))*Fx(4,:  )+(140*3**(3.0/2.0)-256*sqrt(5))*Sx(4,:  )+(1120*&
     &3**(5.0/2.0)-6144*sqrt(5))*F0(4,:  )+(164*5**(3.0/2.0)-80*3**(5.0&
     &/2.0))*Fx(5,:  )+(984*5**(3.0/2.0)-160*3**(7.0/2.0))*F0(5,:  )+(3&
     &28*5**(3.0/2.0)-160*3**(5.0/2.0))*Fx(6,:  )+(20*3**(5.0/2.0)-41*5&
     &**(3.0/2.0))*Sx(6,:  )+(160*3**(7.0/2.0)-984*5**(3.0/2.0))*F0(6,:&
     &  )-144*sqrt(5)*Fx(7,:  )-864*sqrt(5)*F0(7,:  )-288*sqrt(5)*Fx(8,&
     &:  )+36*sqrt(5)*Sx(8,:  )+864*sqrt(5)*F0(8,:  )))/252.0


Vx(3,:) = -(((-64*3**(3.0/2.0)*sqrt(5))-440)*Fx(1,:  )+((-8*3**(3&
     &.0/2.0)*sqrt(5))-55)*Sx(1,:  )-504*Wx(1,:  )+((-64*3**(5.0/2.0)*s&
     &qrt(5))-1320)*F0(1,:  )+((-32*3**(3.0/2.0)*sqrt(5))-220)*Fx(2,:  &
     &)+(64*3**(5.0/2.0)*sqrt(5)+1320)*F0(2,:  )-992*Fx(3,:  )-124*Sx(3&
     &,:  )-2976*F0(3,:  )-496*Fx(4,:  )+2976*F0(4,:  )+(64*3**(3.0/2.0&
     &)*sqrt(5)-440)*Fx(5,:  )+(8*3**(3.0/2.0)*sqrt(5)-55)*Sx(5,:  )+(6&
     &4*3**(5.0/2.0)*sqrt(5)-1320)*F0(5,:  )+(32*3**(3.0/2.0)*sqrt(5)-2&
     &20)*Fx(6,:  )+(1320-64*3**(5.0/2.0)*sqrt(5))*F0(6,:  )-144*Fx(7,:&
     & )-18*Sx(7,:  )-432*F0(7,:  )-72*Fx(8,:  )+432*F0(8,:  ))/504.0


Vx(4,:) = (((-32*3**(3.0/2.0)*sqrt(5))-220)*Fx(1,:  )+((-64*3**(5&
     &.0/2.0)*sqrt(5))-1320)*F0(1,:  )+((-64*3**(3.0/2.0)*sqrt(5))-440)&
     &*Fx(2,:  )+(8*3**(3.0/2.0)*sqrt(5)+55)*Sx(2,:  )+504*Wx(2,:  )+(6&
     &4*3**(5.0/2.0)*sqrt(5)+1320)*F0(2,:  )-496*Fx(3,:  )-2976*F0(3,: &
     &)-992*Fx(4,:  )+124*Sx(4,:  )+2976*F0(4,:  )+(32*3**(3.0/2.0)*sq&
     &rt(5)-220)*Fx(5,:  )+(64*3**(5.0/2.0)*sqrt(5)-1320)*F0(5,:  )+(64&
     &*3**(3.0/2.0)*sqrt(5)-440)*Fx(6,:  )+(55-8*3**(3.0/2.0)*sqrt(5))*&
     &Sx(6,:  )+(1320-64*3**(5.0/2.0)*sqrt(5))*F0(6,:  )-72*Fx(7,:  )-4&
     &32*F0(7,:  )-144*Fx(8,:  )+18*Sx(8,:  )+432*F0(8,:  ))/504.0


Vx(5,:) = (5**((-3.0)/2.0)*((328*5**(3.0/2.0)+160*3**(5.0/2.0))*F&
     &x(1,:  )+(41*5**(3.0/2.0)+20*3**(5.0/2.0))*Sx(1,:  )+252*5**(3.0/&
     &2.0)*Wx(1,:  )+(984*5**(3.0/2.0)+160*3**(7.0/2.0))*F0(1,:  )+(164&
     &*5**(3.0/2.0)+80*3**(5.0/2.0))*Fx(2,:  )+((-984*5**(3.0/2.0))-160&
     &*3**(7.0/2.0))*F0(2,:  )+(2048*sqrt(5)+1120*3**(3.0/2.0))*Fx(3,: &
     & )+(256*sqrt(5)+140*3**(3.0/2.0))*Sx(3,:  )+(6144*sqrt(5)+1120*3*&
     &*(5.0/2.0))*F0(3,:  )+(1024*sqrt(5)+560*3**(3.0/2.0))*Fx(4,:  )+(&
     &(-6144*sqrt(5))-1120*3**(5.0/2.0))*F0(4,:  )+(328*5**(3.0/2.0)+80&
     &*3**(3.0/2.0))*Fx(5,:  )+(41*5**(3.0/2.0)+10*3**(3.0/2.0))*Sx(5,:&
     &  )+(984*5**(3.0/2.0)+80*3**(5.0/2.0))*F0(5,:  )+(164*5**(3.0/2.0&
     &)+40*3**(3.0/2.0))*Fx(6,:  )+((-984*5**(3.0/2.0))-80*3**(5.0/2.0)&
     &)*F0(6,:  )-288*sqrt(5)*Fx(7,:  )-36*sqrt(5)*Sx(7,:  )-864*sqrt(5&
     &)*F0(7,:  )-144*sqrt(5)*Fx(8,:  )+864*sqrt(5)*F0(8,:  )))/252.0


Vx(6,:) = -(5**((-3.0)/2.0)*((164*5**(3.0/2.0)+80*3**(5.0/2.0))*F&
     &x(1,:  )+(984*5**(3.0/2.0)+160*3**(7.0/2.0))*F0(1,:  )+(328*5**(3&
     &.0/2.0)+160*3**(5.0/2.0))*Fx(2,:  )+((-41*5**(3.0/2.0))-20*3**(5.&
     &0/2.0))*Sx(2,:  )-252*5**(3.0/2.0)*Wx(2,:  )+((-984*5**(3.0/2.0))&
     &-160*3**(7.0/2.0))*F0(2,:  )+(1024*sqrt(5)+560*3**(3.0/2.0))*Fx(3&
     &,:  )+(6144*sqrt(5)+1120*3**(5.0/2.0))*F0(3,:  )+(2048*sqrt(5)+11&
     &20*3**(3.0/2.0))*Fx(4,:  )+((-256*sqrt(5))-140*3**(3.0/2.0))*Sx(4&
     &,:  )+((-6144*sqrt(5))-1120*3**(5.0/2.0))*F0(4,:  )+(164*5**(3.0/&
     &2.0)+40*3**(3.0/2.0))*Fx(5,:  )+(984*5**(3.0/2.0)+80*3**(5.0/2.0)&
```

```
     &)*F0(5,:  )+(328*5**(3.0/2.0)+80*3**(3.0/2.0))*Fx(6,:  )+((-41*5*&
     &*(3.0/2.0))-10*3**(3.0/2.0))*Sx(6,:  )+((-984*5**(3.0/2.0))-80*3*&
     &*(5.0/2.0))*F0(6,:  )-144*sqrt(5)*Fx(7,:  )-864*sqrt(5)*F0(7,:  )&
     &-288*sqrt(5)*Fx(8,:  )+36*sqrt(5)*Sx(8,:  )+864*sqrt(5)*F0(8,:  )&
     &))/252.0


  Vx(7,:) = -((-40*Fx(1,:  ))-5*Sx(1,:  )-18*Wx(1,:  )-120*F0(1,:  &
     &)-20*Fx(2,:  )+120*F0(2,:  )-64*Fx(3,:  )-8*Sx(3,:  )-192*F0(3,:&
     &)-32*Fx(4,:  )+192*F0(4,:  )-40*Fx(5,:  )-5*Sx(5,:  )-120*F0(5,:&
     &)-20*Fx(6,:  )+120*F0(6,:  ))/18.0


  Vx(8,:) = ((-20*Fx(1,:  ))-120*F0(1,:  )-40*Fx(2,:  )+5*Sx(2,:  )&
     &+18*Wx(2,:  )+120*F0(2,:  )-32*Fx(3,:  )-192*F0(3,:  )-64*Fx(4,:&
     &)+8*Sx(4,:  )+192*F0(4,:  )-20*Fx(5,:  )-120*F0(5,:  )-40*Fx(6,:&
     &)+5*Sx(6,:  )+120*F0(6,:  ))/18.0
  !-------------------------------------------------------------
 Case Default
  !
  Print*,"Not yet implemented. Accuracy = ", Accuracy
  Stop
  !
End Select
!---------------------------------------------------------------------

End Subroutine GetUpdatedDOF_Vx
```

Appendix B. Fast linear solver

The structure of matrix (59) allows us to build an upper triangular block wise matrix which for each order of accuracy M only require $M-1$ matrix inversion of matrices of size $m \times m$, where m is the number of variables. We remark that this approach is designed for stiff source terms.

Notice that, Newton procedures update the solution as

$$\begin{aligned} \mathbf{V}^0 &= \mathbf{V}^0 + \delta^0, \\ \mathbf{V}^x &= \mathbf{V}^x + \delta^x, \end{aligned} \tag{B.1}$$

where $\delta^k = [\delta_1^k, \ldots, \delta_N^k]$, with $k = 0, x$, solves

$$(\tilde{\mathbf{I}} - \tilde{\bar{B}}(\mathbf{V}^0))\delta^0 = \mathbf{H}^0(\mathbf{V}^0)\,, \tag{B.2}$$

$$(\tilde{\mathbf{I}} - \tilde{\bar{B}}(\mathbf{V}^0))\delta^x = \mathbf{H}^k(\mathbf{V}^x). \tag{B.2}$$

As both systems have the same structure, therefore, we only show the strategy for obtaining δ_0 from $(\tilde{\mathbf{I}} - \tilde{\mathbf{B}}(\mathbf{V}^0))\delta^0 = \mathbf{H}^0(\mathbf{V}^0)$, for second, third and fourth orders of accuracy.

- System associated with the second order scheme has the following block structure

$$\begin{bmatrix} \mathbf{I} - a_{1,1}\bar{\mathbf{B}}_1 & -a_{1,2}\bar{\mathbf{B}}_2 \\ -a_{2,1}\bar{\mathbf{B}}_1 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_1^0 \\ \mathbf{H}_2^0 \end{bmatrix}, \tag{B.3}$$

with $a_{1,1} = \frac{2}{3}$, $a_{1,2} = -\frac{1}{6}$ and $a_{2,1} = 1$. The solution is found as

$$\begin{aligned} \delta_2^0 &= \mathbf{C2}^{-1} \cdot (\mathbf{H}_2^0 + \bar{\mathbf{B}}_1 \cdot \mathbf{H}_1^0), \\ \delta_1^0 &= \mathbf{H}_1^0 - \frac{2}{3}\mathbf{H}_2^0 - \mathbf{C1} \cdot \delta_2^0, \end{aligned} \tag{B.4}$$

where

$$\begin{aligned} \mathbf{C1} &= \frac{1}{6}\bar{\mathbf{B}}_2 - \frac{2}{3}\mathbf{I}, \\ \mathbf{C2} &= \mathbf{I} + \frac{1}{6}(\bar{\mathbf{B}}_1 \cdot \bar{\mathbf{B}}_2) - \frac{2}{3}\bar{\mathbf{B}}_1. \end{aligned} \tag{B.5}$$

Notice that, we only need to invert one matrix, **C2**.

- System associated with the third order scheme has the following block structure

$$\begin{bmatrix} \mathbf{I} - a_{1,1}\bar{\mathbf{B}}_1 & -a_{1,2}\bar{\mathbf{B}}_2 & -a_{1,3}\bar{\mathbf{B}}_3 \\ -a_{2,1}\bar{\mathbf{B}}_1 & \mathbf{I} - a_{2,2}\bar{\mathbf{B}}_2 & -a_{2,3}\bar{\mathbf{B}}_3 \\ -a_{3,1}\bar{\mathbf{B}}_1 & -a_{3,2}\bar{\mathbf{B}}_2 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_1^0 \\ \mathbf{H}_2^0 \\ \mathbf{H}_3^0 \end{bmatrix}, \tag{B.6}$$

with

$$\begin{aligned} a_{1,1} &= \left(\frac{3}{10} - \frac{1}{20\sqrt{3}}\right),\ a_{1,2} = \left(\tfrac{1}{5} - \tfrac{11}{20\sqrt{3}}\right),\ a_{1,3} = \frac{1}{10\sqrt{3}}, \\ a_{2,1} &= \left(\frac{1}{5} + \frac{11}{20\sqrt{3}}\right),\ a_{2,2} = \left(\tfrac{3}{10} + \tfrac{1}{20\sqrt{3}}\right),\ a_{2,3} = -\frac{1}{10\sqrt{3.0}}, \\ a_{3,1} &= \frac{1}{2}, \qquad a_{3,2} = \tfrac{1}{2}, \end{aligned} \tag{B.7}$$

the solution is found as

$$\begin{aligned} \delta_3 &= \mathbf{E33}^{-1} \cdot (\mathbf{R}_3^3 - \mathbf{D32} \cdot \mathbf{R}_3^2))\,, \\ \delta_2 &= -\mathbf{E23} \cdot \delta_3 + \mathbf{R}_4^2, \\ \delta_1 &= -\mathbf{C12} \cdot \delta_2 - \mathbf{C13} \cdot \delta_3 + \mathbf{R}_4^1, \end{aligned} \tag{B.8}$$

where

$$\begin{aligned} \mathbf{C12} &= \frac{a_{2,2}a_{1,1} - a_{2,1}a_{1,2}}{a_{2,1}}\bar{\mathbf{B}}_2, \\ \mathbf{C13} &= \frac{a_{2,3}a_{1,1} - a_{2,1}a_{1,3}}{a_{2,1}}\bar{\mathbf{B}}_3, \\ \mathbf{C21} &= -a_{2,1}\bar{\mathbf{B}}_1, \\ \mathbf{C22} &= \mathbf{I} - a_{2,2}\bar{\mathbf{B}}_2, \\ \mathbf{C23} &= -a_{2,3}\bar{\mathbf{B}}_3, \\ \mathbf{C31} &= -a_{3,1}\bar{\mathbf{B}}_1, \\ \mathbf{C32} &= -a_{3,2}\bar{\mathbf{B}}_2, \\ \mathbf{D22} &= \mathbf{C22} - \mathbf{C21} \cdot \mathbf{C12}, \\ \mathbf{D23} &= \mathbf{C23} - \mathbf{C21} \cdot \mathbf{C13}, \\ \mathbf{D32} &= \mathbf{C32} - \mathbf{C31} \cdot \mathbf{C12}, \\ \mathbf{D33} &= \mathbf{I} - \mathbf{C31} \cdot \mathbf{C13}, \end{aligned}$$

$$\begin{aligned}
\mathbf{E23} &= \mathbf{D22}^{-1}\cdot\mathbf{D23},\\
\mathbf{E33} &= \mathbf{D33}-\mathbf{D32}\cdot\mathbf{E23},
\end{aligned} \tag{B.9}$$

and

$$\begin{aligned}
\mathbf{R}_4^1 &= \mathbf{H}_1^0-\frac{a_{1,1}}{a_{2,1}}\mathbf{H}_2^0,\\
\mathbf{R}_2^2 &= \mathbf{H}_2^0-\mathbf{C21}\cdot\mathbf{R}_4^1,\\
\mathbf{R}_3^2 &= \mathbf{D22}^{-1}\cdot\mathbf{R}_2^2,\\
\mathbf{R}_4^2 &= \mathbf{D22}^{-1}\cdot\mathbf{R}_2^2,\\
\mathbf{R}_3^3 &= \mathbf{H}_3^0-\mathbf{C31}\cdot\mathbf{R}_4^1.
\end{aligned} \tag{B.10}$$

Notice that this strategy involve the inversion of matrices **D22** and **E33**.

- System associated with the fourth order scheme has the following block structure

$$\begin{bmatrix}
\mathbf{I}-a_{1,1}\tilde{\mathbf{B}}_1 & -a_{1,2}\tilde{\mathbf{B}}_2 & -a_{1,3}\tilde{\mathbf{B}}_3 & -a_{1,4}\tilde{\mathbf{B}}_4\\
-a_{2,1}\tilde{\mathbf{B}}_1 & \mathbf{I}-a_{2,2}\tilde{\mathbf{B}}_2 & -a_{2,3}\tilde{\mathbf{B}}_3 & -a_{2,4}\tilde{\mathbf{B}}_4\\
-a_{3,1}\tilde{\mathbf{B}}_1 & -a_{3,2}\tilde{\mathbf{B}}_2 & \mathbf{I}-a_{3,3}\tilde{\mathbf{B}}_3 & -a_{3,4}\tilde{\mathbf{B}}_4\\
-a_{4,1}\tilde{\mathbf{B}}_1 & -a_{4,2}\tilde{\mathbf{B}}_2 & -a_{4,3}\tilde{\mathbf{B}}_3 & \mathbf{I}
\end{bmatrix}
\begin{bmatrix}\delta_1\\ \delta_2\\ \delta_3\\ \delta_4\end{bmatrix}
= \begin{bmatrix}\mathbf{H}_1^0\\ \mathbf{H}_2^0\\ \mathbf{H}_3^0\\ \mathbf{H}_4^0\end{bmatrix}, \tag{B.11}$$

with

$$\begin{aligned}
a_{1,1} &= \frac{41}{252}-\frac{1}{14\sqrt{15}},\\
a_{1,2} &= \frac{64}{315}\frac{-1}{\sqrt{15}},\\
a_{1,3} &= \frac{41}{252}-\sqrt{37}\sqrt{5},\\
a_{1,4} &= -\frac{1}{35},\\
a_{2,1} &= \frac{\sqrt{5}}{7\sqrt{3}}+\frac{55}{504},\\
a_{2,2} &= \frac{31}{126},\\
a_{2,3} &= \frac{55}{504}-\sqrt{57}\sqrt{3},\\
a_{2,4} &= \frac{1}{28},\\
a_{3,1} &= \frac{\sqrt{3}}{7\sqrt{5}}+\frac{41}{252},\\
a_{3,2} &= \frac{1}{\sqrt{15}}+\frac{64}{315},\\
a_{3,3} &= \frac{1}{14\sqrt{15}}+\frac{41}{252},\\
a_{3,4} &= -\frac{1}{35},\\
a_{4,1} &= \frac{5}{18},\\
a_{4,2} &= \frac{4}{9},\\
a_{4,3} &= \frac{5}{18},
\end{aligned} \tag{B.12}$$

in this case the solution to this linear system is given by

$$\begin{aligned}
\delta_4 &= \mathbf{H44}^{-1}\cdot\mathbf{R}_4^3,\\
\delta_3 &= \mathbf{R}_3^3-\mathbf{H34}\cdot\delta_4,\\
\delta_2 &= \mathbf{R}_2^2-\mathbf{G23}\cdot\delta_3-\mathbf{G24}\cdot\delta_4,\\
\delta_1 &= \mathbf{R}_1^1-\mathbf{D12}\cdot\delta_2-\mathbf{D13}\cdot\delta_3-\mathbf{D14}\cdot\delta_4,
\end{aligned} \tag{B.13}$$

where matrices are found by the following sequence of computations

$$\begin{aligned}
\mathbf{D12} &= (a_{1,1}a_{2,2}-a_{2,1}a_{1,2})/a_{2,1}\tilde{\mathbf{B}}_2-a_{1,1}/a_{2,1}\mathbf{I},\\
\mathbf{D13} &= (a_{1,1}a_{2,3}-a_{2,1}a_{1,3})/a_{2,1}\tilde{\mathbf{B}}_3,\\
\mathbf{D14} &= (a_{1,1}a_{2,4}-a_{2,1}a_{1,4})/a_{2,1}\tilde{\mathbf{B}}_4,\\
\mathbf{R}_1^1 &= \mathbf{H}_1^0-a_{1,1}/a_{2,1}\mathbf{H}_2^0,\\
\mathbf{R}_2^1 &= \mathbf{H}_2^0+a_{2,1}\tilde{\mathbf{B}}_1\cdot\mathbf{R}_1^1,\\
\mathbf{R}_3^1 &= \mathbf{H}_3^0+a_{3,1}\tilde{\mathbf{B}}_1\cdot\mathbf{R}_1^1,\\
\mathbf{R}_4^1 &= \mathbf{H}_4^0+a_{4,1}\tilde{\mathbf{B}}_1\cdot\mathbf{R}_1^1,\\
\mathbf{E22} &= \mathbf{I}-a_{2,2}\tilde{\mathbf{B}}_2+a_{2,1}\tilde{\mathbf{B}}_1\cdot\mathbf{D12},\\
\mathbf{E33} &= \mathbf{I}-a_{3,3}\tilde{\mathbf{B}}_3+a_{3,1}\tilde{\mathbf{B}}_1\cdot\mathbf{D13},\\
\mathbf{E44} &= \mathbf{I}+a_{4,1}\tilde{\mathbf{B}}_1\cdot\mathbf{D14},\\
\mathbf{E32} &= -a_{3,2}\tilde{\mathbf{B}}_2+a_{3,1}\tilde{\mathbf{B}}_1\cdot\mathbf{D12},\\
\mathbf{E42} &= -a_{4,2}\tilde{\mathbf{B}}_2+a_{4,1}\tilde{\mathbf{B}}_1\cdot\mathbf{D12},\\
\mathbf{E23} &= -a_{2,3}\tilde{\mathbf{B}}_3+a_{2,1}\tilde{\mathbf{B}}_1\cdot\mathbf{D13},\\
\mathbf{E43} &= -a_{4,3}\tilde{\mathbf{B}}_3+a_{4,1}\tilde{\mathbf{B}}_1\cdot\mathbf{D13},\\
\mathbf{E24} &= -a_{2,4}\tilde{\mathbf{B}}_4+a_{2,1}\tilde{\mathbf{B}}_1\cdot\mathbf{D14},\\
\mathbf{E34} &= -a_{3,4}\tilde{\mathbf{B}}_4+a_{3,1}\tilde{\mathbf{B}}_1\cdot\mathbf{D14},\\
\mathbf{G23} &= \mathbf{E22}^{-1}\cdot\mathbf{E23},\\
\mathbf{G24} &= \mathbf{E22}^{-1}\cdot\mathbf{E24},\\
\mathbf{G33} &= \mathbf{E33}-\mathbf{E32}\cdot\mathbf{G23},\\
\mathbf{G43} &= \mathbf{E43}-\mathbf{E42}\cdot\mathbf{G23},\\
\mathbf{G34} &= \mathbf{E34}-\mathbf{E32}\cdot\mathbf{G24},\\
\mathbf{G44} &= \mathbf{E44}-\mathbf{E42}\cdot\mathbf{G24},\\
\mathbf{R}_2^2 &= \mathbf{E22}^{-1}\cdot\mathbf{R}_2^1,\\
\mathbf{R}_3^2 &= \mathbf{R}_3^1-\mathbf{E32}\cdot\mathbf{R}_2^2,\\
\mathbf{R}_4^2 &= \mathbf{R}_4^1-\mathbf{E42}\cdot\mathbf{R}_2^2,\\
\mathbf{H34} &= \mathbf{G33}^{-1}\cdot\mathbf{G34},\\
\mathbf{H44} &= \mathbf{G44}-\mathbf{G43}\cdot\mathbf{H34},\\
\mathbf{R}_3^3 &= \mathbf{G33}^{-1}\cdot\mathbf{R}_3^2,\\
\mathbf{R}_4^3 &= \mathbf{R}_4^2-\mathbf{G43}\cdot\mathbf{R}_3^3.
\end{aligned} \tag{B.14}$$

If optimized libraries for solving linear system are implemented (as those reported in Fortran recipes chapter 2, in [25]). The matrix inversions $\mathbf{C2}^{-1}\cdot(\mathbf{H}_2^0+\tilde{\mathbf{B}}_1\cdot\mathbf{H}_1^0)$, $\mathbf{E33}^{-1}\cdot(\mathbf{R}_3^3-\mathbf{D32}\cdot\mathbf{R}_3^2))$ and $\mathbf{H44}^{-1}\cdot\mathbf{R}_4^3$ for the second, third and fourth order respectively, are not required. Instead, the optimized solution of a liner system is implemented, which is cheaper than the matrix inversion and the subsequent matrix-vector multiplication.

References

[1] Balsara DS, Meyer C, Dumbser M, Du H, Xu Z. Efficient implementation of ADER schemes for Euler and magneto hydro dynamical flows on structured meshes - speed comparisons with Runge–Kutta methods. J Comput Phys 2013;235(0):934–69.

[2] Balsara DS, Rumpf T, Dumbser M, Munz CD. Efficient, high accuracy ADER-WENO schemes for hydrodynamics and divergence-free magnetohydrodynamics. J Comput Phys 2009;228(7):2480–516.

[3] Balsara DS, Taflove A, Garain S, Montecinos G. Computational electrodynamics in material media with constraint-preservation, multidimensional Riemann solvers and sub-cell resolution part i, second-order FVTD schemes. J Comput Phys 2017.

[4] Castro CE, Behrens J, Pelties C. CUDA-C implementation of the ADER-DG method for linear hyperbolic PDEs. Geosci Model Dev 2013;6:3743–86.

[5] Castro CE, Toro EF. Solvers for the high–order riemann problem for hyperbolic balance laws. J Comput Phys 2008;227:2481–513.

[6] 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.

[7] Dumbser M, Castro MJ, Parés C, Toro EF. ADER schemes on unstructured meshes for nonconservative hyperbolic systems: applications to geophysical flows. Comput Fluids 2009;38(9):1731–48.
[8] 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.
[9] Dumbser M, Hidalgo A, Castro M, Parés C, Toro EF. FORCE schemes on unstructured meshes II: non-conservative hyperbolic systems. Comput Methods Appl Mech Eng 2010;199(9–12):625–47.
[10] 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.
[11] Dumbser M, Käser M, De La Puente J. Arbitrary high-order finite volume schemes for seismic wave propagation on unstructured meshes in 2d and 3d. Geophys J Int 2007;171(2):665–94.
[12] Dumbser M, Käser M, Titarev VA, Toro EF. Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems. J Comput Phys 2007;226(8):204–43.
[13] 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.
[14] Dumbser M, Zanotti O, Hidalgo A, Balsara DS. ADER-WENO finite volume schemes with space-time adaptive mesh refinement. Commun Comput Phys 2013;248:257–86.
[15] 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.
[16] Harten A, Engquist B, Osher S, Chakravarthy SR. Some results on high-order accurate essentially non-oscillatory schemes. Appl Numer Math 1986;2:347–77.
[17] Harten A, Osher S. Uniformly high-order accurate nonoscillatory schemes. i. SIAM J Numer Anal 1987;24(2):279–309.
[18] Kowalevski S. Zur theorie der partiellen differentialgleichung. 1875. 80:1–32.
[19] Lax PD. Hyperbolic systems of conservation laws and the mathematical theory of shock waves. CBMS-NSF regional conference series in applied mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104); 1973.
[20] LeVeque RJ, Yee HC. A study of numerical methods for hyperbolic conservation laws with stiff source terms. J Comput Phys 1990;86:187–210.
[21] 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, vol. 141. Birkhuser Basel; 2001. p. 723–32.
[22] 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.
[23] Montecinos G.I. Analytic solutions for the burgers equation with source terms. 2015.
[24] Montecinos GI, Müller LO, Toro EF. Hyperbolic reformulation of a 1D viscoelastic blood flow model and ADER finite volume schemes. Preprint NI14014-NPA, Isaac Newton Institute for Mathematical Sciences; 2014.
[25] Press WH, Teukolsky SA, Vetterling WT, Flannery BP, Metcalf M. Numerical recipes in Fortran 77. 2. Cambridge University Press; 1992. (September 25, 1992).
[26] Shu CW, Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. J Comput Phys 1988;77(2):439–71.
[27] Toro EF. Anomalies of conservative methods: analysis and numerical evidence. Int J Comut Fluid Dyn 2002;11(2):128–43.
[28] Toro EF, Millington RC, Nejad LAM. Towards very high–order Godunov schemes. In: Toro EF, editor. Godunov methods: theory and applications. Kluwer Academic/Plenum Publishers; 2001. p. 905–37.
[29] Toro EF, Titarev VA. Solution of the generalised Riemann problem for advection–reaction equations. Proc R Soc Lond A 2002;458:271–81.