Noname manuscript No.
(will be inserted by the editor)

Mathematics and Numerics for Balance Partial Differential-Algebraic Equations (PDAEs)

Wanderson Lambert · Amaury Alvarez · Ismael Ledoino · Duilio Conceição · Dan Marchesin · Johannes Bruining

Received: date / Accepted: date

Abstract We study systems of partial differential-algebraic equations (PDAEs) of first order. Classical solutions of the theory of Hyperbolic Partial Differential equation such as discontinuities (shock and contact discontinuities), rarefactions and diffusive traveling waves are extended for variables living on a surface $\mathcal{S}$, which is defined as solution of a set of algebraic equations. We propose here an alternative formulation to study numerically and theoretically the PDAEs by changing the algebraic conditions into partial differential equations with relaxation source terms (PDREs). The solution of such relaxed systems is proved to tend to the surface $\mathcal{S}$, i.e., to satisfy the algebraic equations for long times. We formulate a unified numerical scheme for systems of PDAEs and PDREs. This scheme is naturally parallelizable and has faster convergence. Evidence of its effectiveness is presented by means of simulations for physical and synthetic problems.

W. Lambert
Alfenas Federal University - ICT/MG - Rod. BR-267, km 533, Brazil
E-mail: wanderson.lambert@unifal-mg.edu.br

A. Alvarez
Departamento de Ciência da Computacão, Universidade Federal do Rio de Janeiro, C. P.
E-mail: amaury@dcc.ufrj.br

I. S. Ledoino
Laboratório Nacional de Computação Científica, Av. G. Vargas, 333, Petrópolis, RJ, Brazil
E-mail: ismael.sledoino@gmail.com

D. T. Conceição
Departamento de Matemática, UFRRJ, BR-465, km 7, 23897-000, Seropédica, RJ, Brazil
E-mail: duiliotadeu@gmail.com

D. Marchesin
IMPA, Estrada Dona Castorina 110, 22460-320 Rio de Janeiro, RJ, Brazil
E-mail: marchesi@fluid.impa.br

J. Bruining
TU Delft, Civil Engineering and Geosciences, Stevinweg 1, E-mail: J.Bruining@tudelft.nl

Keywords Partial Differential-Algebraic Equations (PDAEs) · Hyperbolic system of equations · Riemann problems · parallelizable numerical schemes
PACS 47.11.Df · 47.40.Nm · 47.56.+r
Mathematics Subject Classification (2000) 76S05 · 76M10 · 76M20

1 Introduction

In many applications, systems of partial differential equations should be supplemented with algebraic restrictions, which represent laws in physics, chemistry, biology or in any other field of application of these equations, see [15,25, 38,48,49] and references therein. These equations are called Partial Differential-Algebraic Equations (PDAEs). In this work, we study systems of PDAEs given by

$$\partial_t G(U) + \partial_x F(U) = 0, \tag{1}$$
$$H(U) = 0. \tag{2}$$

Here the variables $U = (U_1, U_2, \cdots, U_n) : \mathbb{Q} \subset \mathbb{R} \times \mathbb{R}^+ \longrightarrow \mathcal{V} \subset \mathbb{R}^n$ are the coordinates of *phase space* $\mathcal{V}$; the accumulation and flux functions are $G = (G_1(U), \cdots, G_m(U))^T : \mathcal{V} \subset \mathbb{R}^n \longrightarrow \mathbb{R}^m$, $F = (F_1(U), \cdots, F_m(U))^T : \mathcal{V} \subset \mathbb{R}^n \longrightarrow \mathbb{R}^m$; the algebraic relationships are represented by the vector function $H = (H_1(U), H_2(U), \cdots, H_p(U))^T : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}^p$. Eq. (1) is a system of partial differential equations and Eq. (2) is a system of algebraic equations. We assume that the functions G, F, H are sufficiently smooth.

In the first part of this work, we obtain theoretical results about the partial differential-algebraic equations of first order (1), (2). The solution for such PDAEs for certain models are obtained from index theory, see [7,8,12,18,34, 35] and references therein. In this theory, when the system presents algebraic equations (or algebraic boundary conditions) these algebraic conditions are differentiated to obtain the index of the PDAE, see [7,8,12,18,34,35]. In our model, we apply a similar idea to obtain the eigenpairs (eigenvalues and eigenvectors) associated to PDAEs (1), (2). The eigenpairs are derived by looking for solutions that are self-similar in the variable $\xi = x/t$, through a derivation that leads to a system of differential-algebraic equations (DAE) in the variable ξ. Using index theory, we verify under certain hypothesis that the index of the resulting system for the calculation of eigenpairs is 1. In this way, we extend to PDAEs some classical results for solving Riemann problem (RP) using shocks, contact discontinuities and rarefactions waves for the system of hyperbolic equations. This approach is similar to the one used for classical systems of hyperbolic equations, see [13,40].

The RP is defined as the solutions of PDAEs (1), (2) with piecewise constant initial data

$$\begin{cases} U_L \text{ if } x < 0, \\ U_R \text{ if } x > 0. \end{cases} \tag{3}$$

Of course we will assume that for each U_L and U_R Eq. (2) is satisfied.

For these PDAEs, we define the main waves present in Riemann solutions (shocks, contact discontinuities and rarefactions). Moreover, we obtain geometrical loci or bifurcation structures, which are fundamental to construct the solution of RPs. Therefore, our study is concerned with a generalization of the standard theory for hyperbolic conservation laws, see for instance [13,40]. Here, we construct the general formalism to solve Riemann problems. Efforts to obtain analytical Riemann solutions are always of interest, since they are used to verify numerical procedures.

We recall that Glimm's method for system of conservation laws is based on interactions estimated between shock, rarefactions. An extension of such method for PDAEs (1) is presented in [39].

In our work, to study the PDAEs (1), (2) with initial condition (3), we have two strategies depending on the availability of a global explicit parameterization for the surface $\mathcal{S}$. In the literature, such parameterization is commonly available. Then it is possible to express a certain group of variables in terms of another one. We denote the first group as $V = (U_1, \cdots, U_m)$, and the second group as $W = (U_{l+1}, \cdots, U_n)$ i.e. $U = (V, W)$. Applying the Implicit Function Theorem to $H(U) = 0$ there exists a diffeomorphism W such that $H(V, W(V)) = 0$ and

$$\partial_V H + (\partial_W H)\, \partial_V W = 0 \quad \longrightarrow \quad \partial_V W = -\left(\partial_W H\right)^{-1} \partial_V H. \tag{4}$$

Finally, the PDAE system (1), (2) is rewritten as:

$$\partial_t \tilde{G}(V) + \partial_x \tilde{F}(V) = 0, \tag{5}$$

where $\tilde{G}(V) = G(V, W(V))$, $\tilde{F}(V) = F(V, W(V))$. The theory of this case is classical. Examples of $\mathcal{S}$ are found in [1,3,27,28]. In Appendix A, we draw some results about the class of equations (5).

In the other case, the explicit parameterization is not available. We still assume that the algebraic equations (2) define implicitly a regular manifold in phase space $\mathcal{V}$, which we also denote by $\mathcal{S}$. By using the Implicit Function Theorem we derive the main ingredients in the construction of the RP solutions of PDAEs, which are shocks and contact discontinuities, rarefaction and traveling waves.

Instead of Equation (2), Equation (1) can be supplemented with

$$\partial_t H(U) = 0, \tag{6}$$

with initial conditions (3), for U_L and U_R that satisfying Equation (2); or with the relaxation equation

$$\partial_t H(U) = -\frac{1}{\tau} H(U) \quad \text{with} \quad \tau > 0. \tag{7}$$

In Section 3 we study the former case and in Section 4 we study the latter one. For the problems in Sections 3 or 4 with initial data close to $\mathcal{S}$, we expect the solutions to be also close to $\mathcal{S}$. In Section 4, we verify that when either t tends to infinity or τ tends to zero the solution tends to the surface $\mathcal{S}$. Indeed,

as we show in Proposition 6, the equation (7) is proposed in such a way that the surface $\mathcal{S}$ is an attractor. In this manner, we are "relaxing" the system of equations that models phenomena described by the PDAE (1), (2). The system composed of (1), (7) is called partial differential "relaxed" equation (PDRE).

We are interested in analyzing numerical methods for PDAEs and PDREs. There are many works dealing with numerical modelling of PDAEs. In the works [15, 48–50], the authors cite three strategies used to solve geochemical transport in groundwater. There is a different approach in the works [1, 3, 27, 28].

One contribution of this work is obtaining the numerical solution for PDAEs (1), (2) by means of PDREs (1), (7). We present a fast, robust and parallelizable finite difference solver for a class of equations more general than (1), (2). We call it the *Reaction-Convection-Diffusion equations Solver* (RCD), which is able to approximate the solution to partial differential reaction-convection-diffusion equations:

$$\partial_t G(U) + \partial_x F(U) = \partial_x \left[\mathcal{B}(U)\partial_x U\right] + R(U), \tag{8}$$

$$\tau \partial_t H(U) = -H(U), \qquad \text{with initial conditions } U(x,0) = U_0(x), \tag{9}$$

where the new terms involve the diffusion matrix $\mathcal{B} : \mathcal{V} \subset \mathbb{R}^n \longrightarrow \mathbb{R}^{m\times n}$, and the reaction terms $R : \mathcal{V} \subset \mathbb{R}^n \longrightarrow \mathbb{R}^m$. Notice that (8), (9) extends (1), (2) if we set $\mathcal{B} = 0$, $R = 0$ in Eq. (8) and $\tau = 0$ in Eq. (9). If we consider $\tau \neq 0$ we obtain Eq. (7). Numerically, modeling using relaxation introduces a regularization in the solution, this behavior appears for example in [23].

We summarize each section of this work as follows. In Section 2, we describe the main definitions for solutions of PDAE and we prove equivalence of these definitions. We also study the formalism for shocks, rarefactions and traveling waves.

In Section 3, we study the relationship between systems of PDAEs (1), (2), with systems of PDEs of form (1), (6) and we prove the equivalence between the solutions for suitable initial conditions. In Section 4, we study the alternative relaxation model (1), (7). In Section 5, we introduce the RCD Solver for system (8), (9). In Section 5.4.1, we perform several numerical experiments for system (8), (9). Conclusions and comments are given in Section 6.

To complete our paper, in Appendix A, we draw results for classical hyperbolic systems of equations. In Appendix B, we discuss entropy-entropy pairs for systems (1), (6) and for systems of PDAEs (1), (2). In Appendix C, we define linear PDAEs and finite difference schemes such as *upwind, Lax-Friedrichs, Crank-Nicolson and Lax-Wendroff*. Moreover, in Appendix $C.2$, we shows that the Lax-Richtmyer theorem for *PDEs* extends for linear PDAEs. In Appendix D, we extend these classical schemes for general non-linear PDAEs.

2 Partial Differential-Algebraic Equations

Systems of hyperbolic conservation laws, generically, exhibit discontinuities in the solutions, see [13,41,42]. In this way, it is necessary to define a weak solution for the system of PDAEs (1), (2). The weak solution is defined by an integral form in the presence of discontinuities. Since only (1) possesses derivatives, we define the integral form only for this equation and we assume that Eq. (2) is satisfied pointwise for all states:

Definition 1 We say that $U(x,t) \in (L^{\infty}(\mathbb{Q}))^n$, for $\mathbb{Q} \subset \mathbb{R} \times \mathbb{R}^+$, is a weak solution of PDAEs (1), (2), if and only if, Eq. (1) is satisfied in the classical distribution sense and Eq. (2) is satisfied pointwise.

In addition, to define entropy solution, we will present the traveling wave criterion to select the physical shocks in Section 2.1.3. We also prove, in Section 3, the equivalence of PDAE and a particular class of hyperbolic systems subject to appropriate initial conditions.

Below, we present analysis for the *implicit formulation*. In Appendix A, we present results for the case for which a parameterization for the surface $\mathcal{S}$ is available.

2.1 The implicit formalism

Since we are interested in the Riemann problem for the implicit formulation we analyse shock and rarefaction waves, see [13,41,42]. We also extend the criterion of traveling waves to select the physical shocks. The Riemann problem for system (1), (2) includes the initial conditions of form (3). For this class of problems, we explore the self similarity in the variable $\xi = x/t$ to obtain shock and rarefaction waves. We show the equivalence between the above mentioned waves in the implicit and explicit formalisms. Finally, we obtain traveling waves. An extension of Lax's entropy criterion is useful for PDAEs, see [39].

2.1.1 Characteristic speeds

We assume that $U(x,t) := \hat{U}(\xi)$ is a sufficiently smooth function of the variable $\xi = x/t$. Using the chain rule in the new variable $\hat{U}$

$$\begin{aligned} \partial_t U(x,t) &= \left(d_\xi \hat{U}(\xi)\right) \partial_t \xi = \left(d_\xi \hat{U}(\xi)\right) (-\xi/t)\,, \\ \partial_x U(x,t) &= \left(d_\xi \hat{U}(\xi)\right) \partial_x \xi = \left(d_\xi \hat{U}(\xi)\right) (1/t)\,. \end{aligned} \tag{10}$$

We differentiate (1) with respect the variables x and t and using (10) we obtain:

$$B\partial_t \hat{U} + A\partial_x \hat{U} = 0 \quad \longrightarrow \quad -B\xi d_\xi \hat{U} + A d_\xi \hat{U} = 0; \tag{11}$$

where B and A are the $m \times n$ jacobian matrices of $G(U)$, $F(U)$ in Eqs (1)-(2) with respect to the variable U. Using the fact that the algebraic equation

(2) is an identity for curves parameterized by ξ, we differentiate the algebraic equation (2), obtaining:

$$d_\xi H(\hat{U}(\xi)) = 0, \quad \longrightarrow \quad E d_\xi \hat{U} = 0, \quad . \tag{12}$$

where E is the jacobian matrix of H with respect to U. From Eqs. (11) and (12), we find:

$$M\boldsymbol{r} = 0, \quad \text{where} \quad M(\xi) \equiv \begin{pmatrix} -\xi B + A \\ E \end{pmatrix} \quad \text{and} \quad d_\xi \hat{U} = \boldsymbol{r}. \tag{13}$$

In (13) the orientation of the vector $\boldsymbol{r}$ at each point must be chosen properly.

The solution of the first equation in (13) defines the characteristic field, which exists provided we impose that $\xi = \lambda$ in Eq. (13), namely:

$$det(M(\lambda)) = \det \begin{pmatrix} -\lambda B + A \\ E \end{pmatrix} = 0. \tag{14}$$

The characteristic equation (14) involves a polynomial of degree m or less in the unknown λ. Each root λ is an *eigenvalue* and the corresponding $\boldsymbol{r}$ is the right *eigenvector*, for a *generalized eigenvalue problem*. Similarly we define the *left eigenvector* l for the eigenvalue λ as:

$$\boldsymbol{l}^T M = 0 \quad \text{or} \quad \boldsymbol{l}^T \begin{pmatrix} -\lambda B + A \\ E \end{pmatrix} = 0. \tag{15}$$

By integrating Eq. (13) supplemented with initial conditions $U(0) = (V, W)_0$ we obtain *integral curves*.

In the Proposition that follows, we prove the connection between the characteristic fields in the implicit and the explicit formalisms. To do so, let us consider the generalized eigenvalue problem with matrices $\tilde{A}$ and $\tilde{B}$ representing the jacobian of $\tilde{F}$ and $\tilde{G}$, i.e., the flux and the accumulation functions F and G restricted to surface $\mathcal{S}$ as:

$$\tilde{A}\tilde{r} = \lambda \tilde{B}\tilde{r} \quad \longrightarrow \quad det(\tilde{A} - \lambda \tilde{B}) = 0. \tag{16}$$

Proposition 1 *Assume that the system* (1), (2) *is formally hyperbolic. Then* (*i*) *the characteristic polynomials* (14) *and* (16) *are equal. Moreover,* (*ii*) *each eigenvector solving* (16) *or* (13) *generates the same field on the* m*-dimensional tangent plane* $T_{U^*}\mathcal{S}$ *for each* $U^* \in \mathcal{S}$.

Proof: By assuming that $W = W(V)$, we write $\tilde{B}$, $\tilde{A}$, of Equation (16), as:

$$\tilde{B} = \partial_V G + (\partial_W G)\, \partial_V W \qquad \text{and} \qquad \tilde{A} = \partial_V F + (\partial_W F)\, \partial_V W, \tag{17}$$

as in (4). Then $\tilde{M} \equiv \tilde{A} - \lambda \tilde{B}$ is written as:

$$\tilde{M} = \partial_V F - \lambda \partial_V G + (\partial_W F - \lambda \partial_W G)\, \partial_V W. \tag{18}$$

The characteristic polynomial is obtained from $det(\tilde{M}) = 0$.

To calculate (14) for M given by (13), notice that for the variable $U = (V, W)$ (rearranging if necessary) we write the matrix M as:

$$M = \begin{pmatrix} \partial_V \mathcal{M} & \partial_W \mathcal{M} \\ \partial_V H & \partial_W H \end{pmatrix}, \quad \text{where} \quad \mathcal{M} = F - \lambda G, \quad \lambda \text{ is constant.} \tag{19}$$

We know that curve $W(\xi) = W(V(\xi))$ satisfies $H(V, W(V)) = 0$, which we differentiate to obtain $(4.a)$. To prove (i), we premultiply the matrix M by

$$\begin{pmatrix} \mathbb{I}_{m\times m} & \mathbb{I}_{m\times p} \\ \partial_V W & \mathbb{I}_{p\times p} \end{pmatrix}, \tag{20}$$

here $\mathbb{I}_{m\times p}$ is the identity matrix with m rows and p columns. We obtain:

$$\begin{pmatrix} \tilde{M} & \partial_W \mathcal{M} \\ \partial_V H + (\partial_W H)\partial_V W & \partial_W H \end{pmatrix}, \quad \longrightarrow \quad \begin{pmatrix} \tilde{M} & \partial_W \mathcal{M} \\ 0 & \partial_W H \end{pmatrix}, \tag{21}$$

with $\tilde{M}$ given in Eq. (18). The last matrix in Eq. (21) is obtained using $(4.a)$. From (21), we obtain $\det(M) = \det(\tilde{M}) \det(\partial_W H)$. Notice that $\det(\partial_W H) \neq 0$, thus $det(M) = 0$, if only if $det(\tilde{M}) = 0$, thus we obtain thus the characteristic polynomial for both approaches are the same. Using similar calculations, we prove (ii), i.e., that the eigenvectors of both approaches generate the same field on the tangent space $T_{U^*}\mathcal{S}$ for each each $U^* \in \mathcal{S}$, besides the eigenvectors generate, locally, $\mathcal{S}$. The proof is complete. $\square$.

We say that the system is formally hyperbolic if it satisfies:

Definition 2 We say that PDAE (1), (2) is *formally hyperbolic* if the characteristic polynomial admits m real eigenvalues (accounting for multiplicities)

$$\lambda_1(U) \leq \lambda_2(U) \leq \cdots \leq \lambda_m(U) \tag{22}$$

and m independent eigenvectors $\boldsymbol{r}_i$ (for $i = 1, \cdots, m$) satisfying $(13.a)$. We say that the system is *formally strictly hyperbolic* if the inequalities are strict.

Notice that we have m integral curves for the m eigenpairs $(\lambda, \boldsymbol{r})$.

Remark 1 In A, we discuss this definition when we have a parameterization of surface $\mathcal{S}$, notice that the definition of hyperbolicity is the same the one in the classical case. Notice, however, that for any state $U^* \in \mathcal{S}$ satisfying $H(U^*) = 0$, one can prove that the system is formally hyperbolic if, and only if, the set $\boldsymbol{r}(U^*)$ (eigenvectors evaluated in U^*) forms a basis for the tangent plane $T_{U^*}\mathcal{S}$ for $U^* \in \mathcal{S}$.

The integral curve associated to $(\lambda_i, \boldsymbol{r}_i)$ is called i-integral curve. The i-rarefaction is the part of integral curve for which λ_i is an increasing function, i.e,

$$\nabla \lambda_i \cdot \boldsymbol{r}_i > 0. \tag{23}$$

Using Propositions 1 (or (13 in A), we see that integral curves for each field $(\lambda_k, \mathbf{r}_k)$ generates the same curves using approaches described in Section

2.1 (or A). From each integral curve on the field $(\lambda_k, \mathbf{r}_k)$, we obtain the rarefaction satisfying (23). The next Proposition gives an useful result relating the inequality (23) and the second derivatives of accumulation and flux functions, see [47] for more general applications in the numerical construction of rarefaction curves.

Proposition 2 *Let $\boldsymbol{r}_k, \boldsymbol{l}_k^T$ associated to the eigenvalue λ_k correspond to eigenpairs obtained by Eq.* (14)*. The identity*

$$\nabla\lambda_k \cdot r_k \left(\boldsymbol{l}_k^T \begin{pmatrix} B \\ 0 \end{pmatrix} \boldsymbol{r}_k \right) = \boldsymbol{l}_k^T \begin{pmatrix} -\lambda_k \partial_U^2 G + \partial_U^2 F \\ \partial_U^2 H \end{pmatrix} (\boldsymbol{r}_k, \boldsymbol{r}_k). \tag{24}$$

is satisfied.

Proof: *For fixed $\lambda_k(U(\xi))$ (here we substitute ξ by λ), we differentiate $M\boldsymbol{r}_k = 0$, given in Eq.* (13)*, and we have:*

$$\begin{pmatrix} -d_\xi B\lambda_k - B d_\xi \lambda_k + d_\xi A \\ d_\xi E \end{pmatrix} \boldsymbol{r}_k + M d_\xi \boldsymbol{r}_k = 0. \tag{25}$$

Multiplying (25) *left from $\boldsymbol{l}_k^T$, using* (15) *and applying the chain rule to obtain that*

$$d_\xi B = (\partial_U B)\, d_\xi U = \left(d_U^2 G\right) \boldsymbol{r}_k,\ d_\xi A = (d_U A)\, d_\xi U = d_U^2 F \boldsymbol{r}_k,$$
$$d_\xi E = (d_U E)\, d_\xi U = d_U^2 H \boldsymbol{r}_k,$$

and noticing that $d\lambda/d\xi = \nabla\lambda_k \cdot r_k$, we finally obtain:

$$\boldsymbol{l}_k^T \begin{pmatrix} -d_U^2 G \boldsymbol{r}_k \lambda_k - B \frac{d\lambda_k}{d\xi} + d_U^2 F \boldsymbol{r}_k \\ d_U^2 H \boldsymbol{r}_k \end{pmatrix} \boldsymbol{r}_k = 0, \text{ or}$$
$$\nabla\lambda_k \cdot r_k \left(\boldsymbol{l}_k^T \begin{pmatrix} B \\ 0 \end{pmatrix} \boldsymbol{r}_k \right) = \boldsymbol{l}_k^T \begin{pmatrix} -\lambda_k d_U^2 G + d_U^2 F \\ d_U^2 H \end{pmatrix} (\boldsymbol{r}_k, \boldsymbol{r}_k) \quad \square \tag{26}$$

2.1.2 Discontinuities and shock waves

The Rankine-Hugoniot condition (RHC) for (1), (2) is written as:

$$v^s \left(G(U^+) - G(U^-) \right) = F(U^+) - F(U^-), \tag{27}$$
$$H(U^+) = 0, \tag{28}$$

where (U^-) is the state on the left of the shock and U^+ is the state on the right of the shock; v^s is the shock speed. For a given state U^-, the set of states U^+ that satisfies the *RHC* is called *Rankine-Hugoniot locus (RH locus) of* U^- and it is denoted by $\mathcal{RH}(U^-)$:

$$\mathcal{RH}(U^-) = \{U^+ \in \mathcal{S} \text{ satisfying RHC (27), (28) for all } v^s \}. \tag{29}$$

Notice that from (28) if $det(D_W H(U^-)) \neq 0$, from Implicit Function Theorem (IFT), there is an open neighborhood such that $W^+ = W(V^+)$, for V^+ being a vector of m independent variables. Moreover, the system (27),(28)

yields a system with $n+1$ unknowns, which are, (U^+, s) for n equations. Since we assume that the system is formally hyperbolic, Def. 2, the Rankine Hugoniot locus depends on a single parameter in the neighborhood of U^-. Moreover, the set obtained solving (27), (28) is the same set obtained when the variables are restricted on the surface $\mathcal{S}$, see Eq. (105).

2.1.3 Travelling wave for PDAE

Once the problem on the existence of the solution system (1) is solved, we are interested in analyzing the existence of viscous profiles or traveling waves for shocks on the system (1), (2). Under the assumption that such solution exist, we find sufficient conditions to find it taking into account viscous effects on the system. A procedure consist of to reduce a system of partial differential equation to a easier ordinary differential equation.

Let us denote the system (1) with the viscous effects, i.e.,

$$\partial_t G(V,W) + \partial_x F(V,W) = \delta \partial_x \left(\mathcal{B}(V,W) \partial_x (V,W) \right), \tag{30}$$

where $V = (U_1, \cdots, U_m)$ and $W = (W_{m+1}, \cdots, U_n)$.

In such sense the following proposition is valid

Proposition 3 *Assume that the system* (1), (2) *is formally hyperbolic and the square matrix* $\partial_W H$ *is invertible then the traveling wave equation of system* (1), (2) *is*

$$-v^s \left(\tilde{G}(V) - \tilde{G}(V^-) \right) + \left(\tilde{F}(V) - \tilde{F}(V^-) \right) = \tilde{B}(V) d_\eta V, \tag{31}$$

where v^s *is the shock velocity of the shock between* V^- *and* V

$$\tilde{G}(V) = G(V, W(V)), \quad \tilde{F}(W) = F(V, W(V)), \tag{32}$$

and $\tilde{B}$ *is given by*

$$\tilde{B}(V) = \mathcal{B}(V, W(V)) \begin{pmatrix} I_{m\times m} \\ \partial_V W(V) \end{pmatrix}, \tag{33}$$

where $W(V)$ *satisfies* (4) *and*

$$(V^-, W^-) = \lim_{\eta \longrightarrow -\infty} (V(\eta), W(\eta), \ (V,W) = \lim_{\eta \longrightarrow +\infty} (V^+(\eta), W^+(\eta)). \tag{34}$$

Proof: The proof is straightforward. Let U^* satisfying $H(U^*) = 0$ and $\partial_W H(U^*)$ invertible, then by the Implicit Function Theorem there exist a local difeomorphism $W = W(V)$ in the neighborhood of U^* and $H(V, W(V)) = 0$ and equation (4) holds.

Assuming that $W = W(V)$ on the equilibrium surface $\mathcal{S}$, and substituting this in Eq. (30), we obtain the viscosity formulation of (5) which is written as:

$$\partial_t G(V,W) + \partial_x F(V,W) = \delta \partial_x \left(\mathcal{B}(V,W) \partial_x (V,W) \right). \tag{35}$$

Assuming that the solution of (35) is smooth, the chain rule on the right hand side term of (35) gives:

$$\partial_t \tilde{G}(V) + \partial_x \tilde{F}(V) = \delta \partial_x \left(\tilde{B}(V) \partial_x (V) \right) \tag{36}$$

where $\tilde{G}(V)$,$\tilde{F}(V)$ and $\tilde{B}(V)$ are given in (33).

A traveling solution is a smooth solution that is self similar in the variable $\eta = (x - v^s t)/\delta$, i.e., $(V, W) = (V(\eta), W(\eta))$, connecting two equilibria (V^-, W^-) to (V^+, W^+) with (34).

Here $H(V^-, W^-)$, $H(V^+, W^+)$ and all the derivatives evaluated in (V^-, W^-) and (V^+, W^+) vanish. Integrating (30) from $-\infty$ to η and using (2) we obtain:

$$-v^s \left(G(V, W) - G^- \right) + F(V, W) - F^- = \mathcal{B}(V, W) d_\eta (V, W), \tag{37}$$

$$H(V, W) = 0, \tag{38}$$

where $G^- = G(V^-, W^-)$ and $F^- = F(V^-, W^-)$. For the system (37), (38) we have m ordinary differential equations and p algebraic equations for n unknowns. Since we are interested in obtaining a system of equations of ODE's connecting equilibria, to obtain a complete system, using index theory, we differentiate (38) with respect to variable η and we obtain:

$$\partial_V H d_\eta V + \partial_W H d_\eta W = 0 \quad \longrightarrow \quad d_\eta W = -\left(\partial_W H\right)^{-1} \left(\partial_W H\right) d_\eta W \tag{39}$$

The resulting system has index 1. Notice that (4), (39) given the same equation. Substituting (39) in Eq. (37) we obtain:

$$-v^s \left(G(V, W) - G^- \right) + F(V, W) - F^- = \hat{B}(W) d_\eta W, \tag{40}$$

where $\tilde{B}$ is given in Eq. (33).

Similarly, we integrated (33) for the reduced model and we obtain also (40), i.e., both formulations given the same solution for the traveling profile, which we summarize in the following Proposition:

Proposition 4 *The systems of equations* (30), (2) *and* (33), (34) *generate the same system of ODE's.*

3 Equivalence of the PDAE and the auxiliary PDE.

Here, we prove equivalence of weak solutions of (1), (2) and the auxiliary PDE (1), (6). We assume initial conditions satisfies Eq. (2)

Definition 3 We say that $U(x, t) \in (L^\infty(\mathbb{Q}))^n$, for $\mathbb{Q} \subset \mathbb{R} \times \mathbb{R}^+$, is a weak solution of PDEs (1), (6) if the solution satisfies the system in distribution sense, see [13, 41].

We state the following result

Proposition 5 *Let $U(x,t)$ be a piecewise continuous function, then $U(x,t)$ is a weak solution of PDEs* (1), (6) *if, and only if, $U(x,t)$ is a weak solution of PDAEs* (1), (2).

The proof uses classical theory and is omitted here.

Besides this result, one can prove more direct results by studying the Hugoniot locus and the integral curves.

3.1 Characteristic eigenpairs and rarefaction waves

To construct the rarefaction waves, we first obtain the integral curves. To do so, we assume that $U(\xi)$ is a smooth curve parameterized by ξ. Substituting $U(\xi)$ in (1), (6) we obtain a system similar to (11) and (12), with the latter replaced by $-\xi E dU/d\xi = 0$. For convenience we revert the sign of this equation. To obtain the eigenpair $(\lambda, \boldsymbol{r})$:

$$\begin{aligned} &det\begin{pmatrix} -\lambda B + A \\ E\lambda \end{pmatrix} = 0 \quad \longrightarrow \quad \lambda^p det\left(M(\lambda)\right) = 0, \\ &\text{for the eigenvectors } \boldsymbol{r} \text{ solutions of } \begin{pmatrix} -\lambda B + A \\ E\lambda \end{pmatrix} \boldsymbol{r} = 0. \end{aligned} \tag{41}$$

The matrices A, B, E are obtained in Eqs. (11), (12) and M is given in (13). Notice that the eigenvalues λ vanish or are solutions of $det(M) = 0$. The multiplicity of the eigenvalue $\lambda = 0$ is at least p. Moreover, if the eigenvalue λ is nonzero, the associated eigenspace is the same as the one obtained by solving (13.a).

Moreover, if there are m different eigenvectors associated to the m eigenvalues (accounting for their multiplicities) λ of Λ, these eigenvectors, locally, form a basis of $\mathcal{S}$.

For each pair $(\lambda \neq 0, \boldsymbol{r})$ and for initial data on the surface $\mathcal{S}$ i.e. U^* satisfying $H(U^*) = 0$, the integral curve associated to fixed family $(\lambda, \boldsymbol{r})$ remains on $\mathcal{S}$. This invariance comes from the second equation in (41.c), because the eigenvectors satisfy:

$$E\boldsymbol{r} = 0, \quad \longrightarrow \quad (\partial_U H)\, d_\xi U = 0 \quad \longrightarrow \quad d_\xi H = 0. \tag{42}$$

Since r and $-r$ are eigenvectors, we choose as admissible the direction such that $\nabla\lambda \cdot \boldsymbol{r} > 0$. The resulting integral curve is called the rarefaction wave.

The field $(\lambda = 0, \boldsymbol{r})$ is obtained by solving

$$\begin{pmatrix} A \\ 0 \end{pmatrix} \boldsymbol{r} = 0.$$

This field is linearly degenerate, i.e., $\nabla\lambda \cdot \boldsymbol{r} = 0$.

We see that the system of PDAEs (1), (2) is formally hyperbolic if, and only if, the system of PDEs (1), (6) is hyperbolic. The hyperbolicity is understood as in Definition 2, if we disregard the algebraic equations. The proof is straightforward once we notice that the analyses for $\lambda = 0$ and $\lambda \neq 0$ require different arguments.

3.2 Hugoniot Locus and Shock waves

The Hugoniot locus is obtained from the "Rankine-Hugoniot" condition for (1), (6):

$$v^s(G(U^+) - G(U^-)) = F(U^+) - F(U^-), \tag{43}$$

$$v^s(H(U^+) - H(U^-)) = 0. \tag{44}$$

For U^- fixed the Hugoniot-locus, denoted as $\mathcal{RH}(U^-)$, consists of the states U^+ satisfying (43), (44) for any v^s. Conditions of the functions F, G and H to guarantee the existence of curves satisfying the system of equation (43) – (44) are not trivial. However there are several examples of physical interest where the numerical construction of Hugoniot-locus is successful, see e.g. [1, 2, 6]. However, assuming that such curves exist, we can deduce some necessary conditions about how the Riemann solution still on the surface $\mathcal{S}$ when initial condition belongs to $\mathcal{S}$. To verify that invariance, we use the *compatibility principle of wave sequence*, i.e. the sequence of waves are ordered from the slower to faster.

If $v^s \neq 0$ and U^- lies on the surface $\mathcal{S}$, i.e. satisfies $H(U^-) = 0$, then from (44) we obtain that $H(U^+) = 0$, i.e., the shock curve remains on the surface $\mathcal{S}$, moreover the system (43), (44) reduces to (27), (28).

For the case $v^s = 0$, we have from (43) and (44):

$$F(U^+) = F(U^-). \tag{45}$$

However, $H(U^+) - H(U^-)$ might be nonzero in general. This implies that states U^+ and U^- are not necessarily in $\mathcal{S}$. In particular, for $\lambda = v^s = 0$, i.e., the field is linearly degenerate. The above paragraph are the basis for the following

Claim 1 *Consider Riemann data* (3). *Assume that the Riemann solution for* (1), (6) *consists of a sequence of states and waves (shock, rarefactions and contact discontinuities) connecting these states:*

$$U_L = U_1 \xrightarrow{\omega_1} U_2 \xrightarrow{\omega_2} U_3 \cdots \xrightarrow{\omega_{k-1}} U_k = U_R. \tag{46}$$

In addition, if we assume that $H(U_L) = H(U_R) = 0$, *i.e., the Riemann data are given on the surface* $\mathcal{S}$, *then the Riemann solution* $U(x,t)$ *of form* (46) *remains on the surface* $\mathcal{S}$.

The sketch of the proof of this claim is constructive and general. It should be possible to use it for several models.

Sketch of proof:

Consider the Riemann solution. If we assume by contradiction that the Riemann solution leaves the surface $\mathcal{S}$, there exists a sequence of consecutive states on this solution far from $\mathcal{S}$. Since $U_L \in \mathcal{S}$, the first state of this sequence is not U_L, from previous calculation, to reach the first state of this sequence we have a contact with speed zero (waves leaving $\mathcal{S}$ have zero speed). As

$U_R \in \mathcal{S}$, the last state of this sequence is not U_R, then there exists a contact discontinuity with zero speed connecting the last state of this sequence to $\mathcal{S}$.

From the compatibility principle of wave sequence, all waves of this sequence are contact discontinuity with zero speed. Thus, from *shock triple rule*, see [26], there is only a contact discontinuity connecting the first and the last state of this sequence i.e., the wave sequence does not leave the surface $\mathcal{S}$ and we prove our statement.

□.

The Claim 1 is used to establish the equivalence between the solutions of systems (1), (2) and (1), (6). Such connection, we need to obtain an entropy pairs ($\mathcal{U}$,$\mathcal{U}_1$) (see Appendix B). However, we stress out that the existence of entropy pairs is very rare for most part of hyperbolic systems, however it is very useful for theoretical purposes, including in numerical methods, we briefly discuss this theory in the appendix B.

3.3 Composite wave curve

The i-rarefaction wave stop when either arrive the physical boundary or along the integration path there exist a point satisfying

$$\chi_i(U) = \nabla\lambda_i \cdot \boldsymbol{r}_i = 0, \tag{47}$$

with $\nabla\chi_i(U) \cdot \boldsymbol{r}_i \neq 0$.

In this case, the continuation method for wave curves require a curve passing the submanifold given by $\nabla\lambda_i \cdot \boldsymbol{r}_i = 0$. To construct such wave curve the composite wave is proposed in [32,33] for system of conservation laws. Here, we present a generalization of such concept for PDAEs. A composite wave curve satisfies the set of equations

$$v^s(G(U^+) - G(U^-)) = F(U^+) - F(U^-), \tag{48}$$

$$v^s(U^+, U^-) = \lambda_i(U^-), \tag{49}$$

$$H(U^+) = 0. \tag{50}$$

Here v^s is the shock velocity and λ_i the characteristic velocity associated to the i-rarefaction.

Parametrizations for composite wave curve for system of conservation law were studied in [4]. Such parametrizations can be extended for PDAEs (48)-(50).

4 Relaxation formulation

In general the exact solution for algebraic relation (2) is not available, so the initial data lies near rather than the equilibrium surface $\mathcal{S}$. The nice feature of relaxation it is implement by means of orbits for such initial data which tend to $\mathcal{S}$ monotonically. The purpose of this section is to verify these features

of relaxation. On the other hand, the numerical implementation of relaxation gives rise to stable and reliable algorithm, as shown in Section 5.

To prove that the solution of (1), (7) tends to equilibrium surface $\mathcal{S}$ we define:

Definition 4 Consider the weight function $\omega_\epsilon(x)$ (see Fig. 1) and a $U(x,t)$ solution of eqs. (1), (7). Let H_1 and H_2 be two vector valued functions on the solution space $\mathcal{V}_\tau$ of (1), (7), we define:

$$\Big\{H_1, H_2\Big\}(t) = \int_{-\infty}^{\infty} H_1(U(t,x))H_2(U(t,x))\omega_\epsilon(x)dx, \tag{51}$$

where the product H_1 and H_2 is understood as the inner product between two vectors. Here, $\omega_\epsilon(x)$ is the continuous top hat function given by:

$$\omega_\epsilon(x) = \begin{cases} 1 & \text{if } x \in [a+\epsilon, b-\epsilon], \\ \frac{x}{\epsilon} - \frac{a}{\epsilon} & \text{if } x \in (a, a-\epsilon), \\ -\frac{x}{\epsilon} + \frac{b}{\epsilon} + 1 & \text{if } x \in (b, b+\epsilon), \\ 0 & \text{otherwise.} \end{cases} \tag{52}$$

Here $a < b \in \mathbb{R}$ are any value and $0 < \epsilon << 1$. Let us denote

$$|H(t)|^2 = \Big\{H, H\Big\}(t). \tag{53}$$

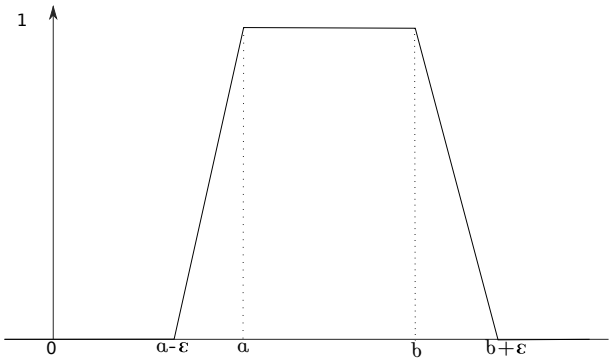

Fig. 1 The weight function $\omega_\epsilon(x)$.

Proposition 6 *Let $U(x,t) \in L^\infty(\mathbb{R} \times \mathbb{R}^+)$ be a solution of* (1), (7)*, where $H(U)$ is a continuously differentiable function. Then the solution of PDRE* (1), (7) *for increasing t satisfies :*

$$|H(t)| \longrightarrow 0, \tag{54}$$

i.e., Eq. (2) *is satisfied almost everywhere in the support of ω_ϵ.*

Proof: We differentiate $|H(t)|^2$, with respect to variable t (weak derivative), thus, we obtain:

$$d_t|H(t)|^2 = d_t\left(\Big\{H,H\Big\}(t)\right) = 2\Big\{\partial_t H, H\Big\}(t). \tag{55}$$

For the last equality comes from the symmetry of pseudo-inner product. Using (7) in (55), we obtain:

$$d_t|H(t)|^2 = 2\left\{-\frac{H}{\tau}, H\right\}(t) = -\frac{2|H(t)|^2}{\tau}. \tag{56}$$

From (56), we have a ODE for the variable $|H(t)|^2$, for the initial condition $U_0 = U(0,x)$, we obtain:

$$|H(0)|^2 = \Big\{H,H\Big\}(0) \tag{57}$$

The solution of (56) with initial condition (57) gives us (after some algebra):

$$|H(t)| = |H(0)|e^{-t/\tau}. \tag{58}$$

Notice that when t increases (or $\tau \longrightarrow 0$) $|H(t)| \longrightarrow 0$, which implies that $H(U(x,t))$ goes to zero almost everywhere in the support of ω_ϵ, i.e., the solution tends to the surface $\mathcal{S}$ almost everywhere. □

From Eq. (58), if the initial data belongs to $\mathcal{S}$, then the solution of PDRE (1), (7) remains on $\mathcal{S}$.

5 Numerical Method for PDAEs - The RCD Solver a general case

Numerical methods are largely used to solve systems of partial differential equations of form (1) Cauchy initial data. There are several kinds of schemes that are used to this end, which we cite, *finite differences*, see [24,30,45], *finite elements*, see [44], *finite volumes*, see [16,24,31], *spectral techniques*, see [17,19]. For the majority of methods, the theory for linear systems of partial differential equations of form (1) is very well known, however, there are few results for system of partial differential and algebraic equations. In Appendix C, we discuss the theory for the linear case, where we prove the convergence and we extend the celebrated Lax-Richtmyer theorem, see [29,45].

Here we are interested in finite difference method. To obtain the discretization by these methods, we introduce a grid of points on the domain (x,t) of PDAEs. The grid are the points $\mathcal{D}_d = \{(x_i,t^\gamma) = (i\Delta x, \gamma\Delta t)$, for $i \in \mathbb{Z}$, $\gamma \in \{0,1,2,\cdots\}\}$, where Δx, Δt are positive numbers. We define $\alpha = \Delta t/\Delta x$. Although Δx, Δt do not need to be constant, here in this work we will assume this condition.

To discretize the system of PDAEs (1), (2) (or Eq. (6)), we take the restriction of vector $V(x,t)$ on $\mathcal{D}_d$ and we call $U_i^\gamma = ((u_1)_i^\gamma, (u_2)_i^\gamma, \cdots, (u_n)_i^\gamma)$ the function defined in the grid $\mathcal{D}_d$ and as $V_i^\gamma = V(x_i,t^\gamma)$ the function $V(x,t)$ defined continuously on (x,t) and restricted to $\mathcal{D}_d$. In D, we obtain the extension

of some classical numerical methods for PDAEs (1), (2) (or Eq. (6)). In this Section, we obtain numerical method for a more complex group of equations, in which we consider also diffusive effects. For this class of problems, because of stability, the most effective methods are the implicit ones, see [45].

In this section, we developed numerical methods for Eq. (8), (9). Most first or second order schemes in space and time for reaction-convection-diffusion equations give rise to numerical methods in which the most consuming task is solving block-tridiagonal linear systems, [43,46]. Among the most used techniques to solve these linear systems, we cite block **LU** factorization, divide and conquer and cyclic reduction methods, see [9–11,36,37]. Block **LU** factorization is by far the fastest among these if there is only one processing unit, although it relies on block diagonal dominance [14]. The divide and conquer algorithm performs better in shared memory machines with small to medium sized blocks, and for large blocks (several PDEs), the cyclic reduction is usually a slightly better choice, particularly for distributed memory machines where the cost of communication is high [5,21,22]. All these methods have been developed and analyzed for uniform block-tridiagonal matrices (PDEs) rather than non-uniform block-tridiagonal ones (PDAEs).

Our solver has specialized routines implemented to take advantage of the zero entries of the non-uniform block-tridiagonal linear system obtained from numerical discretization. We propose a version of the Divide and Conquer algorithm that preserves the sparsity of the matrix during the process of solution, obtaining as a result a method with fewer floating point operations and fewer operation that require memory access. We also obtain asymptotic speedup expressions for the cases of few or many algebraic restrictions in the PDAEs system (8). We also focus on adapting the classical method for solving linear systems used for the numerical resolution of these equations to take advantage of the sparsity of the linear systems, obtaining a fast and robust solver for PDAEs. This approach has also the advantage that the discretization of (1) for implicit methods, such as the Crank-Nicolson method, gives rise to linear systems that are efficiently solved using classical sequential or parallel methods, such as the block **LU** decomposition method [36] or the divide and conquer [37] and cyclic reduction methods [10]. When one cannot obtain the surface $\mathcal{S}$ directly, in the second case described above, the solution to solve the system (1), (2) using the classical available tools is to consider the algebraic restrictions as another system of degenerate PDEs. Although one can solve these equations as if they were n coupled PDEs, the calculations and data necessary for doing so increase dramatically when compared to the case in which the solver routines identify the PDEs which are actually algebraic constraints, as strategies we propose a parallelization of methods.

The remaining of this section is organized as follows: in Subsection 5.1 we briefly present the underlying numerical scheme implemented in RCD, and how we developed it to fit the classical discretization techniques known in the literature for finite difference schemes. Section 5.2 presents a discussion on different block **LU** decompositions and their impact in terms of floating point operation costs and sparsity of the matrix during the decomposition and

substitution part of the solver. We perform a floating point operation count of the methods discussed and we compare them in order to determine the most efficient one for non-uniform block-tridiagonal matrices, which we study in Appendix F. In Section 5.3, we present the Divide and Conquer method for non-uniform block-tridiagonal matrices. We compare the cost of the parallel algorithm proposed in terms of floating point operations and we compare it to the cost of the most efficient sequential block **LU** algorithm. In Section 5.4, we present some numerical experiments for particular PDAEs. We conduct some experiments to determine the validity of the asymptotic speedups obtained theoretically. We also conduct experiments to determine the behavior of RCD for different sized problems.

5.1 Numerical Scheme

We present now the general finite difference scheme used to approximate (8), (9) numerically. We introduce the scalars $\alpha_i \in [0,1]$, $\beta_i = 1-\alpha_i$, $i = 1,2$. These values are used in such a way that the scheme is explicit if $\alpha_i = 0$, implicit if $\alpha_i = 1$ or Crank-Nicholson if $\alpha_i = 1/2$. The difference equations that allow such correspondence are

$$(G_t)^{k+\alpha_i} = \frac{G^{k+1}-G^k}{\Delta t} + O((\beta_i-\alpha_i)\Delta t) + O((\Delta t)^2), \tag{59}$$

and

$$\phi^{k+\alpha_i} = \alpha_i \phi^{k+1} + \beta_i \phi^k + O((\Delta t)^2), \tag{60}$$

where ϕ is a general vector-valued or matrix-valued function, and the subscript $(\cdot)_t$ denotes time differentiation. Applying these equations to system (8), (9) and disregarding the errors, we obtain the approximation

$$\begin{aligned} \frac{G_l^{k+1}-G_l^k}{\Delta t} + \alpha_1 (F_x)_l^{k+1} + \beta_1 (F_x)_l^n = \alpha_1 ((\mathcal{B}U_x)_x)_l^{k+1} + \beta_1 ((\mathcal{B}U_x)_x)_l^k + \\ +\alpha_1 R_l^{k+1} + \beta_1 R_l^k \\ \frac{\tau(H_l^{k+1}-H_l^k)}{\Delta t} = -\alpha_2 H_l^{k+1} - \beta_2 H_l^k \end{aligned} . \tag{61}$$

The approximation in space for the flux and diffusion terms can be done in different ways. To fix ideas, we discretize $(F_x)_l$, $((\mathcal{B}U_x)_x)_l$ using central differences, so that

$$(F_x)_l = \frac{F_{l+1}-F_{l-1}}{2\Delta x} + O((\Delta x)^2), \tag{62}$$

and

$$((\mathcal{B}U_x)_x)_l = \frac{1}{2(\Delta x)^2}\left((\mathcal{B}_{l+1}+\mathcal{B}_l)(U_{l+1}-U_l)\right. \tag{63}$$

$$\left. -(\mathcal{B}_l+\mathcal{B}_{l-1})(U_l-U_{l-1})) + O((\Delta x)^2\right). \tag{64}$$

Thus, the final scheme is written in the form

$$\kappa G_l^{k+1} - \mu_1[(\mathcal{B}_{l+1}^{k+1} + \mathcal{B}_l^{k+1})(U_{l+1}^{k+1} - U_l^{k+1}) - (\mathcal{B}_l^{k+1} + \mathcal{B}_{l-1}^{k+1})(U_l^{k+1} - U_{l-1}^{k+1})] +$$

$$+\eta_1(F_{l+1}^{k+1} - F_{l-1}^{k+1}) - \alpha_1 R_l^{k+1} =$$

$$\kappa G_l^k + \mu_2[(\mathcal{B}_{l+1}^k + \mathcal{B}_l^k)(U_{l+1}^k - U_l^k) - (\mathcal{B}_l^k + \mathcal{B}_{l-1}^k)(U_l^k - U_{l-1}^k)] -$$

$$-\eta_2(F_{l+1}^k - F_{l-1}^k) + \beta_1 R_l^k, \tag{65}$$

$$\tau\kappa H_l^{k+1} + \alpha_2 H_l^{k+1} = \tau\kappa H_l^k - \beta_2 H_l^k, \tag{66}$$

where the constants are given by

$$\kappa = \frac{1}{\Delta t}, \quad \mu_1 = \frac{\alpha_1}{2(\Delta x)^2}, \quad \eta_1 = \frac{\alpha_1}{2\Delta x},$$

$$\mu_2 = \frac{\beta_1}{2(\Delta x)^2}, \quad \eta_2 = \frac{\beta_1}{2\Delta x}.$$

Notice that the scheme is second order convergent in time only if $\alpha_i = 1/2$, $i = 1, 2$. If the scheme is also second order convergent in space, then it is desirable that the approximation of the boundary conditions are also second order convergent. For example, if we consider Robin boundary conditions such as

$$\mathscr{A}_i U + \mathscr{B}_i U_x = \mathscr{V}_i, \tag{67}$$

with $\mathscr{A}_i, \mathscr{B}_i \in \mathbb{R}^{m\times n}$, $\mathscr{V}_i \in \mathbb{R}^m$ and the index $i = L, R$ used to denote left and right boundaries respectively, then a second order convergent approximation in space of these equations would be

$$\alpha_1\mathscr{A}_L U_0^{k+1} + \eta_1\mathscr{B}_L(-3U_0^{k+1} + 4U_1^{k+1} - U_2^{k+1}) = \mathscr{V}_L$$
$$- \beta_1\mathscr{A}_L U_0^k - \eta_2\mathscr{B}_L(-3U_0^k + 4U_1^k - U_2^k), \tag{68}$$

and

$$\alpha_1\mathscr{A}_R U_M^{k+1} + \eta_1\mathscr{B}_R(U_M^{k+1} - 4U_{M-1}^{k+1} + 3U_{M-2}^{k+1}) = \mathscr{V}_R$$
$$- \beta_1\mathscr{A}_R U_0^k - \eta_2\mathscr{B}_R(U_M^k - 4U_{M-1}^k + 3U_{M-2}^k). \tag{69}$$

A first order convergent finite difference is used to approximate U_x if the scheme in the internal mesh points is only first order space convergent. In that case one would approximate the flux term in equation (62) using a forward or backward difference rather than a central one.

To solve the system (65), (66) with (68), (69), we apply the Newton method

$$\mathbf{G}(V^{k+1}) = 0,$$

where $\mathbf{G}(V^{k+1}) = \mathbf{F}(V^{k+1}) - \mathbf{Y}(V^k)$ is such that $\mathbf{F}(V^{k+1})$ is the left hand side of (65), (66), (68), (69), while $\mathbf{Y}(V^k)$ is its right hand side. Given the initial guess $v_{(0)} = U^k$, we perform the iterations

$$v_{(j+1)} = v_{(j)} + \delta_{(j)},$$

where

$$J_{\mathbf{G}}(v_{(j)})\delta_{(j)} = -\mathbf{G}(v_{(j)}), \tag{70}$$

until $||\delta_{(j)}|| \leq \epsilon_1$, $||\mathbf{G}(v_{(j+1)})|| \leq \epsilon_2$ or $j \leq j_{max}$. The equation (70) represents the most expensive calculation in the scheme. However, the jacobian matrix $J_{\mathbf{G}}(v_{(j)})\delta_{(j)}$ is typically block-tridiagonal[1], therefore one can use certain properties to efficiently solve the system. This discussion is presented next in more details.

5.2 Block **LU** decompositions

The block **LU** decompositions are known to be a good tool to solve general block-tridiagonal linear systems directly, as long as the matrix involved is diagonally dominant either by columns or rows [36,14]. For the system (70), the matrix $J_{\mathbf{G}}$ is composed of $M+1$ square blocks of size $n \times n$, and $2M$ non-uniform blocks of size $m \times n$ ($2(M+1)$ if we consider the discretized Robin boundary conditions (68), (69) rather than the classical first order approximation). Because of the non-uniformity of these last blocks, certain decompositions are better suited to be used in the solution of (70) because they do not introduce non-zero entries. As a result, storage is kept at a minimum and computations are saved during the execution of the linear system solver. In this section, we present a version of the block **LU** decomposition that preserves the structure of the non-uniform block-tridiagonal matrix in which it is applied. We also compare two different block **LU** decompositions in terms of floating point operations, and we derive two different asymptotic speedups for the case of only one right hand side vector and for the case of several right hand side vectors.

Let us start by defining a non-uniform block-tridiagonal linear system:

$$\mathbf{AX} = \begin{pmatrix} A_1 & B_1 & & & & \\ C_1 & A_2 & B_2 & & & \\ & C_2 & A_3 & B_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & C_{M-2} & A_{M-1} & B_{M-1} \\ & & & & C_{M-1} & A_M \end{pmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_{M-1} \\ X_M \end{bmatrix} = \begin{pmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{M-1} \\ D_M \end{pmatrix} = b, \tag{71}$$

where A_i, B_i, C_i are square blocks of size $n \times n$, X_i, D_i are vectors of size $n \times 1$, for $i = 1, \cdots, M$. The blocks B_i, C_i have their last $p = n - m$ rows with zero entries. Out of the several different **LU** decompositions for matrix **A** above, we extract the two that have the property that the diagonal blocks of either **L** or **U** are identity blocks. We call them the "top-to-bottom" version,

[1] Appendix E shows the structure of the blocks of $J_{\mathbf{G}}$.

given by

$$\mathbf{LU} = \begin{pmatrix} I & & & & \\ \overline{C}_1 & I & & & \\ & \ddots & \ddots & & \\ & & \overline{C}_{M-2} & I & \\ & & & \overline{C}_{M-1} & I \end{pmatrix} \begin{pmatrix} \overline{A}_1 & \overline{B}_1 & & & \\ & \overline{A}_2 & \overline{B}_2 & & \\ & & \ddots & \ddots & \\ & & & \overline{A}_{M-1} & \overline{B}_{M-1} \\ & & & & \overline{A}_M \end{pmatrix}, \tag{72}$$

and the "left-to-right" version, given by

$$\mathbf{LU} = \begin{pmatrix} \overline{A}_1 & & & & \\ \overline{C}_1 & \overline{A}_2 & & & \\ & \ddots & \ddots & & \\ & & \overline{C}_{M-2} & \overline{A}_{M-1} & \\ & & & \overline{C}_{M-1} & \overline{A}_M \end{pmatrix} \begin{pmatrix} I & \overline{B}_1 & & & \\ & I & \overline{B}_2 & & \\ & & \ddots & \ddots & \\ & & & I & \overline{B}_{M-1} \\ & & & & I \end{pmatrix}. \tag{73}$$

These names are inspired by the decomposition process, as in Algorithms 1 and 3. Although these algorithms have the same mathematical properties, [20], their implementation causes one method to be better than the other in terms of computational cost. The explanation behind this fact is given by line 3 of Algorithm 1 and line 3 of Algorithm 3. In the top-to-bottom **LU** decomposition, one needs to solve the linear system

$$(\overline{A}_i)^T (\overline{C}_i)^T = (C_i)^T, \tag{74}$$

in which the right hand side matrix $(C_i)^T$ has the last p columns with zero entries. As a result, the matrix $(\overline{C}_i)^T$ has also p columns with zero entries, which do not need to be stored. In practice, the top-to-bottom **LU** decomposition can be stored in the same data structure used to store **A**. However, in the left-to-right **LU** decomposition, the full-rank system that needs to be solved is

$$\overline{A}_i \overline{B}_i = B_i. \tag{75}$$

In this case one cannot guarantee that any entry of $\overline{B}_i$ will be zeroed by the decomposition process. As a result, more computations are needed, and if the decomposition is to be stored in the same data structure of **A**, the zero entries of B_i, C_i need to be stored. This comparison makes it clear that Algorithm 1 is more efficient to solve our problem. Algorithms 2 and 4 show how to use the top-to-bottom and left-to-right decompositions to solve the linear system (71). The process is basically the same in both algorithms: first we solve $\mathbf{L}Y = b$ and using Y we solve $\mathbf{UX} = Y$.

As in the block **LU** decompositions, there are two **UL** decompositions processes that have the property that the diagonal blocks of either **U** or **L** are

Algorithm 1 Top-to-bottom **LU** decomposition

Require: **A**
Ensure: **L**, **U**
1: $\overline{A}_1 = A_1$
2: **for** $i = 1$ **to** $M-1$ **do**
3: $\quad \overline{C}_i \leftarrow C_i \overline{A}_i^{-1}$
4: $\quad \overline{B}_i \leftarrow B_i$
5: $\quad \overline{A}_{i+1} \leftarrow A_{i+1} - \overline{C}_i \overline{B}_i$
6: **end for**

Algorithm 2 Top-to-bottom **LU** solver

Require: **L**, **U**, b
Ensure: **X**
1: $Y_1 = D_1$
2: **for** $i = 1$ **to** $M-1$ **do**
3: $\quad Y_{i+1} \leftarrow D_{i+1} - \overline{C}_i Y_i$
4: **end for**
5: $X_M = \overline{A}_M^{-1} Y_M$
6: **for** $i = M-1$ **to** 1 **do**
7: $\quad X_i \leftarrow \overline{A}_i^{-1}(Y_i - \overline{B}_i X_{i+1})$
8: **end for**

Algorithm 3 Left-to-right **LU** decomposition

Require: **A**
Ensure: **L**, **U**
1: $\overline{A}_1 = A_1$
2: **for** $i = 1$ **to** $M-1$ **do**
3: $\quad \overline{B}_i \leftarrow \overline{A}_i^{-1} B_i$
4: $\quad \overline{C}_i \leftarrow C_i$
5: $\quad \overline{A}_{i+1} \leftarrow A_{i+1} - \overline{C}_i \overline{B}_i$
6: **end for**

Algorithm 4 Left-to-right **LU** solver

Require: **L**, **U**, b
Ensure: **X**
1: $Y_1 = \overline{A}_1^{-1} D_1$
2: **for** $i = 1$ **to** $M-1$ **do**
3: $\quad Y_{i+1} \leftarrow \overline{A}_{i+1}^{-1}(D_{i+1} - \overline{C}_i Y_i)$
4: **end for**
5: $X_M = Y_M$
6: **for** $i = M-1$ **to** 1 **do**
7: $\quad X_i \leftarrow Y_i - \overline{B}_i X_{i+1}$
8: **end for**

identity blocks: the "bottom-to-top" process that leads to

$$\mathbf{UL} = \begin{pmatrix} I & \overline{B}_1 & & & \\ & I & \overline{B}_2 & & \\ & & \ddots & \ddots & \\ & & & I & \overline{B}_{M-1} \\ & & & & I \end{pmatrix} \begin{pmatrix} \overline{A}_1 & & & & \\ \overline{C}_1 & \overline{A}_2 & & & \\ & \ddots & \ddots & & \\ & & \overline{C}_{M-2} & \overline{A}_{M-1} & \\ & & & \overline{C}_{M-1} & \overline{A}_M \end{pmatrix}, \qquad (76)$$

and the "right-to-left" process that creates upper and lower matrices given by

$$\mathbf{UL} = \begin{pmatrix} \overline{A}_1 & \overline{B}_1 & & & \\ & \overline{A}_2 & \overline{B}_2 & & \\ & & \ddots & \ddots & \\ & & & \overline{A}_{M-1} & \overline{B}_{M-1} \\ & & & & \overline{A}_M \end{pmatrix} \begin{pmatrix} I & & & & \\ \overline{C}_1 & I & & & \\ & \ddots & \ddots & & \\ & & \overline{C}_{M-2} & I & \\ & & & \overline{C}_{M-1} & I \end{pmatrix}. \tag{77}$$

A similar analysis to that proposed for **LU** decompositions shows that Algorithm 7 is less efficient than Algorithm 5: the right-to-left **UL** decomposition forces performing more calculations and storing more memory than the bottom-to-top **UL** decomposition. Notice that if one were to apply an **LU** or **UL** decomposition with pivoting without taking into account the structure of the blocks, the pivoting process would demand storage of the zero entries of blocks B_i, C_i anyway. However, in the case of block decompositions, the pivoting process can be applied only for the linear systems related to A_i. Algorithms 6 and 8 show how to perform the linear systems $\mathbf{U}Y = b$, $\mathbf{LX} = Y$, necessary to compute the solution using the block **UL** decompositions.

Algorithm 5 Bottom-to-top **UL** decomposition

Require: **A**
Ensure: **L**, **U**
1: $\overline{A}_M = A_M$
2: **for** $i = M-1$ **to** 1 **do**
3: $\quad \overline{B}_i \leftarrow B_i \overline{A}_{i+1}^{-1}$
4: $\quad \overline{C}_i \leftarrow C_i$
5: $\quad \overline{A}_i \leftarrow A_i - \overline{B}_i \overline{C}_i$
6: **end for**

Algorithm 6 Bottom-to-top **UL** solver

Require: **L**, **U**, b
Ensure: **X**
1: $Y_M = D_M$
2: **for** $i = M-1$ **to** 1 **do**
3: $\quad Y_i \leftarrow D_i - \overline{B}_i Y_{i+1}$
4: **end for**
5: $X_1 = \overline{A}_1^{-1} Y_1$
6: **for** $i = 1$ **to** $M-1$ **do**
7: $\quad X_{i+1} \leftarrow \overline{A}_{i+1}^{-1}(D_{i+1} - \overline{C}_i X_i)$
8: **end for**

While the left-to-right and right-to-left algorithms seem to be discarded, they may be chosen over the top-to-bottom and bottom-to-top versions, depending on other issues. When the blocks B_i, C_i have all their entries filled with non-zero values, all the decompositions presented before perform similarly. However, notice that when decomposing, if the right hand side vectors

Algorithm 7 Right-to-Left **UL** decomposition

Require: **A**, M
Ensure: **L**, **U**
1: $\overline{A}_M = A_M$
2: **for** $i = 1$ **to** $M - 1$ **do**
3: $\quad \overline{C}_i \leftarrow \overline{A}_{i+1}^{-1} C_i$
4: $\quad \overline{B}_i \leftarrow B_i$
5: $\quad \overline{A}_i \leftarrow A_i - \overline{B}_i \overline{C}_i$
6: **end for**

Algorithm 8 Right-to-Left **UL** solver

Require: **L**, **U**, b
Ensure: **X**
1: $Y_M = \overline{A}_M^{-1} D_M$
2: **for** $i = M - 1$ **to** 1 **do**
3: $\quad Y_i \leftarrow \overline{A}_i^{-1} (D_i - \overline{B}_i Y_{i+1})$
4: **end for**
5: $X_1 = Y_1$
6: **for** $i = 1$ **to** $M - 1$ **do**
7: $\quad X_{i+1} \leftarrow Y_{i+1} - \overline{C}_i X_i$
8: **end for**

are already known, the first of the back substitutions can be executed while the decomposition is made ($\mathbf{L}Y = b$ or $\mathbf{U}Y = b$). This fact, along with the assumption that the cache size is big enough, allows one to save memory accesses, since the blocks used in the first of the back substitutions can be fetched from memory only once. In that scenario, which only happens for small blocks, the left-to-right and right-to-left decompositions save more memory operations than the bottom-to-top and top-to-bottom ones, since their back substitutions involve more block operations. In subsection 5.4, we will see numerical results that verify this statement. In Appendix $F.1$ we perform an analysis of floating point cost comparison of the block decomposition of the algorithms 1 to 4.

5.3 The Divide and Conquer Method

The Divide and Conquer Method for block-tridiagonal systems (DCB) has been designed and analyzed in [9,37]. The main idea of the method is to use Schur complements to isolate a few degrees of freedom in such a way that their solution is dependent on the solution of several smaller block-tridiagonal linear systems. These small linear systems are solved in parallel, and then the isolated degrees of freedom are also a solution to a block-tridiagonal linear system. In this Subsection, we present the DCB method for non-uniform block-tridiagonal linear systems. We derive a slightly improved asymptotic speedup than that obtained in [37] for uniform block-tridiagonal matrices, and we also show the asymptotic speedup in the case of non-uniform block-tridiagonal matrices.

To briefly present the DCB method, let **A** be a non-uniform block-tridiagonal matrix of size $nM \times nM$. The linear system associated to **A** is the one from

equation (71). Let us select $\rho-1$ unknown vectors ξ_j, and write them in terms of the remaining ρ unknown vectors $\mathbf{X}_i$. To better describe these relations, we partition the original system in the following form

$$\begin{pmatrix}
\mathbf{A}_1 & \theta_1 & & & & & & & \\
\gamma_1^T & \lambda_1 & \varphi_1^T & & & & & & \\
 & \omega_1 & \mathbf{A}_2 & \theta_2 & & & & & \\
 & & \gamma_2^T & \lambda_2 & \varphi_2^T & & & & \\
 & & & \omega_2 & \mathbf{A}_3 & \theta_3 & & & \\
 & & & & \ddots & \ddots & \ddots & & \\
 & & & & & \omega_{\rho-2} & \mathbf{A}_{\rho-1} & \theta_{\rho-1} & \\
 & & & & & & \gamma_{\rho-1}^T & \lambda_{\rho-1} & \varphi_{\rho-1}^T \\
 & & & & & & & \omega_{\rho-1} & \mathbf{A}_\rho
\end{pmatrix}
\begin{bmatrix}
\mathbf{X}_1 \\ \xi_1 \\ \mathbf{X}_2 \\ \xi_2 \\ \mathbf{X}_3 \\ \vdots \\ \mathbf{X}_{\rho-1} \\ \xi_{\rho-1} \\ \mathbf{X}_\rho
\end{bmatrix}
=
\begin{pmatrix}
b_1 \\ \beta_1 \\ b_2 \\ \beta_2 \\ b_3 \\ \vdots \\ b_{\rho-1} \\ \beta_{\rho-1} \\ b_\rho
\end{pmatrix}, \tag{78}$$

where

- $\mathbf{A}_i$, for $i = 1, 2, \ldots \rho$, are non-uniform block-tridiagonal submatrices of size $nM_i \times nM_i$;
- λ_i, for $i = 1, 2, \ldots \rho - 1$, are blocks of size $n \times n$;
- $\mathbf{X}_i$, b_i, for $i = 1, 2, \ldots \rho$, are vectors of size $nM_i \times 1$;
- ξ_i, β_i, for $i = 1, 2, \ldots \rho - 1$, are vectors of size $n \times 1$.

The remaining elements of the partition are matrices of size $nM_i \times n$ such that

$$\omega_i = \begin{bmatrix} \omega_i^N \\ \omega_i^Z \end{bmatrix}, \tag{79}$$

$$\varphi_i = \begin{bmatrix} \varphi_i^N \\ \varphi_i^Z \end{bmatrix}, \tag{80}$$

$$\theta_i = \begin{bmatrix} \theta_i^Z \\ \theta_i^N \end{bmatrix}, \tag{81}$$

$$\gamma_i = \begin{bmatrix} \gamma_i^Z \\ \gamma_i^N \end{bmatrix}, \tag{82}$$

where the superscript $[\cdot]^N$ denotes a non-uniform block of size $m \times n$ filled with nonzero entries, while the superscript $[\cdot]^Z$ denotes matrices of size $(n(M_i-1)+p) \times n$ filled with zero entries. The unknowns $\mathbf{X}_i$ are written in terms of ξ_j in the following manner

$$\mathbf{X}_i = \overline{b}_i - \overline{\omega}_{i-1}\xi_{i-1} - \overline{\theta}_i\xi_i, \qquad i = 1, 2, \ldots \rho, \tag{83}$$

where we have defined $\overline{b}_i$, $\overline{\omega}_{i-1}$, $\overline{\theta}_i$ to be the solution to the non-uniform block-tridiagonal linear systems

$$\mathbf{A}_i\overline{b}_i = b_i, \qquad \mathbf{A}_i\overline{\omega}_{i-1} = \omega_{i-1}, \qquad \mathbf{A}_i\overline{\theta}_i = \theta_i, \tag{84}$$

and $\omega_0\xi_0 = \theta_\rho\xi_\rho = 0$. Replacing the solution (83) into equations of the form

$$\gamma_i^T\mathbf{X}_i + \lambda_i\xi_i + \varphi_i^T\mathbf{X}_{i+1} = \beta_i, \tag{85}$$

with $\gamma_0^T\mathbf{X}_0 = \varphi_\rho^T\mathbf{X}_{\rho+1} = 0$, leads to the following non-uniform block-tridiagonal linear system for the unknowns ξ_i:

$$\begin{bmatrix} \overline{\lambda}_1 & \overline{\varphi}_1 & 0 & 0 & 0 & \cdots & 0 \\ \overline{\gamma}_1 & \overline{\lambda}_2 & \overline{\varphi}_2 & 0 & 0 & \cdots & 0 \\ 0 & \overline{\gamma}_2 & \overline{\lambda}_3 & \overline{\varphi}_3 & 0 & \cdots & 0 \\ 0 & 0 & \overline{\gamma}_3 & \overline{\lambda}_4 & \overline{\varphi}_4 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \\ 0 & 0 & 0 & \cdots & \overline{\gamma}_{\rho-3} & \overline{\lambda}_{\rho-2} & \overline{\varphi}_{\rho-2} \\ 0 & 0 & 0 & \cdots & 0 & \overline{\gamma}_{\rho-2} & \overline{\lambda}_{\rho-1} \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \\ \vdots \\ \xi_{\rho-2} \\ \xi_{\rho-1} \end{bmatrix} = \begin{bmatrix} \overline{\beta}_1 \\ \overline{\beta}_2 \\ \overline{\beta}_3 \\ \overline{\beta}_4 \\ \vdots \\ \overline{\beta}_{\rho-2} \\ \overline{\beta}_{\rho-1} \end{bmatrix}, \tag{86}$$

where the blocks and vectors above are given by

$$\begin{array}{ll} \overline{\lambda}_i = \lambda_i - \gamma_i^T\overline{\theta}_i - \varphi_i^T\overline{\omega}_i, & i = 1, 2, \ldots, \rho-1, \\ \overline{\varphi}_i = -\varphi_i^T\overline{\theta}_{i+1}, & i = 1, 2, \ldots, \rho-2, \\ \overline{\gamma}_i = -\gamma_{i+1}^T\overline{\omega}_i, & i = 1, 2, \ldots, \rho-2, \\ \overline{\beta}_i = \beta_i - \gamma_i^T\overline{b}_i - \varphi_i^T\overline{b}_{i+1}, & i = 1, 2, \ldots, \rho-1. \end{array} \tag{87}$$

The solution of the system (87) can be obtained in two different ways. The first one considers that ρ is small (typically the number of threads in a shared memory environment or the number of nodes of a cluster) and the system can be solved using a direct solver such as the block **LU** decomposition methods (see Subsection 5.2). The second one considers that ρ is as big as it can be, and therefore the partition process can be applied again and again for systems in the form of 87 until a small enough matrix is obtained. This latest methodology is commonly called cyclic reduction, [10,11]. Algorithm 9 summarizes the DCB method. Notice that maybe except for step 3, all the remaining steps are parallelizable.

The solution of the non-uniform block-tridiagonal linear systems (84) can be done using any direct solver. However, depending on the method, computations can be saved due to the particular structure of the right hand sides ω_{i-1}, θ_i. Let us consider, for instance, the block **LU** decompositions introduced in Subsection 5.2. The top-to-bottom block **LU** decomposition (72), when used to solve a linear system with the right hand side is θ_i, is such that the solution to the intermediate linear system $\mathbf{LY} = \theta_i$ gives $\mathbf{Y} = \theta_i$. As a result, the computations used to solve $\mathbf{LY} = \theta_i$ can be saved, reducing the cost of the Algorithm 2. Similarly, if the left-to-right **LU** decomposition (73) is applied to the linear systems with form $\mathbf{A}_i\overline{\theta}_i = \theta_i$, then the process used to solve $\mathbf{LY} = \theta_i$ is reduced to solving only one full-rank linear system with

Algorithm 9 Divide and Conquer Method

Require: $\mathbf{A}$, b, M, ρ
Ensure: $\mathbf{X}$

1: Solve $\begin{cases} \mathbf{A}_i\overline{b}_i = b_i, \text{ for } i = 1,\ldots,\rho \\ \mathbf{A}_i\overline{\omega}_{i-1} = \omega_{i-1}, \text{ for } i = 2,\ldots,\rho \\ \mathbf{A}_i\overline{\theta}_i = \theta_i, \text{ for } i = 1,\ldots,\rho-1 \end{cases}$

2: Calculate $\begin{cases} \overline{\lambda}_i = \lambda_i - \gamma_i^T\overline{\theta}_i - \varphi_i^T\overline{\omega}_i, & i = 1,2,\ldots,\rho-1, \\ \overline{\varphi}_i = -\varphi_i^T\overline{\theta}_{i+1}, & i = 1,2,\ldots,\rho-2, \\ \overline{\gamma}_i = -\gamma_{i+1}^T\overline{\omega}_i, & i = 1,2,\ldots,\rho-2, \\ \overline{\beta}_i = \beta_i - \gamma_i^T\overline{b}_i - \varphi_i^T\overline{b}_{i+1}, & i = 1,2,\ldots,\rho-1. \end{cases}$

3: Solve linear system (86)

4: Calculate $\mathbf{X}_i = \overline{b}_i - \overline{\omega}_{i-1}\xi_{i-1} - \overline{\theta}_i\xi_i, \qquad i = 1,2,\ldots\rho$

the size of the diagonal blocks. That fact also reduces the cost of algorithm 4. Unfortunately, no similar conclusions can be easily reached for the case of block **LU** decompositions with right hand side ω_{i-1}, and therefore the impact of this observation in the cost of the method is small. Notice that although the savings related to each of the block **LU** decompositions is different, the amount of floating point operations being saved is basically the same (the cost of only the forward and backward substitutions c_{fbs} is equal to the cost of the add and multiply operations c_{am} for $n \approx m \gg 1$). In Appendix F.1.1 we perform an analysis of floating point cost comparison of the block decomposition of the algorithm 9.

5.4 Numerical experiments

In this section we present numerical experiments with two goals. The first one is to verify the convergence of numerical method and their behaviour for a system of equations of type (1), (2), which is performed in Section 5.4.1. On the other hand, we are interested in verifying the practical efficiency of the DCB method for our class of PDAEs, and the asymptotic speedup values derived in Appendix F.1.1. For this efficiency analysis, we propose a class of problems which is composed of a typically small system of PDAEs, and our simulations typically take no longer than what a modern memory shared machine can handle. For that reason, we ran our experiments in a *Dell® R910* server machine, composed of 4 sockets with *Intel® Xeon® X7560* processors. The processors operate with a base frequency that ranges from 2.27 GHz to 2.67 GHz, and they have an L3 cache of 24 MB. Each of the processors have 8 physical cores, but 16 logical ones are available through the *multithreading* technology.

5.4.1 Numerical experiments - Physical model

We present a numerical example with RCD solver (see Section 5) considering the solution of the Riemann problem governed by equations of type (1) and

(2) with initial data

$$V(x,t=0)=\begin{cases}V_L \text{ if } x<0,\\ V_R \text{ if } x>0.\end{cases} \tag{88}$$

We apply the Upwind numerical scheme in a particular compositional model

$$s_t+(uf(s))_x=0, \tag{89}$$

$$\phi(sc_1+(1-s)c_2)_t+(uc_1f(s)+uc_2(1-f(s)))_x=0, \tag{90}$$

$$c_1-Kc_2^2=0. \tag{91}$$

where u, ϕ, K are constants and the unknowns are s, c_1 and c_2, representing the water saturation, the concentration of solvent in the water phase and concentration of solvent in the oil phase.

The model in (89), (90), (91) simulates the transport of a solvent in porous media with carbonated water and oil. The solvent appears both in the aqueous and oil phases, where the relationship (91) determines the proportion in which it distributes between phases. Here, the function f is given in [1] which is similar to $f(s)=s^2/(s^2+\nu(1-s)^2)$, where ν is a constant (Corey core Model, see[6]). RCD solver was used successful for the numerical simulations in [3].

Denoting $V=(s,c_1,c_2)$, the accumulation and flux functions are given by $G(V)=(sc_1,sc_1+(1-s)c_2)$, $F(V)=(uf,uc_1f+uc_2(1-f))$. The algebraic restriction is given by $H(V)=c_1-Kc_2^2$.

We use relaxation techniques to incorporate the algebraic restriction to the system (89), (90), (91) with the differential equation

$$\tau(c_1-Kc_2^2)_t=-(c_1-Kc_2^2). \tag{92}$$

Notice that when $\tau=0$, formally, the system of equations (89), (90) and (92) reduces to

$$\phi s_t+(uf(s))_x=0. \tag{93}$$

$$\phi(sKc_2^2+(1-s)c_2)_t+(Kuc_2^2f(s)+uc_2(1-f(s)))_x=0. \tag{94}$$

where the unknowns variables are s, c_2.

We verify numerically that when $\tau\to 0$ or t increasing the solution of equation (89), (90), (92) tends to the solution of system (93), (94). To do so, we solve numerically the Riemann data case I for initial data $V=(s,c_1,c_2)$ given by

$$V(x,t=0)=\begin{cases}(1,2.07,1.08) & \text{if } x<0,\\ (0.15,1.08,0.71) & \text{if } x>0.\end{cases} \tag{95}$$

for system (89), (90) and (92) and the case II with initial data given by

$$V(x,t=0)=\begin{cases}(1,1.083) & \text{if } x<0,\\ (0.15,0.71) & \text{if } x>0.\end{cases} \tag{96}$$

for system (93), (94) with $V=(s,c_2)$.

Since the Riemann solution describes the behavior of solution for long times, we choose several times and stop the simulation when no significant changes are observed in the solution profile.

We verify that in both cases I and II the solution match well. Figures 2 and 3 show the profiles of saturation (s) and concentration solution (c_1) for different times. In simulation we take $\delta x = 0.008$, with $x \in (0, 1.6)$, $\delta t = 0.2$ and $t \in (0, 18000)$. In the Figure 3, the solution is shown in the phase space (s, c_1). The Riemann solution consists of a rarefaction wave connected with a solvent shock wave with slower speed than the saturation shock.

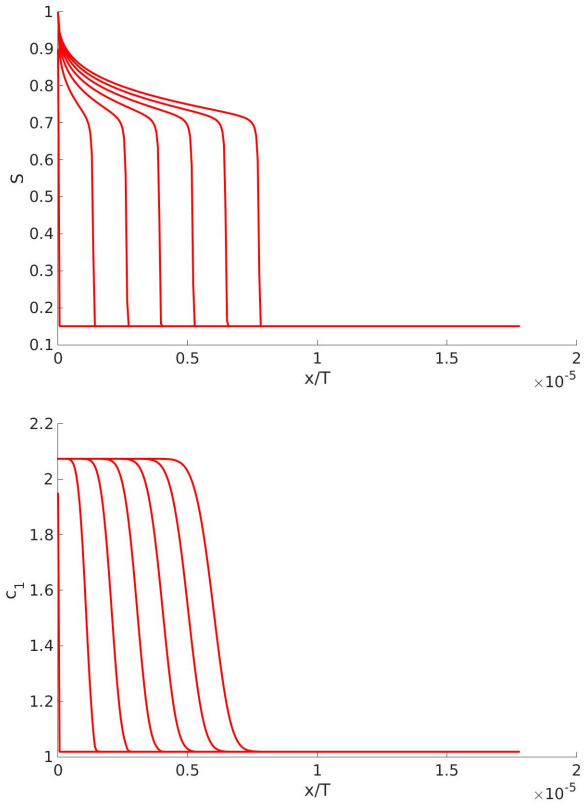

Fig. 2 a).The saturation profile from the RCD simulation at different times. b). The profile of the concentration of the solvent in the aqueous phase (c_1). Here x represents the space and T the final time.

5.4.2 Speedups for a Synthetic Physics

We discuss now the speedup values we observed through a few tests performed in the above mentioned shared memory machine. To obtain non-uniform block-tridiagonal matrices of any desired size and block configuration, we have de-

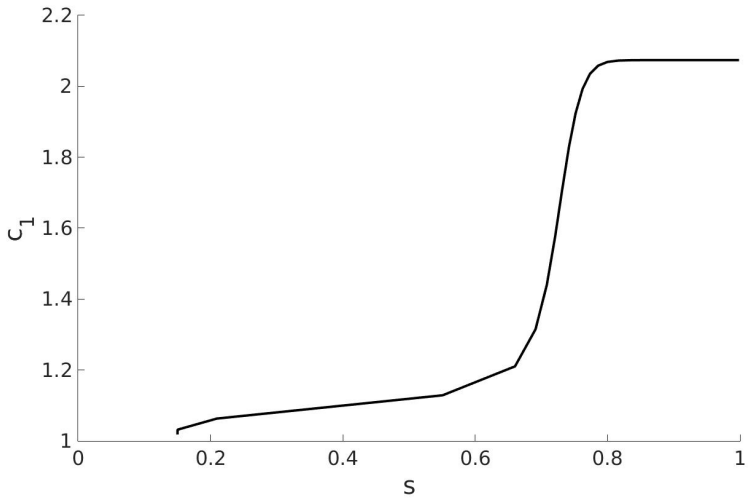

Fig. 3 Solution in the phase space. Saturation (s) and concentratation of first solvent (c_1)

veloped a simple synthetic physics given by the PDAE

$$\partial_t v_i + \partial_x \sum_j a_{i,j} \frac{v_j^2}{2} = 0, \tag{97}$$

$$v_k - \sum_j b_{(k-m),j} \frac{v_j^2}{2} = 0, \tag{98}$$

with boundary conditions

$$v_i(-1,t) = 1, \quad v_i(1,t) = 0, \tag{99}$$

and initial conditions

$$v_i(x,0) = \begin{cases} 1, \, x < 0 \\ 0, \, x \geq 0 \end{cases}, \tag{100}$$

where $i = 1, \ldots, m$, $j = 1, \ldots, m$, and $k = m+1, \ldots, n$. The matrices $a_{i,j}$, $b_{i,j}$ are α-matrices $c_{i,j}(\alpha)$ given by $c_{i,j}(\alpha) = 1$ if $i = j$, $c_{i,j}(\alpha) = \alpha$ otherwise. Notice that the constraint variables v_k do not interfere with the PDE, although their relations with v_i are nonlinear. The solutions are shocks traveling with speed $1 + (m-1)\alpha$ for the first m unknowns, and $p = n - m$ more linear combinations of those shocks for the remaining unknowns. In all the tests in this section we have adopted $a_{i,j} = c_{i,j}(0.001)$, $b_{i,j} = c_{i,j}(0.1)$.

The simulations were made using scheme (65), (66) with $\alpha_1 = \alpha_2 = 1/2$, which corresponds to the Crank-Nicholson scheme for our PDAE (97), (98). Although the scheme is unconditionally stable [45], we chose time step sizes respecting the CFL condition to maintain accuracy. Because the Crank-Nicholson scheme is dispersive, we have added to the right hand side of the

PDE (97) a stabilization term $\epsilon \frac{\partial^2 v_i}{\partial x^2}$, where the small and positive stabilization constant ϵ was chosen to be equal to the space step size.

To estimate the time taken to solve a non-uniform block-tridiagonal linear system derived from the Newton method applied to the discretization of (97), (98) in our shared memory machine, we have collected the times during the first 100 time steps of a simulation. The time needed to solve the linear system is then approximated by the mean value of the collected times. The exact number of linear systems solved is however larger than 100, since the Newton method typically takes more than one iteration (we have observed that more iterations are needed if we have more restrictions).

In what follows next, we will refer to sequential algorithm to be the top-to-bottom block **LU** decomposition algorithms 1-2. We will refer to parallel algorithm to be the DCB algorithm 9 with $\rho = 64$ partitions. We have implemented the step 1 of algorithm 9 in two different ways: when $p = 0$ (no algebraic constraints), the left-to-right block **LU** decomposition method (73) is applied to solve the linear systems using an algorithm that performs both the decomposition and the first of the back substitutions at the same time. This technique allows for a reduced number of memory accesses, since the blocks A_i, C_i are fetched from memory only once. When $p > 0$ (at least one algebraic constraint), the sequential algorithm is applied to solve the linear system.

Behavior for different linear system sizes: In our first test we attempted to determine how the absolute speedup (time taken to execute the sequential version divided by the time taken to execute the parallel version of an algorithm) behaves as the space mesh size increases (size of the linear system increases). In Figure 4 we see the absolute speedup obtained when the number of PDEs varies and the number of algebraic constraints is 0 ($n = m > 0$, $p = n - m = 0$). The best absolute speedup is reached when there are only two PDEs ($n = m = 2$) and the linear system size is between 10^5, 10^6. For a larger number of PDEs, the maximum speedup is reached for a larger linear system size. For this specific block structure, it is evident that the DCB method is better for a small number of PDEs, when the maximum absolute speedup is about 25. For a very large number of PDEs, which in the figure is exemplified by $n = m = 32$, $n = m = 64$, the maximum absolute speedup observed does not go beyond 12. As we have mentioned before, this behavior may be explained by the fact that the cache memory is used more efficiently when the blocks are small, so that the memory used in the block computations can be fetched only once, avoiding cache misses. Another explanation is that for small blocks our shared memory machine can better use the *hyperthreading* feature, since a lesser amount of calculations is necessary for the same amount of memory.

The absolute speedup observed for different space mesh sizes and different number of PDEs with only one algebraic constraint is showed in Figure 5. In this situation, we have calculated in Appendix F.1.1 that the asymptotic speedup obtained when the linear system is large enough is given by $7\rho/22$, as long as only floating point operations are considered. If we consider the

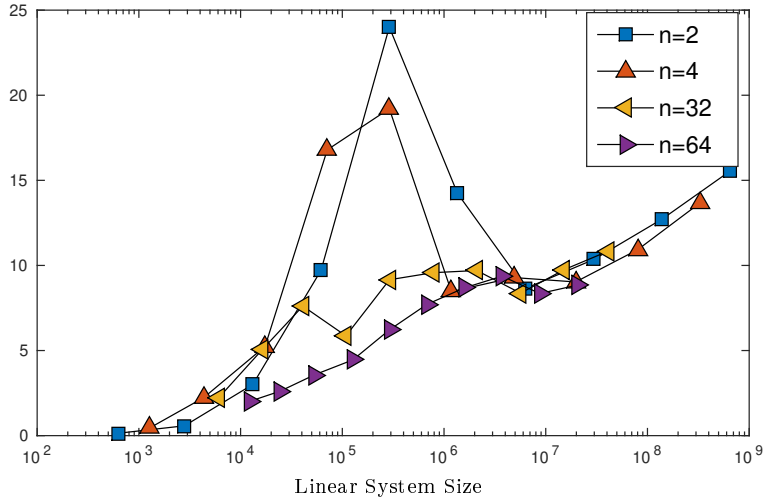

Fig. 4 Absolute speedup obtained for different space meshes, different number of PDEs and no algebraic constraints ($p = 0$). We have used the maximum number of threads 64.

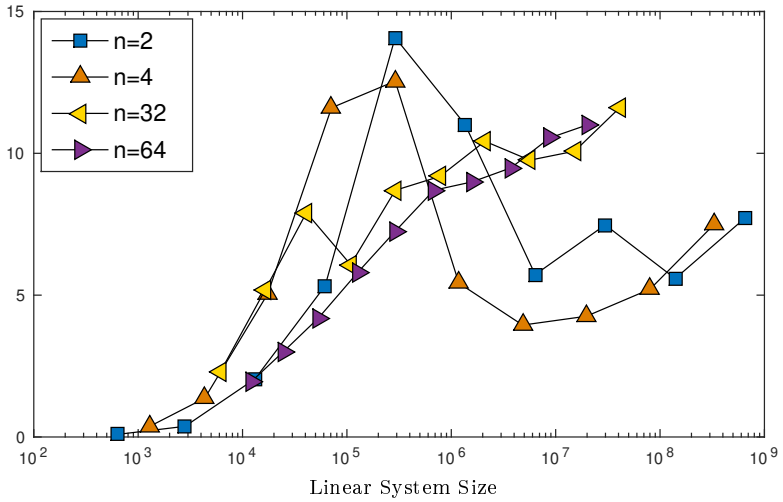

Fig. 5 Absolute speedup obtained for different space meshes, different number of PDEs and 1 algebraic constraints ($p = 1$). We have used the maximum number of threads 64.

number of logical cores of our shared memory machine, this number translates to $7 \times 32/22 \approx 10.2$. Figure 5 shows that for a small number of PDEs, the DCB method can be faster than the value predicted by the theory, reaching an absolute speedup of about 14 when $n = 2$, $n = 4$. However, for a larger system of PDAEs, we observed absolute speedups that reach at most 10.2, as predicted in the theory.

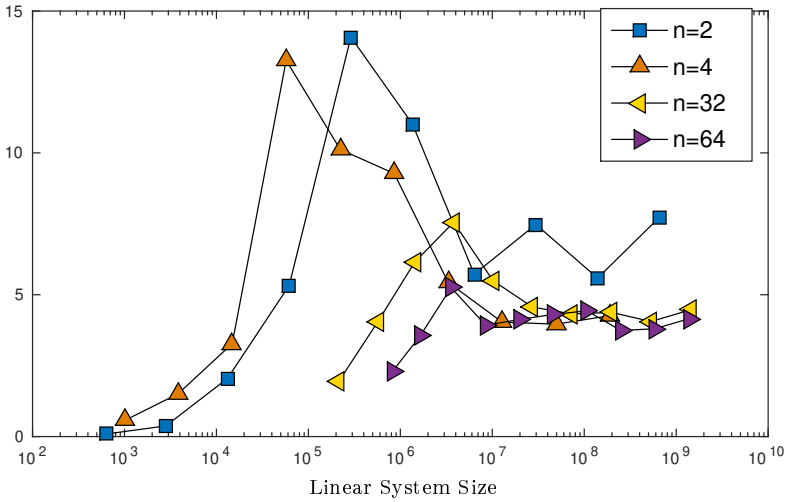

Fig. 6 Absolute speedup obtained for different space meshes, different number of algebraic constraints and one PDE ($p = n - 1$). We have used the maximum number of threads 64.

When the number of algebraic constraints is very large compared to the number of PDEs, we have calculated that the asymptotic speedup obtained when the linear system is large enough is $\rho/7$, which in our case equals to 4.57 if we consider 1e physical cores and to 9.14 if we consider all the logical ones. In Figure 6 we see that for a system of PDAEs of size 2 or 4, the algebraic constraints are not large enough, as the maximum absolute speedup still ranges from 13 to 14. However, when $n = 32$ or $n = 64$, although the maximum absolute speedup can still be larger than predicted by the theory, it tends to stabilize to the predicted absolute speedup for a large enough linear system size.

Small problems: For small problems, the divide and conquer method is known to outperform the cyclic reduction method [21], since the parallelization is very effective even though it requires more memory. By small problems we mean that the space mesh size is as big as to take long enough to solve a linear system, so that the essential tasks running in the system do not interfere much with the time spent solving the problem. The space mesh size was chosen so that the times collected during a linear system solving with the sequential algorithm have a standard deviation at least 100 times smaller than the mean value. In Figure 7 we can see a typical histogram of the collected times.

The calculation of the absolute speedup $\varsigma_a(\rho, M, n, p)$ in solving a non-uniform block-tridiagonal linear system using the DCB method rather than the top-to-bottom block **LU** decomposition method is given by the ratio $\varsigma_a(M, n, p) = t^0(M, n, p)/t^i(\rho, M, n, p)$, where $t^0(M, n, p)$ is the average time spent to solve using the sequential algorithm, while $t^i(\rho, M, n, p)$ is the average time spent to solve using the parallel algorithm for $j > 0$. Rewriting the absolute speedup in terms of the relative speedup $\varsigma_r(\rho, M, n, p) =$

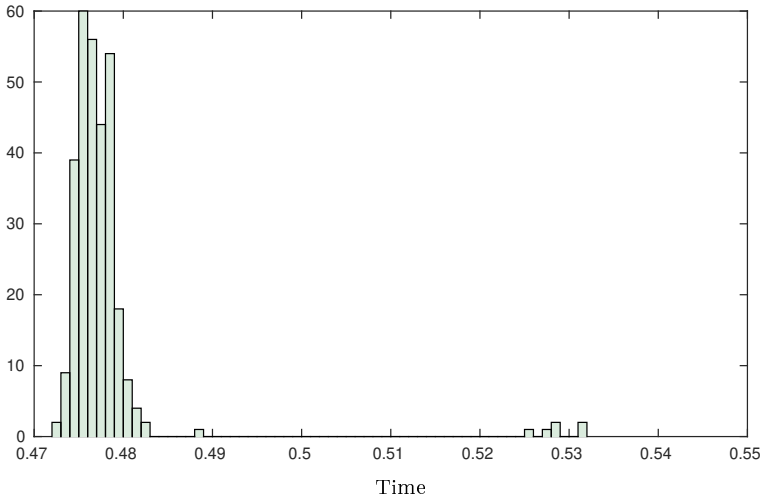

Fig. 7 Histogram of the time spent to solve the linear systems necessary to compute the numerical solution to the PDAE (97), (98) with $n = 16$, $m = 15$, $p = 1$ during the first 100 time steps of a simulation. Algorithms 1-2 were used to solve the linear systems.

$t^1(\rho, M, n, p)/t^i(\rho, M, n, p)$ gives $\varsigma_a(\rho, M, n, p) = (t^0(M, n, p)/t^1(\rho, M, n, p))\varsigma_r(\rho, M, n, p)$. Here ρ is the number of partitions used in the DCB method, M is the size of the linear system, n is the size of the blocks in the diagonal, $m = n - p$ is the size of the blocks in the other two diagonals of the non-uniform block-tridiagonal matrix, and i is the number of processing units.

If we consider the cost of only floating point operations, the ratio satisfies $(t^0(M, n, p)/t^1(\rho, M, n, p))$ is bounded by the inverse of the *redundancy*. Redundancy is a measure given by the total cost of the parallel method divided by the cost of the sequential method, and it tells how many more calculations are necessary to replace the sequential method by the parallel method. The calculation of the redundancy is similar to the calculation of the asymptotic speedup presented in Appendix F.1.1, and for big linear systems it can be approximated by ρ times the inverse of the asymptotic speedup. That is, for $M \gg 1$, $n \gg 1$, $p = 0$ the DCB takes approximately 22/7 times more calculations than the block **LU** decomposition method, while for $M \gg 1$, $n \gg 1$, $p \approx n$ the DCB method is approximately 7 times more costly than the block **LU** decomposition method. If the *DCB* method is perfectly parallelizable, then $\varsigma_r(\rho, M, n, p) = \rho$ and the absolute speedup approaches the asymptotic speedup obtained in Appendix F.1.1 (considering that the number of processing units is equal to the number of partitions ρ). However, communication between the several processing units as well as the inherent cost of the distribution of work among them usually makes the relative speedup be smaller than the maximum theoretical one, and as a result, the absolute speedup is smaller than the asymptotic speedup.

$c_{i,j}$	Relative Speedup	Absolute Speedup
$c_{0,0}$	27.020	11.470
$c_{1,0}$	0.9625	0.0137
$c_{0,1}$	-0.3564	-0.5223
$c_{2,0}$	-0.1102	0.0053
$c_{1,1}$	0.0416	0.0200
$c_{0,2}$	0.0211	0.0115
$c_{3,0}$	0.0024	-0.0006
$c_{2,1}$	-0.0009	0.0005
$c_{1,2}$	-0.0012	-0.0012
$c_{0,3}$	0.0003	0.0002

Table 1 Values of the coefficients $c_{i,j}$ of the speedup surface $z(n,p) = c_{0,0} + c_{1,0}n + c_{0,1}p + c_{2,0}n^2 + c_{1,1}np + {}_{0,2}p^2 + c_{3,0}n^3 + c_{2,1}n^2p + c_{1,2}np^2 + c_{0,3}p^3$. The coefficients were determined using the least squares method. The Algorithm 9 was used in the parallel version, and the Algorithms 1-2 were used in the sequential version.

In Figure 8 we can see the relative speedup surface $\varsigma_r(64, M, n, p)$ when using the maximum number of threads 64 in our shared memory machine. The Figure 9 shows the absolute speedup we observed with 64 threads. We recall that, although we have an allowed maximum number of threads of 64, the maximum number of processing units (cores), is 32. In Table 1 we show the coefficients of the surface constructed using the least squares method for a third degree polynomial in both n, p direction. Although there are variations in the data we collected with the fitted surfaces (which we attribute to cache effects), we can say that the surface is a good approximation to the data. Using the fitted surfaces, we see that the relative speedup ranges from 27.875 to 21.022, while the absolute speedup ranges from 11.488 to 8.8810. The observed absolute speedup is in this case bigger than the relative speedup divided by the estimated redundancy 22/7, and the reason may be explained by the fact that either the linear system or the block sizes are too small to qualify the asymptotic case.

6 Conclusions and Remarks

In this work we discuss a formalism for the class of PDAEs (1), (2) and we study the connections between their solutions with two another formulations: (1), (6) and (1), (7).

We prove that the solution of (1), (6) for initial data on the equilibrium surface $\mathcal{S}$ remains on the equilibrium surface $\mathcal{S}$. Moreover, for Riemann data, this solution is the same solution obtained for the PDAEs (1), (2).

Using relaxation formulation (1), (7), we prove that the solution tends to equilibrium $\mathcal{S}$ if initial data are not on $\mathcal{S}$. If the initial data are on $\mathcal{S}$ the solution remains on $\mathcal{S}$.

We also develop a faster and reliable numerical method that is naturally parallelizable. We perform several numerical experiments to verify that the numerical method agrees with the theory. Moreover, we also study the speedup of the algorithms.

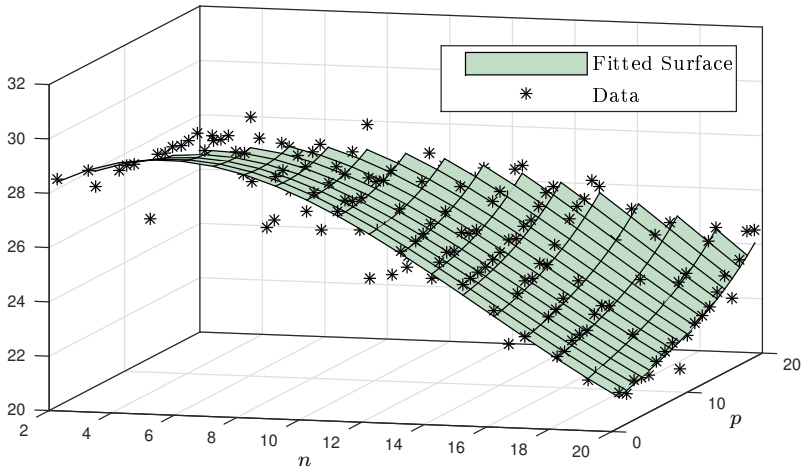

Fig. 8 The relative speedup surface obtained when varying n, p for small problems. The results were collected on a shared memory environment which supports at most 64 threads. Algorithm 9 was used to solve the linear systems with $\rho = 64$.

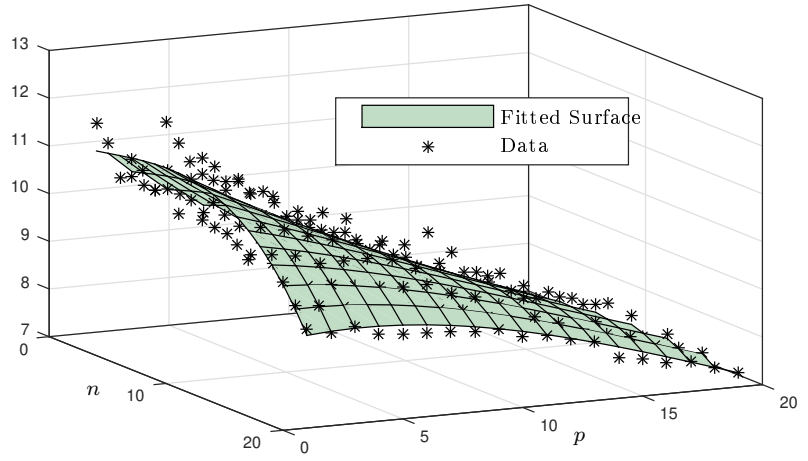

Fig. 9 The absolute speedup surface obtained when varying n, p for small problems. The results were collected on a shared memory environment which supports at most 64 threads. Algorithm 9 was used to solve in parallel with $\rho = 64$ and Algorithms 1-2 were used to solve in sequential.

A Parameterizing the surface $\mathcal{S}$

We assume that $H(U)$ is parameterized by a subgroup of variables $V = (U_1, \cdots, U_m)$ of original set $U = (U_1, \cdots, U_n)$. The remain variables $W = (U_{m+1}, \cdots, U_n)$ are given as a function of V. As discussed in Introduction, if for a U^*, we have that $det(\partial_W H(U^*)) \neq 0$, in a state then there is a diffeomorphism in a neighborhood of U^* and we can write W as function of V and we rewrite system (1), (2) as (5). This form motivates the definition for hyperbolicity (see [39] for a most complete study about this theme)

Definition 5 We say that PDAE (1), (2) is formally hyperbolic if (5) is hyperbolic, i.e., if the system (16):

$$\tilde{A}\tilde{r} = \lambda \tilde{B}\tilde{r} \quad \longrightarrow \quad det(\tilde{B} - \lambda \tilde{A}) = 0$$

admits m real linearly independent eigenvectors $\tilde{r} = (\tilde{r}_1, \cdots, \tilde{r}_m)$ and m real eigenvalues $\lambda := \lambda(W)$ (accounting the multiplicities if necessary). Here $\tilde{A}$, $\tilde{B}$ are jacobian of $\tilde{F} = F(V, W(V))$, $\tilde{G} = G(V, W(V))$, with respect to W, respectively.

The eigenvectors satisfying (16) form a local basis for the variable V, thus the eigenvectors supplemented with the condition $W = W(V)$, satisfying $H(V, W(V)) = 0$, form a local basis for $\mathcal{S}$.

We have m eigenpairs denoted as $(\lambda_i, \tilde{r}_i)$ for $i = 1, \cdots, m$, which eigenvalues we can order as:

$$\lambda_1(V) \leq \lambda_2(V) \leq \cdots \leq \lambda_m(V). \tag{101}$$

The integral curve of family i is given by $V = V(\xi)$:

$$d_\xi V_j = (\tilde{r}_i)_j, \quad \text{for} \quad j = 1, 2, \cdots, m, \text{ on this curve} \quad W = W(V(\xi)), \tag{102}$$

where $(\tilde{r}_i)_j$ is the j-th component of vector $\tilde{r}_i$. To obtain the integral curve it is necessary to set the correct direction of eigenvector. The variable W on the integral curve is obtained as $W = W(V(\xi))$. For numerical purposes, sometimes, is valid to obtain the integral equation for each component of W by differentiating $W_i = W_i(V)$, for $i = 1, \cdots, p$ as:

$$\partial_\xi W_i = \sum_{l=1}^{m} \left(\partial_{V_l} W_i\right) \partial_\xi V_l = \sum_{l=1}^{m} \left(\partial_{V_l} W_i\right) (\tilde{r}_i)_l = \nabla_V W_i \cdot \tilde{r}_i. \tag{103}$$

Here ∇_V is the gradient with respect to variable V.

The k-rarefaction curve is the segment of k-integral curve for which the characteristic speed λ_k is an increasing function, i.e., the direction for which

$$\nabla_V \lambda \cdot \tilde{r}_j > 0. \tag{104}$$

A.1 Hugoniot-Locus and shock curves

Another important wave appears when we look for discontinuities of the solution in the (x, t)-plane. For non-linear hyperbolic systems is well known that these discontinuities appear in solutions for any initial data. These discontinuities should satisfy an important identity, the so called Rankine-Hugoniot (RH) condition. This condition expresses conservation of physical and chemical quantities in the system. For system (5) is written as:

$$v^s(\tilde{G}(V^+) - \tilde{G}(V^-)) = \tilde{F}(V^+) - \tilde{F}(V^-). \tag{105}$$

Moreover, $W = W(V)$, i.e, $W^\pm = W(V^\pm)$ on the surface $\mathcal{S}$. For a fixed V^-, the states V^+ satisfying (105) form the *Rankine-Hugoniot locus* through V^-, which we denote as $\mathcal{RH}(V^-)$.

It is well known that weak solutions are, in general, non-unique, see [13, 41, 42]. To overcome this non-uniqueness it is necessary to define some entropy criterion. A weak solution satisfying a entropy criterion is called *entropy solution*. Since the approach discussed in this

section we can reduce PDAE system (1), (2) into a system to the form (5), we can use as entropy criterion Lax's or Liu's criteria (depending on the characteristic of each field) or we can use the traveling wave or viscous profile to select physical shocks satisfying (105). In Section 2.1.3 we discuss the existence of traveling wave. But, since the theory for this class of equations is well known, we recommend [13,41,42].

In other hand, in many different applications it is impossible to obtain a subset of U to parameterize $\mathcal{S}$. In the next Sections we draw some formalism to deal with the system of PDAEs (1), (2) where we can not have a explicit parametrization to $\mathcal{S}$.

B Entropy for the system (1), (6).

Definition 6 Let $U = U(x,t)$ be a weak solution of the hyperbolic PDE system (1), (6). Then we say that U satisfies an entropy inequality if there exists functions $\mathcal{U}, \mathcal{U}_1, \mathcal{F} : \mathbb{R}^n \longrightarrow \mathbb{R}$ such that $\mathcal{U}$, $\mathcal{U}_1$ are convex satisfying

$$D\mathcal{U} = D\mathcal{U}_1 \begin{pmatrix} B \\ E \end{pmatrix}, \quad D\mathcal{U}_1 \begin{pmatrix} A \\ 0 \end{pmatrix} = \mathcal{F} \quad \text{and} \quad \partial_t \mathcal{U}(U) + \partial_x \mathcal{F}(U) \leq 0 \tag{106}$$

in the distributional sense. The pair $(\mathcal{U}, \mathcal{F})$ is called an entropy pair.

From this definition, we can prove the following result:

Proposition 7 *Let us assume that there are functions $\mathcal{U}$, $\mathcal{U}_1$, $\mathcal{F}$ satisfying Definition 6. Let $U_\epsilon \in H^{1,2}(\mathbb{R} \times \mathbb{R}^+, \mathbb{R}^m)$ be a solution of*

$$\partial_t \begin{pmatrix} G(U_\epsilon) \\ H(U_\epsilon) \end{pmatrix} + \partial_x \begin{pmatrix} F(U_\epsilon) \\ 0 \end{pmatrix} = \epsilon I \partial_x^2 U_\epsilon \quad \mathbb{R} \times \mathbb{R}^+. \tag{107}$$

Additionally we assume that $\mathcal{U}, \mathcal{U}_1, \mathcal{F}, G, H, F, U_\epsilon$ are sufficiently smooth, and $U_\epsilon \longrightarrow V$ (when $\epsilon \longrightarrow 0$) in $\mathbb{R} \times \mathbb{R}^+$, thus V is solution of PDEs (1), (6) in the distributional sense. Besides, we have that entropy inequality (106) is satisfied in the distributional sense.

Proof: First, we prove that $V(x,t)$ (that represents the limit when $\epsilon \longrightarrow 0$) is a weak solution of (1), (6). To to so, we multiply (107) by a test function φ and integrate by parts and we obtain:

$$\int\int_{\mathbb{Q}} \left(\partial_t \varphi(x,t) \begin{pmatrix} G(U_\epsilon) \\ H(U_\epsilon) \end{pmatrix} + \partial_x \varphi(x,t) \begin{pmatrix} F(U_\epsilon) \\ 0 \end{pmatrix} + \epsilon I U_\epsilon \partial_x^2 \varphi(x,t) \right) dxdt - \int\int_{\mathbb{Q}} \partial_t \varphi(x,t) \begin{pmatrix} G(U_\epsilon(x,0)) \\ H(U_\epsilon(x,0)) \end{pmatrix} dx = 0 \tag{108}$$

Taking $\epsilon \longrightarrow 0$, using the dominated convergence theorem and using that $H(U_\epsilon(x,0)) = 0$, we obtain the weak solution of (1), (2).

To prove that (106) is satisfied, we multiply (to right) (107) by $D\mathcal{U}_1$, performing the chain rule in (107) (here the solution of (107) are smooth functions) and using eq. (106), we obtain:

$$\partial_t \mathcal{U}(U_\epsilon) + \partial_x \mathcal{F}(U_\epsilon) = (\epsilon D\mathcal{U}_1)\, \partial_x^2 U_\epsilon \quad \longrightarrow \quad \partial_t \mathcal{U}(U_\epsilon) + \partial_x \mathcal{F}(U_\epsilon) = \epsilon \partial_x^2 \mathcal{U}_1(U_\epsilon) - D^2\mathcal{U}_1 \left(\partial_x U_\epsilon\right)^2 \tag{109}$$

Notice that $D^2\mathcal{U}_1 \left(\partial_x U_\epsilon\right)^2 > 0$, thus (109) satisfies

$$\partial_t \mathcal{U}(U_\epsilon) + \partial_x \mathcal{F}(U_\epsilon) < \epsilon \partial_x^2 \mathcal{U}_1(U_\epsilon) \tag{110}$$

Multiplying (110) by a test function $\varphi(x,t)$ (with compact support for $t > 0$), performing integration by parts, we obtain:

$$\int\int_{\mathbb{Q}} \partial_t \varphi(x,t) \mathcal{U}(U_\epsilon) + \partial_x \varphi(x,t) \mathcal{F}(U_\epsilon) dxdt + \epsilon \partial_x^2 \varphi(x,t) \mathcal{U}_1(U_\epsilon) > 0 \tag{111}$$

Taking the limit of $\epsilon \longrightarrow 0$ in Eq. (111) and using the theorem of dominated convergence, we obtain that (111) satisfies (106) in the distributional sense and the result is proved. □

C Finite Difference Methods for PDAEs - The linear case

In this Section we are interested in obtaining some results for the linear PDAEs. There are many numerical methods applied for PDEs of form (5), that we can cite, *finite differences*, see [24,30,45], *finite elements*, see [44], *C finite volumes*, see [16,24,31], *spectral techniques*, see [17,19], however, here we focus only on finite difference methods. We also extend the Lax-Richtmyer theorem, see [45], is a very important theorem stating that *consistence and stability* for linear systems of partial differential equations implies in *convergence*. For non-linear system of partial differential equations this theorem is not valid, however, proposition of numerical methods for nonlinear systems should satisfy the same condition for linear ones. Here, we extend the Lax-Richtmyer theorem, besides some sufficient conditions for stability, and we prove the convergence of extension of classical numerical methods for LPDAE. Here the term linear means that accumulation, flux and algebraic equations are linear, i.e.,

$$G(U) = BU, \qquad F(U) = AU \qquad \text{and} \qquad H(U) = CU - \alpha, \tag{112}$$

where B, A are $m \times n$ matrices, C is a $p \times n$ matrix and α column p-vector. We also extend the numerical methods for general systems of PDAEs.

C.1 Finite Difference Methods for LPDAEs

We can consider the most used explicit (first and second order) and the implicit Crank-Nicholson schemes, see [45], and to adapt these schemes for LPDAEs system. The basic idea here is to utilize these numerical schemes for linear PDEs and to apply these for (2). For the algebraic equation, given by restriction (2), we assume that this equation is satisfied for any U_i^γ for all $(x_\gamma, t^i) \in \mathcal{D}_d$.

Here, we admit that we know the states U_i^θ are known for all $i \in \mathbb{Z}$ and $\theta \leq \gamma$. Then the numerical schemes will be condition to obtain the solution for the time level $\gamma + 1$. The numerical schemes will be written for the (1), (2) (and (112)) system in the matricial form as:

$$\mathcal{L}(S^+, S^-)U^{\gamma+1} = \mathcal{Q}(S^+, S^-)U^\gamma + \Gamma, \tag{113}$$

where are using that $S^+U_j = U_{j+1}$, $S^-U_j = U_{j-1}$. Here $\mathcal{L}$, $\mathcal{Q}$ are polynomials of S^+, S^- and the matrices A, B, C.

The classical methods for hyperbolic systems of equations are extended for PDAE as:
Upwind: Forward-time/forward-space

$$B\left(\frac{U_i^{\gamma+1} - U_i^\gamma}{\Delta t}\right) + A\left(\frac{U_{i+1}^\gamma - U_i^\gamma}{\Delta x}\right) = 0, \tag{114}$$

$$CU_i^{\gamma+1} = \vartheta. \tag{115}$$

We can rewrite (114), (115) in the form (113) as:

$$JU_i^{\gamma+1} = KU_i^\gamma - \alpha L\left(U_{i+1}^\gamma - U_i^\gamma\right) + \Gamma, \tag{116}$$

where matrices J, K, L, Γ are obtained are given by

$$J = \begin{pmatrix} B \\ C \end{pmatrix}, \quad K = \begin{pmatrix} B \\ 0 \end{pmatrix}, \quad L = \begin{pmatrix} A \\ 0 \end{pmatrix}; \quad M = \begin{pmatrix} 0 \\ C \end{pmatrix}; \quad \text{and the vector } \Gamma = \begin{pmatrix} \hat{0} \\ \alpha \end{pmatrix}. \tag{117}$$

For (116), matrices $\mathcal{L}$, $\mathcal{Q}$ are

$$\mathcal{L} = J \quad \text{and} \quad \mathcal{Q} = (K - \alpha L) + \alpha L S^+.$$

Since the states U_i^γ are known, using the matrices and vector defined in (117), and since matrix J admits inverse, the numerical system (114), (115) is recasted as:

$$U_i^{\gamma+1} = J^{-1}\left(KU_i^\gamma - \alpha L\left(U_{i+1}^\gamma - U_i^\gamma\right) + \Gamma\right). \tag{118}$$

Upwind: Forward-time/backward-space. The backward-space is written in the form (113) with $\mathcal{L}$, $\mathcal{Q}$ given by:

$$\mathcal{L} = J \quad \text{and} \quad \mathcal{Q} = (K - \alpha L) + \alpha L S^{-}.$$

Then the numerical method is written in the matricial form as:

$$U_i^{\gamma+1} = J^{-1}\left(KU_i^{\gamma} - \alpha L\left(U_i^{\gamma} - U_{i-1}^{\gamma}\right) + \Gamma\right). \tag{119}$$

Lax-Friedrichs scheme. The matricial form for Lax-Friedrichs scheme for (1), (2) (and (112)) is extended in the form (113) with $\mathcal{L}$, $\mathcal{Q}$ given by:

$$\mathcal{L} = J \quad \text{and} \quad \mathcal{Q} = \left(\frac{K}{2} - \alpha\frac{L}{2}\right)S^{+} + \left(\frac{K}{2} + \alpha\frac{L}{2}\right)S^{-},$$

which after some manipulation reduces to:

$$U_i^{\gamma+1} = J^{-1}\left(\frac{K}{2}\left(U_{i+1}^{\gamma} + U_{i-1}^{\gamma}\right) - \frac{\alpha}{2}L\left(U_{i+1}^{\gamma} - U_{i-1}^{\gamma}\right) + \Gamma\right). \tag{120}$$

Lax-Wendroff scheme. The Lax-Wendroff scheme for PDEs is second order of accuracy, see [24,45]. For (1), (2) (and (112)), the Lax-Wendroff scheme is written in the matricial form (113) with $\mathcal{L}$, $\mathcal{Q}$ given by:

$$\mathcal{L} = J \quad \text{and} \quad \mathcal{Q} = \left(\frac{L^2\alpha}{2\Delta x} - \frac{\alpha}{2}L\right)S^{+} + K - 2\frac{L^2\alpha}{2\Delta x} + \left(\frac{L^2\alpha}{2\Delta x} + \frac{\alpha}{2}L\right)S^{-},$$

which after some manipulation, is written as:

$$U_i^{\gamma+1} = J^{-1}\left(KU_i^{\gamma} - \frac{\alpha}{2}L\left(U_{i+1}^{\gamma} - U_{i-1}^{\gamma}\right) + \frac{L^2\alpha}{2\Delta x}\left(U_{i+1}^{\gamma} - 2U_i^{\gamma} + U_{i-1}^{\gamma}\right) + \Gamma\right). \tag{121}$$

Crank-Nicholson scheme. This is the most used implicit numerical scheme for PDEs. It is based in means performed in the space and time. For (1), (2) (and (112)), the Crank-Nicholson, in the matricial form, is written as:

$$B\left(\frac{U_i^{\gamma+1} - U_i^{\gamma}}{\Delta t}\right) + A\left(\frac{U_{i+1}^{\gamma+1} - U_{i-1}^{\gamma+1} + U_{i+1}^{\gamma} - U_{i-1}^{\gamma}}{4\Delta x}\right) = 0, \tag{122}$$

$$CU_i^{\gamma+1} = \vartheta. \tag{123}$$

Passing the known states to the right side and letting the unknowns on the right side, the system (122), (123) is written as:

$$A\frac{\alpha}{4}U_{i+1}^{\gamma+1} + BU_i^{\gamma+1} - A\frac{\alpha}{4}U_{i-1}^{\gamma+1} = -A\frac{\alpha}{4}U_{i+1}^{\gamma} + BU_i^{\gamma} + A\frac{\alpha}{4}U_{i-1}^{\gamma}, \tag{124}$$

$$CU_i^{\gamma+1} = \vartheta. \tag{125}$$

The Crank-Nicholson scheme is also written in matricial form (113) with $\mathcal{L}$, $\mathcal{Q}$ given by:

$$\mathcal{L} = -\frac{\alpha}{4}LS^{-} + J + \frac{\alpha}{4}LS^{+} \quad \text{and} \quad \mathcal{Q} = \frac{\alpha}{4}LS^{-} + K - \frac{\alpha}{4}LS^{+}.$$

The Crank-Nicholson scheme is applied for limited domains with left and right boundary conditions. If we set the spatial indices $i = \{0, 1, \cdots, l\}$ and denote the right side of (122) as b_i and assuming that the boundary states U_0, U_n are known, called Dirichlet condition (other kinds of boundary conditions are possible as Newman or Robin conditions), the system (1), (2) (and (112)) is written as the tridiagonal matrix form:

$$\begin{pmatrix} J & -\frac{\alpha}{4}K & 0 & 0 & \cdots \\ \frac{\alpha}{4}K & J & -\frac{\alpha}{4}K & 0 & \cdots \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & \cdots & \frac{\alpha}{4}K & J & -\frac{\alpha}{4}K \\ 0 & \cdots & 0 & \frac{\alpha}{4}K & J \end{pmatrix} \begin{pmatrix} U_1^{\gamma+1} \\ U_2^{\gamma+1} \\ \\ \vdots \\ \\ U_{n-1}^{\gamma+1} \end{pmatrix} = \begin{pmatrix} b_1 + K\frac{\alpha}{4}U_0 \\ \\ b_2 \\ \vdots \\ b_{n-2} \\ b_{n-1} - K\frac{\alpha}{4}U_n \end{pmatrix} \tag{126}$$

An important property that we are interested in the above numerical schemes is that for each level time the solution remains on the surface $\mathcal{S}$. This condition motivates the following definition:

Definition 7 A numerical scheme for a LPDAE is $\mathcal{S}$-persistent if states on $\mathcal{S}$ remains on $\mathcal{S}$ for the time evolution.

We can state the following proposition.

Proposition 8 *U_i^γ, obtained by any numerical schemes* (118), (121), *lies on $\mathcal{S}$, i.e., the numerical schemes* (118), (121) *are $\mathcal{S}$-persistent.*

Proof: Notice that since J^{-1} is the inverse of matrix J, then:

$$\begin{aligned} &JJ^{-1} = \begin{pmatrix}(B)_{m\times n}\\ (C)_{p\times n}\end{pmatrix} J^{-1} = (I)_{n\times n} \longrightarrow \text{ thus } KJ^{-1} = \begin{pmatrix}(I)_{m\times m} & (0)_{m\times p}\\ (0)_{p\times m} & (0)_{p\times p}\end{pmatrix} \\ &\text{and } MJ^{-1} = \begin{pmatrix}(0)_{m\times m} & (0)_{m\times p}\\ (0)_{p\times m} & (I)_{p\times p}\end{pmatrix} \end{aligned} \tag{127}$$

Notice also that $KU = ((BU)_{m\times 1}, (0)_{p\times 1})$, $LU = ((AU)_{m\times 1}, (0)_{p\times 1})$ and $L^2U = ((A^2U)_{m\times 1}, (0)_{p\times 1})$.

For any scheme (118), (121), multiplying both side by C on the right by C and using (127) and previous observation, we obtain $CU_i^{\gamma+1} = \alpha$, i.e., $U_i^{\gamma+1}$ is on the surface $\mathcal{S}$.

□

In section 5.4.1, we present some numerical experiments of the above schemes for different systems of equations. In the next section, we prove necessary conditions to guarantee the convergence of numerical methods, extending the fundamental theorem of Lax-Richtmyer.

To prove the convergence, it is useful to extend the function U_i^γ defined in the grid to all $\mathbb{R}\times\mathbb{R}$, we define for $0 \le \gamma$:

$$U(x,t) := U_i^\gamma, \quad \text{for } \gamma\Delta t \le t < (1+\gamma)\Delta t \text{ and } \left(i-\frac{1}{2}\right)\Delta x < x \le \left(i-\frac{1}{2}\right)\Delta x. \tag{128}$$

For a fixed time $t^\gamma = \Delta t\gamma$, we denote the function $U(x,t)$ only as $U^\gamma := U(x,t^\gamma)$.

C.2 Consistency, Stability and Convergence - Lax-Richtmyer theorem

The most important question arising when we propose numerical schemes is: *The solution U_i^γ obtained from numerical scheme converges to the solution $V(x,t)$ of LPDAEs when Δt, Δx tend to zero?*

Generically, to obtain the answer for this question is very hard. In 1956, Lax and Richtmyer, see [29, 45], proved a very important theorem for Linear systems of partial differential equations that utilizes the prove of two more simple conditions on the numerical schemes for linear PDES that are stability and consistence. The original theorem states, roughly speaking, that a consistent numerical scheme for linear PDEs is convergence if, only if, is stable.

For stability we mean a certain uniform boundedness on U_i^γ on $\mathcal{D}_d$ for any Δt, Δx and for consistence we mean that the numerical schemes approximates, locally, the time and spatial derivatives. There are many ways to define stability, however for our purposes we define stability as:

Definition 8 For a fixed time T, we say that the numerical scheme (113) is stable with respect to a norm $||\quad||$ if there exist constants $c_1(T)$, c_2, c_3 and τ satisfying:

$$||U^\gamma|| \le c_1(T)e^{c_2\gamma\Delta t/T}||u^0|| + c_3||\Gamma|| \quad \text{for all } 0 \le \Delta t < \tau \text{ and } \gamma \le \frac{T}{\Delta t}. \tag{129}$$

Besides stability we need to define consistence, i.e., how the numerical method approximates the solution

Definition 9 We say that the numerical scheme (113) is consistent of order (q,p) with respect to the norm $||\cdot||$ if we have for any smooth solution $V(x,t)$ of the differential equation (1), (2) (and (112))

$$||V(\cdot,t^{\gamma+1}) - (\mathcal{L}^{-1}\mathcal{Q}V(\cdot,t^{\gamma}) + \mathcal{L}^{-1}\Gamma)|| = \mathcal{O}(\Delta t^{q+1}) + \mathcal{O}(\Delta x^{p}\Delta t). \tag{130}$$

Moreover we assume that the $\mathcal{L}$, $\mathcal{Q}$ satisfy:

$$\mathcal{L}^{-1}\mathcal{Q}\mathcal{L}^{-1}\Gamma = 0. \tag{131}$$

Notice that the consistence condition guarantees that the scheme is $\mathcal{S}$-persistent.

Definition 10 The numerical scheme (113) is convergent of order (q,p) with respect to the norm $||\cdot||$ if

$$||v(\cdot,t^{n}) - u^{n}|| = \mathcal{O}(\Delta t^{q}) + \mathcal{O}(\Delta x^{p})\text{uniformly for all } n \in \mathbb{N}, \tag{132}$$

here v is the exactly solution of (1), (2) (and (112)).

Since we are able to prove the stability of numerical scheme using the norm of operators, it is useful to prove the following result:

Lemma 1 *A sufficient condition for stability of numerical scheme* (113) *if there exist constants* $c_1(T)$, c_2, c_3, τ *such that:*

$$||\mathcal{L}|| \leq c_3 \text{ and } ||\mathcal{L}^{-1}\mathcal{Q}||^{\gamma} \leq c_1(T)e^{c_2\gamma\Delta t/T} \quad \text{for all } 0 \leq \Delta t < \tau \text{ and } \gamma \leq \frac{T}{\Delta t}. \tag{133}$$

Proof: Since J is invertible, there exists constants k_1, k_2 such that $||J|| < k_1$, $||J^{-1}|| < k_2$. Moreover we have that:

$$\begin{aligned} ||U^{\gamma}|| = ||\mathcal{L}^{-1}\mathcal{Q}U^{\gamma-1} + \mathcal{L}^{-1}\Gamma|| &= ||\mathcal{L}^{-1}\mathcal{Q}(\mathcal{L}^{-1}\mathcal{Q}U^{\gamma-2} + \mathcal{L}^{-1}\Gamma) + \mathcal{L}^{-1}\Gamma|| \\ &= ||(\mathcal{L}^{-1}\mathcal{Q})^{2}U^{\gamma-2} + \mathcal{L}^{-1}\Gamma||. \end{aligned}$$

Here we have used the consistence conditions (131). Applying recursively and using (131) we obtain that:

$$||U^{\gamma}|| = ||(\mathcal{L}^{-1}\mathcal{Q})^{\gamma}U^{0} + \mathcal{L}^{-1}\Gamma||, \tag{134}$$

which give us:

$$||U^{\gamma}|| \leq ||(\mathcal{L}^{-1}\mathcal{Q})||^{\gamma}||U^{0}|| + ||\mathcal{L}^{-1}||||\Gamma||, \tag{135}$$

Taking $||(\mathcal{L}^{-1}\mathcal{Q})||^{\gamma}$, $||\mathcal{L}||$ satisfying (133), from stability definition Eq. (129) the result follows. □.

Remark 2 Notice that from Eq. (133) that we have

$$||\mathcal{L}^{-1}\mathcal{Q}||^{\gamma} \leq c_1(T)e^{c_2\gamma\Delta t/T} \longrightarrow ||\mathcal{L}^{-1}\mathcal{Q}|| \leq \sqrt[\gamma]{c_1(T)}e^{c_2\Delta t/T}, \tag{136}$$

taking $\Delta t \longrightarrow 0$ then, for a fixed T, we have that $\gamma \longrightarrow \infty$, from Eq. (136) we have that a sufficient condition for stability is that:

$$||\mathcal{L}^{-1}\mathcal{Q}|| \leq 1, \tag{137}$$

Remark 3 Let the partial differential system (1) (and (112)). The differential form for this system is given by:

$$\mathcal{L}_1(S^{+},S^{-})U^{\gamma+1} = \mathcal{Q}(S^{+},S^{-})U^{\gamma}. \tag{138}$$

Notice that $\mathcal{Q}(S^{+},S^{-})$ is the same that for the discretization of system (1), (2) (and (112)). If the $\mathcal{L}_1 = \mathcal{L}$ the numerical method for (1), (2) (and (112)) is stable if, only if, the numerical scheme for (1) is stable. Generically, for explicit numerical methods this condition is verified.

Now, we are able to prove the equivalence result connecting stability, consistence and convergence.

Proposition 9 *Let us assume that* (1), (2) *and* (113) *are linear in* $V(x,t)$, U, *that* (133) *is satisfied and that* (113) *is consistent of order* (q,p) *and* $||U^0 - V(\cdot,0)|| = \mathcal{O}(\Delta x^p)$. *Then* (113) *is convergent of order* (q,p), *i.e.,*

$$||U^\gamma - V(\cdot,t^\gamma)|| = \mathcal{O}(\Delta x^p) + \mathcal{O}(\Delta t^q) \text{ uniformly for all } \gamma \leq \frac{T}{\Delta t}. \tag{139}$$

Proof: Since the scheme is consistent of order (q,p), from (130), the exact solution satisfies:

$$V^\gamma := V(\cdot,t^\gamma) = \mathcal{L}^{-1}\mathcal{Q}V^{\gamma-1} + \mathcal{L}^{-1}\Gamma + \Delta t\mathcal{R}, \tag{140}$$

where the rest $\mathcal{R}$ satisfies $||\mathcal{R}|| = \mathcal{O}(\Delta x^p) + \prime(\Delta t^q)$. Taking the difference between the exact solution and the numerical approximation $W^\gamma := V^\gamma - U^\gamma$, we obtain:

$$W^\gamma = \mathcal{L}^{-1}\mathcal{Q}V^{\gamma-1} + \mathcal{L}^{-1}\Gamma + \Delta t\mathcal{R} - \left(\mathcal{L}^{-1}\mathcal{Q}U^{\gamma-1} + \mathcal{L}^{-1}\Gamma\right) = \mathcal{L}^{-1}\mathcal{Q}(V^{\gamma-1} - U^{\gamma-1}) + \Delta t\mathcal{R}$$

Applying recursively, we obtain:

$$\begin{aligned} W^\gamma &= (\mathcal{L}^{-1}\mathcal{Q})^2(V^{\gamma-2} - U^{\gamma-2}) + \Delta t\mathcal{L}^{-1}\mathcal{Q}\mathcal{R} + \Delta t\mathcal{R} \\ &= (\mathcal{L}^{-1}\mathcal{Q})^\gamma(V^0 - U^0) + \Delta t\sum_{i=0}^{\gamma-1}(\mathcal{L}^{-1}\mathcal{Q})^i\mathcal{R}. \end{aligned}$$

Since $||V^0 - U^0||\mathcal{O}(\Delta x^p)$, and using (133.$b$) we obtain:

$$\begin{aligned} ||U^\gamma - V^\gamma|| = ||W^\gamma|| &\leq \Delta t\sum_{i=0}^{\gamma-1}||(\mathcal{L}^{-1}\mathcal{Q})^i\mathcal{R}|| + \mathcal{O}(\Delta x^p) \\ &\leq ||\mathcal{R}||\Delta t\sum_{i=0}^{\gamma-1} c_1 e^{c_2 j\Delta t/T} + \mathcal{O}(\Delta x^p) \\ &\leq ||\mathcal{R}||\gamma\Delta t c_1 e^{c_2\gamma\Delta t/T} + \mathcal{O}(\Delta x^p) = ||\mathcal{R}||Tc_1e^{c_2} + \mathcal{O}(\Delta x^p) = \mathcal{O}(\Delta x^p) + \mathcal{O}(\Delta t^q), \end{aligned}$$

that proves the convergence of numerical scheme (113). □

D Numerical schemes for Eqs. (1), (2) (or Eq. (6))

To define the numerical method for (1), (2) (or Eq. (6)) we utilize the integral form. Assume that we have a uniform grid on $\mathbb{R}\times\mathbb{R}^+$ with $x_i = i\Delta x$, $t^\gamma = \gamma\Delta t$. Integrating Eq. (1) in the box $[x_{i-1/2}, x_{i+1/2}]\times[t^\gamma, t^{\gamma+1}]$ we obtain:

$$\int_{x_{i-1/2}}^{x_{i+1/2}}(G(U(x,t^{\gamma+1}) - G(U(x,t^\gamma))dx + \int_{t^\gamma}^{t^{\gamma+1}}(F(U(x_{1+\frac{1}{2}},t) - F(U(x_{1-\frac{1}{2}},t))dt = 0, \tag{141}$$

If we consider the numerical method for (1), (2), the numerical discretization for (2) reduces to:

$$H(U(x,t^\gamma)) = 0. \tag{142}$$

For equation (6), the integration on the box $[x_{i-1/2}, x_{i+1/2}]\times[t^\gamma, t^{\gamma+1}]$ give us

$$\int_{x_{i-1/2}}^{x_{i+1/2}}(H(U(x,t^{\gamma+1}) - H(U(x,t^\gamma))dx = 0. \tag{143}$$

We define:

$$G_i^\gamma = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} G(U(x,t^\gamma)dx \quad \text{and} \quad H_i^\gamma = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} H(U(x,t^\gamma)dx. \tag{144}$$

We define V_i^γ the solution $(G(V_i^\gamma) - G_i^\gamma, H(V_i^\gamma) - H_i^\gamma) = 0$. Since we assume that the jacobian of $(G,H)^T$ is nonsingular this solution is unique.

We also define the approximation $f : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}^m$ the flux function as:

$$f_{i+\frac{1}{2}}^\gamma := f(U_i^\gamma, U_{i+1}^\gamma) \sim \frac{1}{\Delta t}\int_{t^\gamma}^{t^{\gamma+1}} F(U(x_{1+\frac{1}{2}}, t)dt \tag{145}$$

Then, we can write (141), (143) as

$$G_i^{\gamma+1} - G_i^\gamma = -\alpha\left(f_{i+\frac{1}{2}}^\gamma - f_{i-\frac{1}{2}}^\gamma\right) \quad \text{and} \quad H_i^{\gamma+1} = H_i^\gamma. \tag{146}$$

The function f is called the numerical flux and is assumed to be consistent with the flux function $F(V)$ if

$$f(U,U) = F(U), \quad \text{for all} \quad U \in \mathbb{R}^n. \tag{147}$$

Definition 11 Let $F \in \mathcal{C}^2(\mathbb{R}^n, \mathbb{R}^m)$, $f \in \mathcal{C}(\mathbb{R}^n \times \mathbb{R}^n, \mathbb{R}^m)$ and suppose that f is consistent, i.e., satisfies (147). Assume that we have a sequence $V_i^0 \in \mathbb{R}^n$, $i \in \mathbb{Z}$ of initial values and Δt, $\Delta x \in \mathbb{R}^+$. Then we define successively for $\gamma \geq 1$, $i \in \mathbb{Z}$ the numerical method given by (146). Then we call f the numerical flux and (146) the numerical scheme in conservation form for the system (1), (6).

Remark 4 Notice that the conservation form guarantees that $\left(\sum_i G_i^{\gamma+1} = \sum_i G_i^\gamma, \sum_i H_i^{\gamma+1} = \sum_i H_i^\gamma\right)$. If $\sum_i H_i^0 = 0$, (i.e., initial states on the surface $\mathcal{S}$) then $\sum_i H_i^\gamma = 0$ for all γ.

Using similar calculations, we can define the numerical method for (1), (2):

Definition 12 Let $F \in \mathcal{C}^2(\mathbb{R}^n, \mathbb{R}^m)$, $f \in \mathcal{C}(\mathbb{R}^n \times \mathbb{R}^n, \mathbb{R}^m)$ and suppose that f is consistent, i.e., satisfies (147). Assume that we have a sequence $V_i^0 \in \mathbb{R}^n$, $i \in \mathbb{Z}$ of initial values and Δt, $\Delta x \in \mathbb{R}^+$. Then we define successively for $\gamma \geq 1$, $i \in \mathbb{Z}$ the numerical method given by

$$G_i^{\gamma+1} - G_i^\gamma = -\alpha\left(f_{i+\frac{1}{2}}^\gamma - f_{i-\frac{1}{2}}^\gamma\right) \quad \text{and} \quad H_i^{\gamma+1} = 0. \tag{148}$$

Then we call f the numerical flux and (146) the numerical scheme in conservation form for the system (1), (6).

We can generalize (146), (148) for more general case (that consider implicit one) substituting (146.a), (148.a) by convex mean:

$$G_i^{\gamma+1} - G_i^\gamma = -\alpha\left(\theta\left(f_{i+\frac{1}{2}}^{\gamma+1} - f_{i-\frac{1}{2}}^{\gamma+1}\right) + (1-\theta)\left(f_{i+\frac{1}{2}}^\gamma - f_{i-\frac{1}{2}}^\gamma\right)\right), \tag{149}$$

for $0 \leq \theta \leq 1$. For $\theta = 0$ the scheme reduces to (146), (148) in the explicit form; for $\theta = 1$ the scheme is completely implicit.

We can consider some examples of numerical schemes. For any numerical scheme, the discretization of Eq. (2) is $H(U_i^{\gamma+1}) = 0$ and for Eq. (6) the numerical discretization give us $H(U_i^{\gamma+1}) = H(U_i^{\gamma+1})$ (single time step). For Eqs. (1) the numerical discretizations are the same obtained only for a PDE, for instance:

Upwind: Forward-time/backward-space

$$\frac{G\left(U_i^{\gamma+1}\right) - G\left(U_i^\gamma\right)}{\Delta t} + \frac{F\left(U_i^\gamma\right) - F\left(U_{i-1}^\gamma\right)}{\Delta x} = 0.$$

here $f(u,v) = F(v)$. *Upwind: Forward-time/forward-space*

$$\frac{G\left(U_i^{\gamma+1}\right) - G\left(U_i^{\gamma}\right)}{\Delta t} + \frac{F\left(U_{i+1}^{\gamma}\right) - F\left(U_i^{\gamma}\right)}{\Delta x} = 0.$$

here $f(u,v) = F(u)$.

Lax-Friedrichs scheme.

$$\frac{G\left(U_i^{\gamma+1}\right) - \frac{(G\left(U_{i+1}^{\gamma}\right)+(G\left(U_{i-1}^{\gamma}\right)}{2}}{\Delta t} + \frac{F\left(U_{i+1}^{\gamma}\right) - F\left(U_i^{\gamma}\right)}{\Delta x} = 0.$$

The most complex numerical implementation is to solve the nonlinear case, because we need to linearize the numerical schemes that we are interested in obtaining the solution. We consider (146) in the explicit case, other methods are very similar and can be obtained using the same process. Since we are interested in obtain $V_i^{\gamma+1}$ it is obtained by solving $(G(V_i^{\gamma}) - G_i^{\gamma}, H(V_i^{\gamma}) - H_i^{\gamma}) = 0$, we substitute $-G_i^{\gamma}$ H_i^{γ} by $G(V_i^{\gamma})$, $H(V_i^{\gamma})$ in the numerical scheme that is writing as:

$$G(V_i^{\gamma}) - G_i^{\gamma} = -\alpha\left(f_{i+\frac{1}{2}}^{\gamma} - f_{i-\frac{1}{2}}^{\gamma}\right) \quad \text{and} \quad H(V_i^{\gamma}) = H_i^{\gamma}. \tag{150}$$

We apply a Newton method to linearize the system. Our initial estimative for $V_i^{\gamma+1}$ is V_i^{γ}. Each approximation of the iterative method is denoted as $(V_i)^l$, then $(V_i)^0 = V_i^{\gamma}$. The linearize scheme (150) is written, in the matricial form, as:

$$J_{G,H}\left((V_i)^l\right)((\Delta V_i)^l) = -\begin{pmatrix} -\alpha\left(f_{i+\frac{1}{2}}^{l} - f_{i-\frac{1}{2}}^{l} - \left(f_{i+\frac{1}{2}}^{\gamma} - f_{i-\frac{1}{2}}^{\gamma}\right)\right) \\ (H_i^l - H_i^{\gamma}) \end{pmatrix}, \tag{151}$$

where $(\Delta V_i)^l = (V_i)^{l+1} - (V_i)^l$, $H_i^l = H\left((V_i)^l\right)$ and $f_{i+\frac{1}{2}}^{l}$, $f_{i-\frac{1}{2}}^{l}$ are the numerical flux evaluated in $(V_i)^l$.

The state $(V_i)^l$ is updated by $(V_i)^l = (V_i)^l + (\Delta V_i)^l$.

For defined tolerances *tol* the method stops if $||(\Delta V_i)^l|| < tol$ or $l \geq n_{max}$, for n_{max} the maximum number of iterations. We also can substitute $||(\Delta V_i)^l|| < tol$ using the image of function that we want obtain the root.

E Expression of the non-uniform block-tridiagonal matrix

In this section we derive expressions for the blocks of the non-uniform block matrix $J_{\mathbf{G}}$ in equation (70), which has the general form

$$J_{\mathbf{G}} = \begin{pmatrix} A_0 & B_0 & \tilde{B} & & & \\ C_0 & A_1 & B_1 & & & \\ & C_1 & A_2 & B_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & C_{M-2} & A_{M-1} & B_{M-1} \\ & & & \tilde{C} & C_{M-1} & A_M \end{pmatrix}. \tag{152}$$

The expression above differs from matrix $\mathbf{A}$ of equation (71) by the presence of the two extra blocks $\tilde{B}$ and $\tilde{C}$. This fact implies on slight modifications to the decompositions and algorithms of Section 5.2, but it does not change considerably the cost or the speedup ratios discussed in Section 5.3.

The expression of blocks A_l, B_l and C_{l-1}, for $0 < l < M$, is obtained by deriving the residuals of equations (65), (66), with respect to U_k^{l+1}, for $0 < k < M$. This calculation implies on matrices of the form

$$A_l = \begin{bmatrix} \kappa[G']_l^{k+1} - \mu_1 \left[[\mathcal{B}']_l^{k+1}(U_{l+1}^{k+1} - 2U_l^{k+1} + U_{l-1}^{k+1}) - \right. \\ \left. -(\mathcal{B}_{l+1}^{k+1} + 2\mathcal{B}_l^{k+1} + \mathcal{B}_{l-1}^{k+1})I \right] - \alpha_1[R']_l^{k+1} \\ \\ (\tau\kappa + \alpha_2)\left[H'\right]_l^{k+1} \end{bmatrix}, \tag{153}$$

$$B_l = \begin{bmatrix} -\mu_1[\mathcal{B}']_{l+1}^{k+1}(U_{l+1}^{k+1} - U_l^{k+1}) - \mu_1(\mathcal{B}_{l+1}^{k+1} + \mathcal{B}_l^{k+1})I + \eta_1[F']_{l+1}^{k+1} \\ \\ \mathbf{0} \end{bmatrix}, \tag{154}$$

$$C_{l-1} = \begin{bmatrix} \mu_1[\mathcal{B}']_{l-1}^{k+1}(U_l^{k+1} - U_{l-1}^{k+1}) - \mu_1(\mathcal{B}_l^{k+1} + \mathcal{B}_{l-1}^{k+1})I - \eta_1[F']_{l-1}^{k+1} \\ \\ \mathbf{0} \end{bmatrix}, \tag{155}$$

with I denoting an identity matrix of size $n \times n$ and G', F', $\mathcal{B}'$, R' and H' denoting the jacobian matrices of G, F, $\mathcal{B}$, R and H respectively. For $l = 0$ and $l = M$, we have to find the derivatives of the residuals of the boundary conditions in (68), (69). In this case, we obtain

$$A_0 = \left[\alpha_1\mathscr{A}_L - 3\eta_1\mathscr{B}_L\right], \quad B_0 = \left[4\eta_1\mathscr{B}_L\right], \quad \tilde{B} = \left[-\eta_1\mathscr{B}_L,\right] \tag{156}$$

and

$$\tilde{C} = \left[3\eta_1\mathscr{B}_R\right], \quad C_{M-1} = \left[-4\eta_1\mathscr{B}_R\right], A_M = \left[\alpha_1\mathscr{A}_R + \eta_1\mathscr{B}_R\right]. \tag{157}$$

F A floating point cost comparison of the block decompositions

F.1 A floating point cost comparison of the block decompositions: 1 to 8

To be more precise about the difference among the decompositions presented in Section 5.2, we now perform an approximate floating point count of the Algorithms 1 and 3. Let us start defining $c_{ls}(n, m)$ to be the cost of a solver algorithm for a full-rank linear system of size $n \times n$ with m right hand sides, and $c_{am}(n, p, m)$ to be the cost of the add and multiply operation $A := B + CD$ for A, B of size $n \times m$, C of size $n \times p$, D of size $p \times m$. Using these definitions, we can see that the flop count for the block **LU** decomposition in Algorithm 1 is

$$c_{bdcp}(M, n, m) := \sum_{i=1}^{M-1} c_{ls}(n, m) + c_{am}(m, m, m) = (M-1)(c_{ls}(n, m) + c_{am}(m, m, m)), \tag{158}$$

while the flop count for Algorithm 3 is $c_{bdcp}(M, n, n)$. As a result, the advantage of the Algorithm 1 over 3 can be expressed as

$$\varsigma(M, n, p) = \frac{c_{bdcp}(M, n, n)}{c_{bdcp}(M, n, n-p)}. \tag{159}$$

Here ς represents how many times is one algorithm faster than the other, at least in terms of floating point operations. We will refer to this quantity as *speedup* (in analogy to the speedup quantity in parallel programming).

The asymptotic speedup $\varsigma(M, n, p)$ when $p \approx n \gg 1$ can be measured theoretically as follows: to fix ideas, let us take the linear solver algorithm to be the one for the classical LU decomposition method, so that

$$c_{ls}(n, m) = c_{dcp}(n) + c_{fbs}(n, m), \tag{160}$$

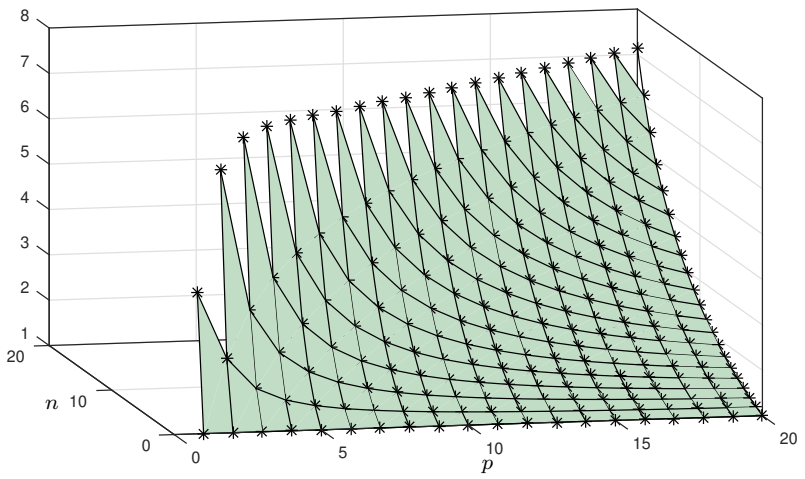

Fig. 10 Graph of the speedup $\varsigma(M,n,p)$ defined in (159), using equations (160) and (161). Here M was considered very large, so that $M \approx M-1$ is true.

where the cost of the decomposition is $c_{dcp}(n) \approx \frac{4n^3-3n^2+5n}{6}$ and the cost of the forward and backward substitution for m right hand sides is $c_{fbs}(n,m) \approx nm(2n-1)$. Taking the cost of the add and multiply algorithm to be

$$c_{am}(n,p,m) \approx 2mnp, \tag{161}$$

one can easily prove that

$$\lim_{(M,n,p)\to(\infty,\infty,n)} \varsigma(M,n,p) = \lim_{(n,p)\to(\infty,n)} \frac{28n^3-9n^2+5n}{28n^3-48n^2p+36np^2-12p^3-9n^2+6np+5n} = 7, \tag{162}$$

which means that for M, n large, and p close to n, Algorithm 1 is up to 7 times faster than Algorithm 3. In Figure 10 we can see the graph of $\varsigma(M,n,p)$ for n up to 20, $M \gg 1$. In the worst case scenario, which corresponds to $n=20$, $p=1$, $m=19$, all algorithms perform similarly: $\varsigma(n,p) \approx 1.09$. In the best case scenario, corresponding to $n=6$, $p=5$, $m=1$, we get $\varsigma(n,p) \approx 7.33$ and it is clear that choosing Algorithm 3 is the best choice amongst the discussed ones. For case $n=20$, $p=10$, $m=10$, we still have $\varsigma(n,p)$ greater than 1: $\varsigma(n,p) \approx 2.46$, and therefore it is evident that using the appropriate decomposition matters. Even when $m=n$, the cost of both algorithms is the same, so there is no disadvantage in using the top-to-bottom block **LU** decomposition rather than the left-to-right block **LU** version.

Usually the decomposition process represents most of the work to be done for the linear solver. That is the case for only one right hand side vector: the decomposition process is of order 3, while the forward and backward substitution process is only order 2. However, the block forward and backward substitution part can become order 3 as well if there are sufficient right hand side vectors. That is the case for the *Divide and Conquer method*, discussed in the next section. By taking the right hand side b to be of size $nM \times q$, we can measure the asymptotic speedup obtained by solving (71) using Algorithm 1 along with 2 instead of using 3 along with 4 by the following expression:

$$\varsigma^*(M,n,p,q) = \frac{c_{bls}(M,n,n,q)}{c_{bls}(M,n,n-p,q)}, \tag{163}$$

where

$$c_{bls}(M,n,m,q) = c_{bdcp}(M,n,m) + c_{bfbs}(M,n,m,q), \tag{164}$$

for $c_{bfbs}(M,n,m,q) \approx 2(M-1)c_{am}(m,n,q) + Mc_{fbs}(n,q)$ being the cost of the block forward and backward substitutions with q right hand side vectors. As expected, if there is only one right hand side vector in the block-tridiagonal linear system, we have

$$\lim_{(M,n,p,q)\to(\infty,\infty,n,1)} \varsigma^*(M,n,p,q) = \lim_{(n,p,q)\to(\infty,n,1)} \frac{4n^3+12n^2p+12p^3+9n^2+18np-n}{28n^3-48n^2p+36np^2-12p^3+21n^2-18np-n} = 7, \tag{165}$$

and the maximum theoretical speedup is still 7. However, considering the case $q = n$, we have

$$\lim_{(M,n,p,q)\to(\infty,\infty,n,n)} \varsigma^*(M,n,p,q) \lim_{(n,p,q)\to(\infty,n,n)} \frac{4n^3+12n^2p+24npq+12n^2q+12p^3+\cdots}{28n^3-48n^2p+36np^2-24npq+36n^2q-12p^3+\cdots} = 4, \tag{166}$$

and asymptotic speedup is now reduced to 4. The reason for this reduction in the speedup for a big number of right hand side vectors is explained by the fact that the floating point operations necessary to compute the forward and backward substitutions is independent of p in equation (163). As a result, no matter the value of p, whenever $q = n$ we have that $c_{fbs}(n,q)$ is of order $O(n^3)$, which causes this term to influence both the numerator and denominator equally. However, independently on the number of right hand sides in the non-uniform block-tridiagonal linear system, using the top-to-bottom block **LU** decomposition is still better than left-to-right version.

*F.1.1 A floating point cost comparison to the block **LU** decomposition: algorithm 9*

We will now perform a floating point operation count of the most expensive part of the DCB method: the step 1 of Algorithm 9. We will further compare it to the floating point cost of the top-to-bottom block **LU** decomposition presented in Subsection 5.2, the fastest sequential method we have discussed.

Let us consider that the machine in which the non-uniform block-tridiagonal linear system is to be solve has ρ processing units. We assume the perfect partition $M = \rho M_1 + (\rho - 1)$, so that $M_i = M_1$, for $i = 2, \cdots, \rho$ in (78). Considering only one right hand side vector, the most expensive step in Algorithm 9 is the first one, with floating point count that can be approximated by:

$$c_{stp1}(M_1,n,m) = c_{bdcp}(M_1,n,m) + c_{bfbs}(M_1,n,m,n+1) + c^*_{bfbs}(M_1,n,m,n), \tag{167}$$

where $c^*_{bfbs}(M_1,n,m,q) \approx (M_1-1)c_{am}(m,n,q) + M_1 c_{fbs}(n,q)$ is the floating point operation count of the block forward and backward substitution that accounts for the special structure of θ_i, $i = 1, \cdots \rho - 1$. Considering that the decomposition process is the most expensive in the top-to-bottom block **LU** decomposition method (Algorithm 1), we can approximate the speedup ς^{DCB} of the DCB method as

$$\varsigma^{DCB}(\rho, M_1, n, p) = \frac{c_{bdcp}(\rho M_1 + (\rho-1), n, n-p)}{c_{stp1}(M_1, n, n-p)}, \tag{168}$$

where $p = n - m$ is the number of zero rows in the non-uniform blocks of the matrix. We first consider the case $p = 0$, which gives

$$\lim_{(M_1,n,p)\to(\infty,\infty,0)} \varsigma^{DCB}(\rho, M_1, n, p) = \lim_{n\to\infty} \frac{\rho(28n^2-9n+5)}{88n^2+15n-1} = \frac{7\rho}{22}. \tag{169}$$

The asymptotic speedup $\frac{7\rho}{22}$ is a slight improvement of the one obtained in [37]: $\frac{7\rho}{25}$. The reason for that is simply because of the floating point operation savings due to the special

form of θ_i, $i = 1, \cdots \rho - 1$. We now consider the case of non-uniform blocks filled with several rows of zeros:

$$\lim_{(M_1,n,p)\to(\infty,\infty,n)} \varsigma^{DCB}(\rho, M_1, n, p) = \lim_{n\to\infty} \frac{\rho(4n^2 - 3n + 5)}{28n^2 - 3n - 1} = \frac{\rho}{7}, \tag{170}$$

and therefore the asymptotic speedup is reduced when more null rows are inserted into the non-linear block systems. Although the ratio $\frac{\rho}{7}$ may look discouraging at first, we have only considered floating point operations in our calculations. Several other aspects usually influence in the parallelization of the method, such as memory operations.

References

1. A. C. Alvarez, T. Blom, W. J. Lambert, J. Bruining, and D. Marchesin. Analytical and numerical validation of a model for flooding by saline carbonated water. *Journal of Petroleum Science and Engineering*, 167:900–917, 2018.
2. A. C. Alvarez, J. Bruining, W. J. Lambert, and D. Marchesin. The riemann solution for carbonated waterflooding. *ECMOR XV - 15th European Conference on the Mathematics of Oil Recovery*, 2016.
3. A. C. Alvarez, J. Bruining, W. J. Lambert, and D. Marchesin. Analytical and numerical solutions for carbonated waterflooding. *Computational Geosciences*, 22(2):505–526, 2018.
4. A. C. Alvarez, G. T. Goedert, and D. Marchesin. Resonance in rarefaction and shock curves: local analysis and numerics of the continuation method. *arXiv preprint arXiv:1902.04182*, 2019.
5. P. Arbenz, A. Cleary, J. Dongarra, and M. Hegland. A comparison of parallel solvers for diagonally dominant and general narrow-banded linear systems ii. In Patrick Amestoy, Philippe Berger, Michel Daydé, Daniel Ruiz, Iain Duff, Valérie Frayssé, and Luc Giraud, editors, *Euro-Par'99 Parallel Processing*, pages 1078–1087, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.
6. A. V. Azevedo and D. Marchesin. Multiple viscous solutions for systems of conservation laws. *Transactions of the American Mathematical Society*, 347(8):3061–3077, 1995.
7. A. Bartel and M. Gunther. Pdaes in refined electrical network modeling. *SIAM Review*, 60(1):56–91, 2018.
8. B. Benhammouda and H. Vazquez-Leal. Analytical solutions for systems of partial differential algebraic equations. *Springer Plus*, 3(137):9, 2014.
9. S. Bondeli. Divide and conquer: a parallel algorithm for the solution of a tridiagonal linear system of equations. *Parallel Computing*, 17(4):419 – 434, 1991.
10. S. Bondeli and W. Gander. Cyclic reduction for special tridiagonal systems. *SIAM Journal on Matrix Analysis and Applications*, 15, 01 1994.
11. G. Chavez, G. Turkiyyah, S. Zampini, H. Ltaief, and D. Keyes. Accelerated cyclic reduction: A distributed-memory fast solver for structured linear systems. *Parallel Computing*, 2017.
12. K. Chudej, H. J. Pesch, and J. Rang. *Chapter: Index Analysis of Models of the Book "Molten Carbonate Fuel Cells"*. Wiley, Weinheim, 2007.
13. C. Dafermos. *Hyperbolic Conservation Laws in Continuum Physics*. Springer Verlag, USA, 2010.
14. J. W. Demmel, J. H. Higham, and R. S. Schreiber. Stability of block lu factorization. *Numerical Linear Algebra with Applications*, 2, 1995.
15. P. Engesgaard and K. L. Kipp. A geochemical transport model for redox?controlled movement of mineral fronts in groundwater flow systems: A case of nitrate removal by oxidation of pyrite. *Water Resources Research*, 28:2829 – 2843, 1992.
16. E. Godlewski and P. A. Raviart. *Numerical Approximation of Hyperbolic Systems of Conservation Laws*. Springer, 1996.
17. D. Gottlieb and J.S. Hesthaven. *Spectral methods for hyperbolic problems*, volume 7 of *Numerical Analysis 2000*. Elsevier, Amsterdam, 2001.

18. M. Gunther and Y. Wagner. Index concepts for linear mixed systems of differential-algebraic and hyperbolic-type equations. *SIAM J. Sci. Comput.*, 22(5):1610 – 1629, 2000.
19. J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for Time-Dependent Problems*. Cambridge University Press, 2007.
20. N.J. Higham. *Accuracy and Stability of Numerical Algorithms: Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2002.
21. G. Hime. Parallel solution of nonlinear balance systems. Master's thesis, Laboratorio Nacional de Computacao Cientifica, Petropolis - RJ, Brazil, 2007.
22. S. P. Hirshman, S. Perumalla, V.E. Lynch, and R. Sanchez. Bcyclic: A parallel block tridiagonal matrix cyclic solver. *Journal of Computational Physics*, 229, 2010.
23. S. Jin and P. Xin. The relaxation schemes for systems of conservation laws in arbitrary space dimensions. *Comm. Pure Appl. Math.*, 48(3):253 – 281, 1995.
24. D. Kroner. *Numerical Schemes for Conservation Laws*. Wiley Teubner, USA, 1997.
25. D. A. Kulik, T. Wagner, S. V. Dmytrieva, G. Kosakowski, F. F. Hingerl, K. V. Chudnenko, and U. R. Berner. Gem-selektor geochemical modeling package: revised algorithm and gems3k numerical kernel for coupled simulation codes. *Computational Geosciences*, 17(1):1–24, 2013.
26. W. Lambert and D. Marchesin. The riemann problem for multiphase flows in porous media with mass transfer between phases. *Journal of Hyperbolic Equations*, 8(02):149–156, 2009.
27. W.J. Lambert, A.C. Alvarez, D. Marchesin, and J. Bruining. Mathematical theory of two-phase geochemical flow with chemical species. *In: Klingenberg C., Westdickenberg M. (eds) Theory, Numerics and Applications of Hyperbolic Problems II. HYP 2016. Springer Proceedings in Mathematics and Statistics*, 237:255 – 267, 2018.
28. W.J. Lambert, A.C. Alvarez, V. Matos, D. Marchesin, and J. Bruining. Nonlinear wave analysis of geochemical injection for multicomponent two phase flow in porous media. *Journal of Differential Equations*, 266:406 – 454, 2019.
29. P. D. Lax and R. D. Richtmyer. Survey of the stability of linear finite difference equations. *Communications on Pure and Applied Mathematics*, 9(2):267–293, 1956.
30. R. J. Leveque. *Numerical Methods for Conservation Laws*. Lecture Notes in Mathematics Springer Basel AG, 1990.
31. R. J. Leveque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge TExts in Applied Mathematics, Cambridge, 2002.
32. T. P. Liu. The riemann problem for general $2\times$ 2 conservation laws. *Transactions of the American Mathematical Society*, 199:89–112, 1974.
33. Tai-Ping Liu. The riemann problem for general systems of conservation laws. *Journal of Differential Equations*, 18(1):218–234, 1975.
34. W. Marszalek. Analysis of partial differential algebraic equations. Master's thesis, North Carolina State University, Raleigh, USA, 1997.
35. W. S. Martinson and P. I. Barton. A differentiation index for partial differential-algebraic equations. *SIAM J. Sci. Comput.*, 21(6):2295 – 2315, 2000.
36. R. M. M. Mattheij. The stability of lu-decompositions of block tridiagonal matrices. *Bulletin of the Australian Mathematical Society*, 29, 4 1984.
37. V. Mehrmann. Divide and conquer methods for block tridiagonal systems. *Parallel Computing*, 19, 1993.
38. M. Muniruzzamana and M. Rollea. Modeling multicomponent ionic transport in groundwater with iphreeqc coupling: Electrostatic interactions and geochemical reactions in homogeneous and heterogeneous domains. *Advances in Water Resources*, 98:1 – 15, 2016.
39. M. Reintjes. Constrained systems of conservation laws: a geometric theory. *Methods and Applications of Analysis*, 24(4):407 – 404, 2017.
40. D. Serre. *Systems of Conservation Laws 1: Hyperbolicity, entropies, shock waves*. Cambridge University Press, 1999.
41. D. Serre. *Systems of Conservation Laws, 2 volums*. Cambridge University Press, England, 1999.
42. J. Smoller. *Shock Waves and Reaction-Diffusion Equations*. Springer-Verlag, USA, 1983.

43. G. A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27, 1978.
44. E. Sonnendrucker. *Finite Element methods for hyperbolic systems.* Lecture notes Wintersemester 2014-2015, Max Planck Institut, 2015.
45. J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations.* Wadsworth Publ. Co., Belmont, CA, USA, 1989.
46. J. M. Varah. On the solution of block-tridiagonal systems arising from certain finite-difference equations. *Mathematics of Computation*, 26(120):859–868, 1972.
47. H. Wahanik. *Thermal effects in the injection of CO2 in deep underground aquifers.* PhD thesis, IMPA, Rio de Janeiro, Brazil, 2011.
48. G. T. Yeh and V. S. Tripathi. A critical evaluation of recent developments in hydrogeochemical transport models of reactive multichemical components. *Water Resources Research*, 25(1):93 – 108, 1989.
49. G. T. Yeh and V. S. Tripathi. A model for simulating transport of reactive multispecies components: Model development and demonstration. *Water Resources Research*, 27(12):3075 – 3094, 1991.
50. G.T. Yeh, Y. Fang, F. Zhang, J. Sun, Y. Li, M. H. Li, and M. D. Siegel. Numerical modeling of coupled fluid flow and thermal and reactive biogeochemical transport in porous and fractured medias. *Computational Geosciences*, 14:149–170, 2010.