Implicit time schemes for compressible fluid models based on relaxation methods

David Coulette, Emmanuel Franck, Philippe Helluy, Ahmed Ratnani, Eric Sonnendrücker

Implicit time schemes for compressible fluid models based on relaxation methods

D. Coulette[†], E. Franck[♯], P. Helluy[♯], A. Ratnani[‡], E. Sonnendrücker[§]

April 26, 2017

Abstract

In this work, we are interested in the implicit discretization of compressible fluid models. When applied to this type of models, standard implicit schemes suffer from a lack of efficiency due to the ill-conditioning of the associated matrices. In this paper we propose an alternative strategy based on a relaxation method coupled with an implicit scheme. This method allows us to approximate the full complex problem by a collection of simpler ones than we can solve with standard methods. We obtain eventually a second order implicit method with some additional parallelism able to treat hyperbolic systems with small diffusion.

Contents

[1]Inria Nancy Grand-Est, Nancy, France.
[2]University of Strasbourg, Irma, Strasbourg, France.
[3]TUM, Garching bei München, Germany.
[4]Max-Planck-Institut für Plasmaphysik, Boltzmannstraße 2D-85748 Garching, Germany.

1 Introduction

In this work we consider the time discretization of compressible fluid models which appear in gas dynamics, biology, astrophysics or plasma physics for Tokamaks. Such models can be unified under the generic form

$$\partial_t \boldsymbol{U} + \nabla \cdot \boldsymbol{F}(\boldsymbol{U}) = \nu \nabla \cdot (D(\boldsymbol{U}) \nabla \boldsymbol{U})$$

with $\boldsymbol{U} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, $\boldsymbol{F}(\boldsymbol{U}) : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ and $D(\boldsymbol{U}) : \mathbb{R}^{2n} \longrightarrow \mathbb{R}^n$. We assume for $\nu = 0$ that the system is a hyperbolic system. Consequently the matrix $\sum_{i=1}^{d} \partial F_i(\boldsymbol{U}) n_i$ with $\mathbf{n} \in \mathbb{R}^d$ a normal vector is a diagonalizable. A first possibility for the time discretization of (1) is to use an explicit scheme. In that case the time step is constrained by $\Delta t < \min(\frac{\Delta x}{\lambda_{max}}, \frac{\Delta x^2}{\nu})$. This parabolic CFL condition arising from the diffusion term is often overly restrictive and generally leads to switching to an implicit scheme for the parabolic part of the system, while keeping the resolution of the hyperbolic part explicit. However if the characteristic velocity of the fluid V is very small compared to the fastest velocity scale λ_{max} it can be advantageous to use an implicit scheme for the transport part as well. Indeed the implicit scheme allows to filter out unwanted fast scales and choose a time step adapted to the characteristic velocity of the fluid independently of the mesh resolution.
A first example of such a situation is the Euler equation in the low Mach number regime. For this regime it is interesting to have a time step adapted to the fluid velocity and avoid a CFL constrained by the acoustic wave which is comparatively very fast in this case. A second typical example is magnetohydrodynamics (MHD) for which the fast magneto-sonic waves generate very restrictive CFL conditions.
For some applications such as MHD simulations for magnetic fusion devices, system (1) has to be solved on a finely resolved three dimensional mesh. With those constraints, the use of direct solvers lead to prohibitive costs, both in terms of computing power and memory load. As a consequence, iterative solvers are more suited to the task. The efficiency of the iterative solvers is given by the conditioning number of the matrix which is the ratio between the maximal and minimal eigenvalues (or singular values). Let us consider for instance the following system, obtained by application of the simplest implicit time discretization:

$$\boldsymbol{U} + \Delta t \nabla \cdot \boldsymbol{F}(\boldsymbol{U}) - \Delta t \nu \nabla \cdot (D(\boldsymbol{U}) \nabla \boldsymbol{U}) = \boldsymbol{U}^n. \tag{1}$$

When the diffusion term is small ($\nu \ll 1$) and the time step is large ($\Delta t \gg 1$) , inverting the previous system is akin to inverting $\nabla \cdot \boldsymbol{F}(\boldsymbol{U}) = 0$. The difficulty is that this steady-state system is very hard to invert: consequently for large time steps the invertibility of the full system is also

a complicate problem. The first reason is that the conditioning of the matrix associated with the discretization of $\nabla \cdot \boldsymbol{F}(\boldsymbol{U})$ is linked to the ratio between the characteristic wave velocities (eigenvalues of the Jacobian). Indeed if this ratio is very large (either because of fast velocities in the MHD case or null velocity for low Mach regimes) the conditioning of the discrete system roughly follows the same ratio. Consequently for some cases the scale gap between the various waves of the systems generates ill-conditioned matrices. The second problem comes from the structure of the steady-state hyperbolic system which is not invertible. In order to illustrate that point, we linearize around a constant state $\boldsymbol{U}^0$ to obtain

$$\nabla \cdot (A(\boldsymbol{U}^0)\boldsymbol{U}) = 0.$$

We then perform the Fourier transform of the previously linearized system with $\hat{U}$ the Fourier transform of $\boldsymbol{U}$ and $\mathbf{k}$ the Fourier variable. We obtain

$$A(\boldsymbol{U}_0, \mathbf{k})\hat{\boldsymbol{U}} = 0.$$

Diagonalizing the system, we obtain a new one $\Lambda(\boldsymbol{U}_0, \mathbf{k})\hat{\mathbf{V}} = 0$ with $\mathbf{V} = P^{-1}(\boldsymbol{U}_0, \mathbf{k})\hat{\boldsymbol{U}}$. The difficulty stems from the fact that there always exists values of $\mathbf{k}$ which make the system singular, ie lead to the apparition of null eigenvalues. On a finely resolved spatial mesh, a large number of discrete Fourier modes can lead to this situation. To summarize the problem, the steady system is not invertible since we can have eigenvalues which tend to zero for some physical regimes (such a the low Mach one for instance) and since the problem is not isotropic there is not enough information to build a solution.
Getting back to problem (1), there exists in that case a unique solution but when $\Delta t >> 1$, the model tends to the singular steady hyperbolic system and consequently the conditioning number tends to the conditioning number of the steady hyperbolic problem (which is infinite).
In order to avoid those issues we need an algorithm which allows to split the initial system, which is non-linear and complex to solve, into a sequence of smaller and better conditioned ones. A first possibility is to design a preconditionner for the implicit problem.We refer for instance to the works of ([4]-[3]-[11]-[5]) which use approximations and reformulations of the physical model to obtain a preconditioning or the works of [12] which proposed a block approximation (based for instance on directional splitting). One can also use an implicit or semi-implicit reformulation which allows to exhibit simpler operators (second order for instance) with can be solved with standard techniques [8]. Yet another method is the semi-implicit scheme which discretizes implicitly the stiff-part of the equations (only the fast wave for instance) [2]-[8].

In this work, we propose an alternative method to treat hyperbolic systems using an implicit scheme. The key idea is to use the "relaxation method" proposed initially by S. Jin and L. Xin [9]-[10]-[7]-[1]. This method allows to approximate the full nonlinear system by a larger linear system with a nonlinear local source term. The larger system can be decomposed into a collection of independent simple systems. Coupling this model with a time splitting scheme and with a specialized solver dedicated to linear hyperbolic systems, we end up with an implicit scheme which requires only the resolution of a set of weakly coupled simple systems.

In the first section we will introduce the relaxation method for general systems in one dimension and propose a generalization which allows to treat source and small diffusion terms. We will then introduce the splitting scheme for this model and study the consistency of the method before proposing high order extensions. In the second section we will consider the multidimensional extension of the relaxation method. In the last section we will discuss the implementation and spatial solver before illustrating the method on the various examples of models such as the Burgers equation or the compressible Navier-Stokes equations. In this work we strictly restrict ourselves to the numerical validation of the method, with no emphasis on performance optimization or parallelization.

2 Relaxation method for 1D hyperbolic system with small diffusion

In this section we present the relaxation method for the design of a implicit scheme for a one dimensional general system. This method allows to approximate a nonlinear system by another one where the nonlocal part is linear and the non-linearity is local. We will first recall a formal result on the approximation by the relaxation method and then we will present a slight generalization which allows to take small diffusion and source terms into account.

2.1 Relaxation systems

In this section we consider the following system

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \nu \partial_x (D(\boldsymbol{U}) \partial_x \boldsymbol{U}) + S(\boldsymbol{U}) \tag{2}$$

To begin we consider only the hyperbolic part (the source and diffusion part will be treat after). Xin and Jin's relaxation method consists in approximating the hyperbolic part of the previous system by

$$\begin{cases} \partial_t \boldsymbol{U} + \partial_x \boldsymbol{V} = 0 \\ \partial_t \boldsymbol{V} + \alpha^2 \partial_x \boldsymbol{U} = \dfrac{1}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) \end{cases} \tag{3}$$

This system is an approximation of the model (2) when ϵ tends to zero and the advantage is that the non-linearity is local and the non locality is linear. Before introducing a scheme based on this approximate model, we prove formally that this system is an approximation to (2).

Lemma 2.1. *The model (3) is equivalent the following system*

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \epsilon \partial_x ((\alpha^2 I_d - A(\boldsymbol{U})^2) \partial_x \boldsymbol{U})$$

with $A(\boldsymbol{U})$ the Jacobian of $\boldsymbol{F}(\boldsymbol{U})$. The limit system is stable if $(\alpha^2 I_d - A(\boldsymbol{U})^2)$ it is a definite positive matrix.

Proof. The second set of equations is equivalent to

$$\boldsymbol{V} = \boldsymbol{F}(\boldsymbol{U}) - \epsilon(\partial_t \boldsymbol{V} + \alpha^2 \partial_x \boldsymbol{U})$$

Therefore $\partial_t \boldsymbol{V} = \partial_t \boldsymbol{F}(\boldsymbol{U}) + O(\epsilon)$. Now we multiply the first set of equations by $A(\boldsymbol{U})$ to obtain

$$\begin{aligned}
&A(\boldsymbol{U})(\partial_t \boldsymbol{U} + \partial_x \boldsymbol{V}) = 0 \\
&A(\boldsymbol{U})(\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U})) = O(\epsilon) \\
&\partial_t \boldsymbol{F}(\boldsymbol{U}) + A(\boldsymbol{U})\partial_x(\boldsymbol{F}(\boldsymbol{U})) = O(\epsilon) \\
&\partial_t \boldsymbol{F}(\boldsymbol{U}) + A(\boldsymbol{U})^2 \partial_x \boldsymbol{U} = O(\epsilon)
\end{aligned}$$

therefore

$$\boldsymbol{u} = \boldsymbol{F}(\boldsymbol{U}) - \epsilon \left(\left(\alpha^2 I_d - A(\boldsymbol{U})^2 \right) \partial_x \boldsymbol{U} \right)$$

and we obtain

$$\partial_t \boldsymbol{U} + \partial_x(\boldsymbol{F}(\boldsymbol{U})) = \epsilon \partial_x \left(\left(\alpha^2 I_d - A(\boldsymbol{U})^2 \right) \partial_x \boldsymbol{U} \right)$$

For the stability analysis we use the entropy of the hyperbolic system $\eta(\boldsymbol{U})$ and the entropy flux $Q(\boldsymbol{U})$. Multiplying the limit equation by the derivative of the entropy and using the property of the entropy flux we obtain

$$\partial_t \eta(\boldsymbol{U}) + \partial_x(Q(\boldsymbol{U})) = \epsilon \eta'(\boldsymbol{U}) \partial_x \left(\left(\alpha^2 I_d - A(\boldsymbol{U})^2 \right) \partial_x \boldsymbol{U} \right)$$

Integrating over the domain we have

$$\partial_t \int \eta(\boldsymbol{U}) + \int \partial_x(Q(\boldsymbol{U})) - \epsilon \int \eta'(\boldsymbol{U}) \partial_x \left(\left(\alpha^2 I_d - A(\boldsymbol{U})^2 \right) \partial_x \boldsymbol{U} \right) = 0$$

Using a Green theorem and integration by part we obtain

$$\partial_t \int \eta(\boldsymbol{U}) + \epsilon \int \left(\eta''(\boldsymbol{U}) \left(\alpha^2 I_d - A(\boldsymbol{U})^2 \right) \partial_x \boldsymbol{U}, \partial_x \boldsymbol{U} \right) = 0$$

Since the entropy η is convex, by definition the matrix $\eta''(\boldsymbol{U}) \left(\alpha^2 I_d - A(\boldsymbol{U})^2 \right)$ is positive if $\left(\alpha^2 I_d - A(\boldsymbol{U})^2 \right) > 0$ and in this case the total entropy is dissipated in time which ensures the stability of the scheme. □

This result makes apparent the respective roles and importance of the two parameters ϵ and α. The relaxation parameter ϵ allows to adapt the approximation. The smallest this parameter is, the better the approximation. The parameter α is essential for the stability of the scheme. Indeed if α it is larger than the maximal speed (ie if one overestimates the velocity of the transport) of the system the scheme generates diffusion which stabilizes the system. If it is not the case, the schemes generates anti-diffusion and the limit system is unstable.

2.2 Treatment of diffusion and source term terms

We will now add diffusion effects to the model while keeping the overall structure of the relaxation algorithm unchanged. Indeed in this case we have a linear hyperbolic system with a local nonlinear source term. In order to preserve the ability to split the linear and local non-linear parts

we propose to use the diffusion generated by the relaxation method itself to capture the physical diffusion. This idea is commonly used in schemes based on the Lattice Boltzmann method where the relaxation parameters are tweaked so that the diffusion stemming from the relaxation scheme mimics the physical one. To that end we consider now that the relaxation parameter is not a scalar anymore but a matrix depending on the primitive variables. We obtain

$$\begin{cases} \partial_t \boldsymbol{U} + \partial_x \boldsymbol{V} = 0 \\ \partial_t \boldsymbol{V} + \alpha^2 \partial_x \boldsymbol{U} = \dfrac{R(\boldsymbol{U})}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) \end{cases} \tag{4}$$

Using the same asymptotic analysis we obtain this limit

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \epsilon \partial_x (R^{-1}(\boldsymbol{U})(\alpha^2 - A(\boldsymbol{U})^2)\partial_x \boldsymbol{U}) + O(\epsilon^2)$$

Since R is a matrix, we have an additional degree of freedom to modify the structure of the diffusion induced by the relaxation. It is obvious that $\epsilon = \nu$ and $R = (\alpha^2 - A(\boldsymbol{U})^2)D(\boldsymbol{U})^{-1}$. This method works only if the diffusion matrix is invertible. In practice it is not always the case (there is often no diffusion on the first equation in fluid mechanics for instance. If it is the case, the matrix $D(\boldsymbol{U})$ can be replaced by another invertible matrix with difference between the matrices homogeneous to $O(\nu^2)$. All in all the relaxation method is an approximation to the initial problem with an error homogenous $O(\nu^2)$. It is consequently only valid for low diffusion ($\nu \ll 1$) regimes.
Let us now add the effect of the source term. A first possibility is just to directly add the source term in the relaxation system.

$$\begin{cases} \partial_t \boldsymbol{U} + \partial_x \boldsymbol{V} = \boldsymbol{G}(\boldsymbol{U}) \\ \partial_t \boldsymbol{V} + \alpha^2 \partial_x \boldsymbol{U} = \dfrac{R}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) \end{cases} \tag{5}$$

If we use this solution and apply the previous asymptotic proof, we obtain at the limit

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \boldsymbol{G}(\boldsymbol{U}) + \epsilon \partial_x (R(\boldsymbol{U})^{-1}(\alpha^2 - A(\boldsymbol{U})^2)\partial_x \boldsymbol{U}) + \epsilon \partial_x R(\boldsymbol{U})^{-1}(A(\boldsymbol{U})\boldsymbol{G}(\boldsymbol{U}))$$

As the coefficient ϵ is now used to set the diffusion scale, its value is fixed by the physical diffusion. Consequently the error between the relaxation model and the full model with sources and diffusion terms is of the same order of magnitude as the physical diffusion. To fix this issue we propose another relaxation system

$$\begin{cases} \partial_t \boldsymbol{U} + \partial_x \boldsymbol{V} = \boldsymbol{G}(\boldsymbol{U}) \\ \partial_t \boldsymbol{V} + \alpha^2 \partial_x \boldsymbol{U} = \dfrac{R(\boldsymbol{U})}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) + \boldsymbol{H}(\boldsymbol{U}) \end{cases} \tag{6}$$

Lemma 2.2. *We assume that $D(\boldsymbol{U})$ is an invertible matrix. The model (6) with $\boldsymbol{H}(U) = A(\boldsymbol{U})\boldsymbol{G}(\boldsymbol{U})$, $\epsilon = \nu$ and $R = (\alpha^2 - A(\boldsymbol{U})^2)D(\boldsymbol{U})^{-1}$ is equivalent the following system*

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \nu \partial_x (D(\boldsymbol{U})\partial_x \boldsymbol{U}) + \boldsymbol{G}(\boldsymbol{U}) + O(\nu^2)$$

with $A(\boldsymbol{U})$ the Jacobian of $\boldsymbol{F}(\boldsymbol{U})$.

Proof. The second set of equations is equivalent to

$$\boldsymbol{V} = \boldsymbol{F}(\boldsymbol{U}) - \epsilon R(\boldsymbol{U})^{-1}(\partial_t \boldsymbol{V} + \alpha^2 \partial_x \boldsymbol{U} + \boldsymbol{H}(\boldsymbol{U})).$$

Therefore $\partial_t \boldsymbol{V} = \partial_t \boldsymbol{F}(\boldsymbol{U}) + O(\epsilon)$. Now we multiply the first set of equations by $A(\boldsymbol{U})$ to obtain

$$\begin{aligned}
&A(\boldsymbol{U})(\partial_t \boldsymbol{U} + \partial_x \boldsymbol{V}) = A(\boldsymbol{U})\boldsymbol{G}(\boldsymbol{U}) \\
&A(\boldsymbol{U})(\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U})) = A(\boldsymbol{U})\boldsymbol{G}(\boldsymbol{U}) + O(\epsilon) \\
&\partial_t \boldsymbol{F}(\boldsymbol{U}) + A(\boldsymbol{U})\partial_x(\boldsymbol{F}(\boldsymbol{U})) = A(\boldsymbol{U})\boldsymbol{G}(\boldsymbol{U}) + O(\epsilon) \\
&\partial_t \boldsymbol{F}(\boldsymbol{U}) + A(\boldsymbol{U})^2 \partial_x \boldsymbol{U} = A(\boldsymbol{U})\boldsymbol{G}(\boldsymbol{U}) + O(\epsilon)
\end{aligned}$$

therefore

$$\boldsymbol{V} = \boldsymbol{F}(\boldsymbol{U}) - \epsilon R(\boldsymbol{U})^{-1}\left(\left(\alpha^2 I_d - A(\boldsymbol{U})^2\right)\partial_x \boldsymbol{U} + A(\boldsymbol{U})\boldsymbol{G}(\boldsymbol{U}) - \boldsymbol{H}(\boldsymbol{U})\right)$$

Taking $\boldsymbol{H}(U) = A(\boldsymbol{U})\boldsymbol{G}(\boldsymbol{U})$, $\epsilon = \nu$ and $R^{-1} = (\alpha^2 - A(\boldsymbol{U})^2)D(\boldsymbol{U})^{-1}$ we obtain the result □

All in all, the relaxation model (6) is an approximation of the full initial model (2) with an error homogenous to the square of the diffusion parameter. This is a limitation of the method which can only be applied when diffusion is small.

2.3 First order time scheme

The main advantage of the relaxation model is to split the spatial part from to the relaxation part we obtain a algorithm where the two parts are easy to solve. Indeed the relaxation step is local and during the transport step we solve N independent linear wave equations. We will now prove formally the consistency of the first order time splitting with the limit system. Firstly, we consider only the relaxation step

$$\begin{cases} \partial_t \boldsymbol{U} = 0 \\ \partial_t \boldsymbol{V} = \dfrac{R(\boldsymbol{U})}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) + \boldsymbol{H}(\boldsymbol{U}) \end{cases} \tag{7}$$

Since the relaxation source term is stiff and since we want to avoid a CFL condition we use a standard implicit scheme. We obtain

$$\begin{cases} \boldsymbol{U}^{n+1} = \boldsymbol{U}^n \\ \boldsymbol{V}^{n+1} = \boldsymbol{V}^n + \Delta t \frac{R(\boldsymbol{U}^{n+1})}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}^{n+1}) - \boldsymbol{V}^{n+1}) + \Delta t \boldsymbol{H}(\boldsymbol{U}^{n+1}) \end{cases} \tag{8}$$

which is exactly equal to

$$\begin{cases} \boldsymbol{U}^{n+1} = \boldsymbol{U}^n \\ \boldsymbol{V}^{n+1} = \boldsymbol{V}^n + \Delta t \frac{R(\boldsymbol{U}^n)}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}^n) - \boldsymbol{V}^{n+1}) + \Delta t \boldsymbol{H}(\boldsymbol{U}^n) \end{cases}$$

Plugging the first equation of (8) in the second equation of (8) we obtain

$$\left(\frac{\epsilon + \Delta t R(\boldsymbol{U}^n)}{\epsilon}\right)\boldsymbol{V}^{n+1} = \boldsymbol{V}^n + \frac{\Delta t}{\epsilon} R\boldsymbol{F}(\boldsymbol{U}^n) + \Delta t \boldsymbol{H}(\boldsymbol{U}^n)$$

We remark that using the conservation of the primitive variables during the relaxation step we avoid to invert a nonlinear local problem. Eventually we obtain

$$\boldsymbol{V}^{n+1} = \left(\frac{\epsilon + \Delta t R(\boldsymbol{U}^n)}{\epsilon}\right)^{-1} \boldsymbol{V}^n + \Delta t \left(\frac{\epsilon + \Delta t R(\boldsymbol{U}^n)}{\epsilon}\right)^{-1} \left(\frac{R(\boldsymbol{U}^n)}{\epsilon}\boldsymbol{F}(\boldsymbol{U}^n) + \boldsymbol{H}(\boldsymbol{U}^n)\right)$$

Now we introduce the full first order splitting scheme. In this case we propose to split the physical source from the other step but the source can also be introduced in the transport or in the relaxation step without any particular difficulties.

Definition 2.1. *The first order splitting scheme for the system (6) is given by*

$$T(\Delta t) \circ S(\Delta t) \circ Re(\Delta t)\boldsymbol{V}^{n+1} = \boldsymbol{V}^n \tag{9}$$

with

$$T(\Delta t) = \begin{cases} \boldsymbol{U}^{**} + \Delta t \partial_x \boldsymbol{V}^{**} = \boldsymbol{U}^* \\ \boldsymbol{V}^{**} + \alpha^2 \Delta t \partial_x \boldsymbol{U}^{**} = \boldsymbol{V}^* \end{cases} \tag{10}$$

$$S(\Delta t) = \begin{cases} \boldsymbol{U}^{**} - \Delta t \boldsymbol{G}(\boldsymbol{U}^{**}) = \boldsymbol{U}^* \\ \boldsymbol{V}^{**} = \boldsymbol{V}^* \end{cases} \tag{11}$$

$$Re(\Delta t) = \begin{cases} \boldsymbol{U}^{**} = \boldsymbol{U}^* \\ \boldsymbol{V}^{**} = \left(\frac{\epsilon + \Delta t R(\boldsymbol{U}^*)}{\epsilon}\right)^{-1} \boldsymbol{V}^* + \Delta t \left(\frac{\epsilon + \Delta t R(\boldsymbol{U}^*)}{\epsilon}\right)^{-1} \left(\frac{R}{\epsilon}\boldsymbol{F}(\boldsymbol{U}^*) + \boldsymbol{H}(\boldsymbol{U}^*)\right) \end{cases} \tag{12}$$

Following the introduction of the time scheme, we propose a formal study of the consistency. To that end we begin with a formal lemma on the solution to the relaxation system.

Lemma 2.3. *We assume the regularity of the solutions. The solutions of the system (6) satisfy*

$$\begin{cases} \partial_{tt}\boldsymbol{U} = \partial_x(\alpha^2 \partial_x \boldsymbol{U}) - \partial_x\left(\frac{R(\boldsymbol{U})}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V})\right) + \partial_t \boldsymbol{G}(\boldsymbol{U}) - \partial_x \boldsymbol{H}(U) \\ \\ \partial_{tt}\boldsymbol{V} = \partial_x(\alpha^2 \partial_x \boldsymbol{V}) + \partial_t \boldsymbol{H}(\boldsymbol{U}) - \partial_x \boldsymbol{G}(\boldsymbol{U}) - \frac{R(\boldsymbol{U})^2}{\epsilon^2}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) \\ + \frac{R(\boldsymbol{U})}{\epsilon}\left(\alpha^2 \partial_x \boldsymbol{U} - A(\boldsymbol{U})\partial_x \boldsymbol{V}\right) + \frac{\partial_t R(\boldsymbol{U})}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) \end{cases} \tag{13}$$

Proof. To obtain this result we apply a time derivative on the first equation and a spatial derivative on the second one. We obtain

$$\begin{cases} \partial_{tt}\boldsymbol{U} + \partial_{tx}\boldsymbol{V} = \partial_t \boldsymbol{G}(\boldsymbol{U}) \\ \partial_{tx}\boldsymbol{V} + \partial_x(\alpha^2 \partial_x \boldsymbol{U}) = \partial_x\left(\frac{R(\boldsymbol{U})}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V})\right) + \partial_x \boldsymbol{H}(\boldsymbol{U}) \end{cases} \tag{14}$$

We then compute $\partial_{tx}\boldsymbol{V}$ using the second equation and we plug this result in the first equation of (14). We obtain the first result. To obtain the second one we use the same computation but derivating in time the equation on $\boldsymbol{V}$ and in space the equation on $\boldsymbol{U}$. To obtain this result we have used that $\boldsymbol{H}(\boldsymbol{U}) = A(\boldsymbol{U})G(\boldsymbol{U})$. □

Now we propose a formal result on the consistency between the first order splitting scheme and the limit model (2).

Lemma 2.4. *The scheme (9) - (10) - (11) - (12) is consistent at the limit with the model*

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \nu \partial_x (D(\boldsymbol{U}) \partial_x \boldsymbol{U}) + \boldsymbol{G}(\boldsymbol{U}) + \mathbf{E} + O(\epsilon \Delta t + \Delta t^2 + \epsilon^2) \tag{15}$$

with

$$\mathbf{E} = \frac{\Delta t}{2} \partial_x (\alpha^2 \partial_x \boldsymbol{U}) + \frac{\Delta t}{2} \partial_x \left(\left(\alpha^2 I_d - A(\boldsymbol{U})^2 \right) \partial_x \boldsymbol{U} \right) + \frac{\Delta t}{2} \left(\partial_t \boldsymbol{G}(\boldsymbol{U}) + \partial_x \boldsymbol{H}(\boldsymbol{U}) \right) \tag{16}$$

Proof. The splitting scheme with the first order term error can be written in the following form

- **First step**: $T_{\Delta t} \begin{pmatrix} \boldsymbol{U}^* \\ \boldsymbol{V}^* \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}^n \\ \boldsymbol{V}^n \end{pmatrix}$
- **Second step**: $S_{\Delta t} \begin{pmatrix} \boldsymbol{U}^{**} \\ \boldsymbol{V}^{**} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}^* \\ \boldsymbol{V}^* \end{pmatrix}$
- **Third step**: $R_{\Delta t} \begin{pmatrix} \boldsymbol{U}^{n+1} \\ \boldsymbol{V}^{n+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}^{**} \\ \boldsymbol{V}^{**} \end{pmatrix}$

with

$$T_{\Delta t} = I_d + \Delta t \begin{pmatrix} \partial_x I_d^v \\ \alpha^2 \partial_x I_d^u \end{pmatrix}$$

$$S_{\Delta t} = I_d + \Delta t \begin{pmatrix} \boldsymbol{G}(I_d^u) \\ 0 \end{pmatrix}$$

$$R_{\Delta t} = I_d + \Delta t \begin{pmatrix} 0 \\ -\frac{R}{\epsilon} \boldsymbol{F}(I_d^u) + \frac{R(I_d^u)}{\epsilon} I_d^v - \boldsymbol{H}(I_d^u) \end{pmatrix}$$

At the splitting scheme is given by

$$T_{\Delta t} \circ S_{\Delta t} \circ R_{\Delta t} \begin{pmatrix} \boldsymbol{U}^{n+1} \\ \boldsymbol{V}^{n+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}^n \\ \boldsymbol{V}^n \end{pmatrix} \tag{17}$$

Now (we propose to) compute the operator $T_{\Delta t} \circ S_{\Delta t} \circ R_{\Delta t}$.

$$T_{\Delta t} \circ S_{\Delta t} \circ R_{\Delta t} = I_d + \Delta t \begin{pmatrix} \partial_x \\ \alpha^2 \partial_x \end{pmatrix} + \Delta t \begin{pmatrix} \boldsymbol{G}(I_d^u) \\ 0 \end{pmatrix} + \Delta t \begin{pmatrix} 0 \\ -\frac{R(I_d^u)}{\epsilon} \boldsymbol{F}(I_d^u) + \frac{R}{\epsilon} I_d^v - \boldsymbol{H}(I_d^u) \end{pmatrix} + E_s$$

with the splitting error E_s given by

$$E_s = \Delta t^2 \begin{pmatrix} -\frac{R(I_d^u)}{\epsilon} \partial_x (\boldsymbol{F}(I_d^u) - I_d^v) - \frac{\partial_x R(I_d^u)}{\epsilon} (\boldsymbol{F}(I_d^u) - I_d^v) - \partial_x \boldsymbol{H}(I_d) \\ 0 \end{pmatrix} + \Delta t^2 \begin{pmatrix} 0 \\ \alpha^2 \partial_x \boldsymbol{G}(I_d^u) \end{pmatrix}$$

Let us study the equivalent equation associated with the scheme (17). To this end we use a first order Taylor expansion (which is)

$$\frac{f^{n+1}-f^n}{\Delta t} = \partial_t f + \frac{\Delta t}{2}\partial_{tt} f.$$

Applying this extension for each variable, we obtain the following equivalent equation for the splitting scheme:

$$\begin{cases} \partial_t \boldsymbol{U} + \partial_x \boldsymbol{V} = \boldsymbol{G}(\boldsymbol{U}) + \Delta t \partial_x \left(\dfrac{R(\boldsymbol{U})}{\epsilon} (\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) \right) + \Delta t \partial_x \boldsymbol{H}(\boldsymbol{U}) + \dfrac{\Delta t}{2} \partial_{tt} \boldsymbol{U} + O(\Delta t^2) \\ \\ \partial_t \boldsymbol{V} + \alpha^2 \partial_x \boldsymbol{U} = \dfrac{R(\boldsymbol{U})}{\epsilon} (\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) + \boldsymbol{H}(\boldsymbol{U}) - \Delta t \alpha^2 \partial_x \boldsymbol{G}(\boldsymbol{U}) + \dfrac{\Delta t}{2} \partial_{tt} \boldsymbol{V} + O(\Delta t^2) \end{cases}$$

Plugging the result in the second derivative in the previous system, we obtain

$$\begin{cases} \partial_t \boldsymbol{U} + \partial_x \boldsymbol{V} = \boldsymbol{G}(\boldsymbol{U}) + \dfrac{\Delta t}{2} \partial_x (\alpha^2 \partial_x \boldsymbol{U}) + \dfrac{\Delta t}{2} \partial_x \left(\dfrac{R(\boldsymbol{U})}{\epsilon} (\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) \right) \\ \\ + \dfrac{\Delta t}{2} \left(\partial_t \boldsymbol{G}(\boldsymbol{U}) + \partial_x \boldsymbol{H}(\boldsymbol{U}) \right) + O(\Delta t^2) \\ \\ \partial_t \boldsymbol{V} + \alpha^2 \partial_x \boldsymbol{U} = \dfrac{R(\boldsymbol{U})}{\epsilon} (\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) + \boldsymbol{H}(\boldsymbol{U}) + \dfrac{\Delta t}{2} \partial_x (\alpha^2 \partial_x \boldsymbol{V}) + \dfrac{\Delta t}{2} \dfrac{R(\boldsymbol{U})}{\epsilon} \left(\alpha^2 \partial_x \boldsymbol{U} - A(\boldsymbol{U}) \partial_x \boldsymbol{V} \right) \\ \\ + \dfrac{\Delta t}{2} \dfrac{\partial_t R(\boldsymbol{U})}{\epsilon} (\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) - \dfrac{\Delta t}{2} \dfrac{R(\boldsymbol{U})^2}{\epsilon^2} (\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) - \dfrac{\Delta t}{2} \left(\partial_t \boldsymbol{H}(\boldsymbol{U}) - (1+\alpha^2) \partial_x \boldsymbol{G}(\boldsymbol{U}) \right) + O(\Delta t^2) \end{cases}$$

Now we (propose to) use a Chapman Enskog expansion given by $\boldsymbol{U} = \boldsymbol{U}_0 + \epsilon \boldsymbol{U}_1 + \epsilon^2 \boldsymbol{U}_2 + O(\epsilon^3)$ and the same for $\boldsymbol{V}$.

- **Term in ϵ^2:** $\quad \boldsymbol{F}(\boldsymbol{U}_0) - \boldsymbol{V}_0 = 0$
- **Term in ϵ:**

$$\begin{cases} \partial_x (R(\boldsymbol{U}_0)(\boldsymbol{F}(\boldsymbol{U}_0) - \boldsymbol{V}_0)) = 0 \\ R(\boldsymbol{U}_0) \left(\alpha^2 \partial_x \boldsymbol{U}_0 - A(\boldsymbol{U}_0) \partial_x \boldsymbol{V}_0 \right) - R(\boldsymbol{U}_0)^2 (A(\boldsymbol{U}_0) \boldsymbol{U}_1 - \boldsymbol{V}_1) = 0 \end{cases} \tag{18}$$

The term with $\partial_x R(\boldsymbol{U}_0)$ cancels out since with have $\boldsymbol{F}(\boldsymbol{U}_0) - \boldsymbol{V}_0 = 0$. We also use that $\boldsymbol{H}(\boldsymbol{U}) = A(\boldsymbol{U})\boldsymbol{G}(\boldsymbol{U})$. Using the second equation of (18) we obtain that

$$\begin{aligned} (A(\boldsymbol{U}_0)\boldsymbol{U}_1 - \boldsymbol{V}_1) &= R^{-1}(\boldsymbol{U}_0) \left(\alpha^2 \partial_x \boldsymbol{U}_0 - A(\boldsymbol{U}_0) \partial_x \boldsymbol{V}_0 \right) \\ (A(\boldsymbol{U}_0)\boldsymbol{U}_1 - \boldsymbol{V}_1) &= R^{-1}(\boldsymbol{U}_0) \left(\alpha^2 \partial_x \boldsymbol{U}_0 - A(\boldsymbol{U}_0) \partial_x \boldsymbol{F}(\boldsymbol{U}_0) \right) \\ (A(\boldsymbol{U}_0)\boldsymbol{U}_1 - \boldsymbol{V}_1) &= R^{-1}(\boldsymbol{U}_0) \left(\alpha^2 I_d - A(\boldsymbol{U}_0)^2 \right) \partial_x \boldsymbol{U}_0 \end{aligned}$$

- **Term in** $O(1)$:

$$\begin{cases} \partial_t \boldsymbol{U}_0 + \partial_x \boldsymbol{F}(\boldsymbol{U}_0) = \boldsymbol{G}(\boldsymbol{U}_0) + \dfrac{\Delta t}{2}\partial_x(\alpha^2\partial_x \boldsymbol{U}_0) + \dfrac{\Delta t}{2}\left(\partial_x\left(R(\boldsymbol{U}_0)A(\boldsymbol{U}_0)\boldsymbol{U}_1 - \partial_x \boldsymbol{V}_1\right)\right) + \dfrac{\Delta t}{2}\left(\partial_t \boldsymbol{G}(\boldsymbol{U}_0) + \partial_x \boldsymbol{H}(\boldsymbol{U}_0)\right) + O(\Delta t^2) \\ \partial_t \boldsymbol{V}_0 + \alpha^2\partial_x \boldsymbol{U}_0 = \dfrac{\Delta t}{2}\partial_x(\alpha^2\partial_x \boldsymbol{V}_0) + R(\boldsymbol{U}_0)\left(A(\boldsymbol{U}_0)\boldsymbol{U}_1 - \boldsymbol{V}_1\right) + \boldsymbol{H}(\boldsymbol{U}_0) - \dfrac{\Delta t}{2}R(\boldsymbol{U}_0)^2\left(A^{'}(\boldsymbol{U}_0)\boldsymbol{U}_2 - \boldsymbol{V}_2\right) \\ -\dfrac{\Delta t}{2}\left(R^{'}(\boldsymbol{U}_0)R(\boldsymbol{U}_0) + R(\boldsymbol{U}_0)R^{'}(\boldsymbol{U}_0)\right)\boldsymbol{U}_1\left(A^{'}(\boldsymbol{U}_0)\boldsymbol{U}_1 - \boldsymbol{V}_1\right) + \dfrac{\Delta t}{2}R\left(\alpha^2\partial_x \boldsymbol{U}_1 - A(\boldsymbol{U}_0)\partial_x \boldsymbol{V}_1 - A(\boldsymbol{U}_1)\partial_x \boldsymbol{V}_0\right) \\ -\frac{\Delta t}{2}\left(\partial_t \boldsymbol{H}(\boldsymbol{U}_0) - (1+\alpha^2)\partial_x \boldsymbol{G}(\boldsymbol{U}_0)\right) \end{cases}$$

Eventually we plug the expression of the term $(A(\boldsymbol{U}_0)\boldsymbol{U}_1 - \boldsymbol{V}_1)$ in the first equation to obtain

$$\partial_t \boldsymbol{U}_0 + \partial_x \boldsymbol{F}(\boldsymbol{U}_0) = \boldsymbol{G}(\boldsymbol{U}_0) + \frac{\Delta t}{2}\partial_x(\alpha^2\partial_x \boldsymbol{U}_0) + \frac{\Delta t}{2}\partial_x\left(\left(\alpha^2 I_d - A(\boldsymbol{U}_0)^2\right)\partial_x \boldsymbol{U}_0\right) + \frac{\Delta t}{2}\left(\partial_t \boldsymbol{G}(\boldsymbol{U}_0) + \partial_x \boldsymbol{H}(\boldsymbol{U}_0)\right) + O(\Delta t^2)$$

Now we consider the term homogeneous to ϵ.

Term in $O(\epsilon)$:

$$\partial_t \boldsymbol{U}_1 + \partial_x \boldsymbol{V}_1 = \boldsymbol{G}^{'}(\boldsymbol{U}_0)\boldsymbol{U}_1 + O(\Delta t)$$

Using the relation between $\boldsymbol{U}_1$ and $\boldsymbol{U}_0$ we obtain

$$\partial_t \boldsymbol{U}_1 + \partial_x(A(\boldsymbol{U}_0)\boldsymbol{U}_1) - \partial_x\left(R^{-1}(\alpha^2 I_d - A(\boldsymbol{U}_0)^2)\partial_x \boldsymbol{U}_0\right) = O(\Delta t)$$

The limit equation is given by

$$\begin{aligned} &\partial_t(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1) + \partial_x\left(\boldsymbol{F}(\boldsymbol{U}_0) + \epsilon A(\boldsymbol{U}_0)\boldsymbol{U}_1\right) = \boldsymbol{G}(\boldsymbol{U}_0) + \epsilon\boldsymbol{G}^{'}(\boldsymbol{U}_0)\boldsymbol{U}_1 + \frac{\Delta t}{2}\partial_x(\alpha^2\partial_x \boldsymbol{U}_0) \\ &+ \frac{\Delta t}{2}\partial_x\left(\left(\alpha^2 I_d - A(\boldsymbol{U}_0)^2\right)\partial_x \boldsymbol{U}_0\right) + \epsilon\partial_x\left(R(\boldsymbol{U}_0)^{-1}\left(\alpha^2 I_d - A(\boldsymbol{U}_0)^2\right)\partial_x \boldsymbol{U}_0\right) \\ &+ \frac{\Delta t}{2}\left(\partial_t \boldsymbol{G}(\boldsymbol{U}_0) + \partial_x \boldsymbol{H}(\boldsymbol{U}_0)\right) + O(\epsilon\Delta t + \Delta t^2) \\ &\partial_t(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1) + \partial_x\left(\boldsymbol{F}(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1)\right) = \boldsymbol{G}(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1) + \frac{\Delta t}{2}\partial_x(\alpha^2\partial_x \boldsymbol{U}_0) \\ &+ \frac{\Delta t}{2}\partial_x\left(\left(\alpha^2 I_d - A(\boldsymbol{U}_0)^2\right)\partial_x \boldsymbol{U}_0\right) + \epsilon\partial_x\left(R(\boldsymbol{U}_0)^{-1}\left(\alpha^2 I_d - A(\boldsymbol{U}_0)^2\right)\partial_x \boldsymbol{U}_0\right) \\ &+ \frac{\Delta t}{2}\left(\partial_t \boldsymbol{G}(\boldsymbol{U}_0) + \partial_x \boldsymbol{H}(\boldsymbol{U}_0)\right) + O(\epsilon\Delta t + \Delta t^2) \\ &\partial_t(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1) + \partial_x\left(\boldsymbol{F}(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1)\right) = \boldsymbol{G}(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1) + \frac{\Delta t}{2}\partial_x(\alpha^2\partial_x(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1)) \\ &+ \frac{\Delta t}{2}\partial_x\left(\left(\alpha^2 I_d - A(\boldsymbol{U}_0)^2\right)\partial_x(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1)\right) + \epsilon\partial_x\left(R(\boldsymbol{U}_0)^{-1}\left(\alpha^2 I_d - A(\boldsymbol{U}_0)^2\right)\partial_x(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1)\right) \\ &+ \frac{\Delta t}{2}\left(\partial_t \boldsymbol{G}(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1) + \partial_x \boldsymbol{H}(\boldsymbol{U}_0 + \epsilon\boldsymbol{U}_1)\right) + O(\epsilon\Delta t + \Delta t^2) \end{aligned}$$

consequently at the end we have

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \boldsymbol{G}(\boldsymbol{U}) + \frac{\Delta t}{2}\partial_x(\alpha^2 \partial_x \boldsymbol{U}) + \frac{\Delta t}{2}\partial_x\left(\left(\alpha^2 I_d - A(\boldsymbol{U})^2\right)\partial_x \boldsymbol{U}\right) + \epsilon\partial_x\left(R(\boldsymbol{U})^{-1}\left(\alpha^2 I_d - A(\boldsymbol{U})^2\right)\partial_x \boldsymbol{U}\right)$$
$$+ \frac{\Delta t}{2}\left(\partial_t \boldsymbol{G}(\boldsymbol{U}) + \partial_x \boldsymbol{H}(\boldsymbol{U})\right) + O(\epsilon\Delta t + \Delta t^2 + \epsilon^2)$$

Eventually we use that $\epsilon = \nu$ and $R(\boldsymbol{U}_0) = \left(\alpha^2 I_d - A(\boldsymbol{U})^2\right) D(\boldsymbol{U}_0)^{-1}$, we obtain the final result. □

Using the same argument as for the relaxation model and without source the stability of the limit system at the first order is insured if α is sufficiently large. We also remark that the generalization of the relaxation term (with a matrix R) allows to modify the physical diffusion (homogeneous to ϵ) but does not modify the numerical diffusion induced by the splitting.

2.4 High order time scheme

In this section we present an implicit method which allows to use very large time steps to capture steady states or to filter fast scales. However with a first order scheme and large time step we obtain a too diffusive method. To avoid this issue it will be essential to increase the order in time of the method . For this we use a second order implicit scheme for each step (a Cranck-Nicolson scheme, written as a θ scheme) scheme and combine this with a second order Strang splitting. Before writing the full scheme, we rewrite the Relaxation step as for the first order case.

$$\begin{cases} \boldsymbol{U}^{n+1} = \boldsymbol{U}^n \\ \boldsymbol{V}^{n+1} = \boldsymbol{V}^n + \theta\Delta t\frac{R}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}^{n+1}) - \boldsymbol{V}^{n+1}) + (1-\theta)\Delta t\frac{R}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}^n) - \boldsymbol{V}^n) + \theta\Delta t\boldsymbol{H}(\boldsymbol{U}^{n+1}) + (1-\theta)\Delta t\boldsymbol{H}(\boldsymbol{U}^n) \end{cases}$$

Plugging the first equation in the second we obtain

$$\left(\frac{\epsilon + \Delta t\theta R}{\epsilon}\right)\boldsymbol{V}^{n+1} = \left(1 - (1-\theta)\Delta t\frac{R}{\epsilon}\right)\boldsymbol{V}^n + \frac{\Delta t}{\epsilon}R\boldsymbol{F}(\boldsymbol{U}^n) + \Delta t\boldsymbol{H}(\boldsymbol{U}^n)$$

We remark that using the conservation of the primitive variables during the relaxation step we avoid to invert a nonlinear local problem. To finish we obtain

$$\boldsymbol{V}^{n+1} = \left(\frac{\epsilon + \theta\Delta tR}{\epsilon}\right)^{-1}\left(\left(1 - (1-\theta)\frac{\Delta tR}{\epsilon}\right)\boldsymbol{V}^n + \frac{\Delta tR}{\epsilon}\boldsymbol{F}(\boldsymbol{U}^n) + \Delta t\boldsymbol{H}(\boldsymbol{U}^n)\right)$$

As for the first order scheme, we introduced the full scheme in the next definition.

Definition 2.2. *The second order splitting scheme for the system (6) is given by*

$$T\left(\frac{\Delta t}{2}\right) \circ S\left(\frac{\Delta t}{2}\right) \circ R(\Delta t) \circ S\left(\frac{\Delta t}{2}\right) \circ T\left(\frac{\Delta t}{2}\right)\boldsymbol{V}^{n+1} = \boldsymbol{V}^n \tag{19}$$

with

$$T(\Delta t) : \begin{cases} \boldsymbol{U}^{**} + \theta\Delta t\partial_x \boldsymbol{V}^{**} = \boldsymbol{U}^* - (1-\theta)\Delta t\partial_x \boldsymbol{V}^* \\ \boldsymbol{V}^{**} + \theta\alpha^2\Delta t\partial_x \boldsymbol{U}^{**} = \boldsymbol{V}^* - \alpha^2(1-\theta)\Delta t\partial_x \boldsymbol{U}^* \end{cases} \tag{20}$$

$$S(\Delta t) : \begin{cases} \boldsymbol{U}^{**} - \theta \Delta t \boldsymbol{G}(\boldsymbol{U}^{**}) = \boldsymbol{U}^{*} + (1-\theta)\Delta t \boldsymbol{G}(\boldsymbol{U}^{*}) \\ \boldsymbol{V}^{**} = \boldsymbol{V}^{*} \end{cases} \tag{21}$$

$$Re(\Delta t) = \begin{cases} \boldsymbol{U}^{**} = \boldsymbol{U}^{*} \\ \boldsymbol{V}^{**} = M\left(\left(1 - (1-\theta)\frac{\Delta t R(\boldsymbol{U}^{*}))}{\epsilon}\right)\boldsymbol{V}^{*} + \frac{\Delta t R(\boldsymbol{U}^{*})}{\epsilon}\boldsymbol{F}(\boldsymbol{U}^{*}) + \Delta t \boldsymbol{H}(\boldsymbol{U}^{*})\right) \end{cases} \tag{22}$$

with

$$M = \left(\frac{\epsilon + \theta \Delta t R(\boldsymbol{U}^{*})}{\epsilon}\right)^{-1}$$

Let us state a few remarks

(i) In order to obtain the second order scheme we need to take $\theta = 0.5$ for all steps.

(ii) In [9] the authors show the second order Strang splitting does not converge at the second order for small ϵ. However,as will we show numerically in the following, the scheme converges with second order accuracy. The fact that the splitting scheme proposed here has better properties than the standard one comes from probably from the fact that each substep and consequently the full scheme is symmetric in time (the use Cranck-Nicolson scheme is a crucial point for this property to hold). Finally, adding implicit discretization we are able to obtain a second order splitting scheme for the stiff problem.

(iii) When the source term is equal to zero the splitting does not conserve $Spl(\Delta t = 0) = I_d$ (as explained in [6]). In order to increase the order in time using composition method [6], we need to preserve this property. The following splitting satisfy ($Spl(\Delta t = 0) = I_d$):

$$T\left(\frac{\Delta t}{4}\right) \circ S\left(\frac{\Delta t}{2}\right) \circ R\left(\frac{\Delta t}{2}\right) \circ T\left(\frac{\Delta t}{2}\right) \circ R\left(\frac{\Delta t}{2}\right) \circ S\left(\frac{\Delta t}{2}\right) \circ T\left(\frac{\Delta t}{4}\right)$$

3 Extension for multidimensional problem

In this section we introduce briefly the relaxation method and its two dimensional generalization. We don't detail the three dimensional case to avoid some complex notation but it is possible as explained in [9]. The number of additional variables induced by the relaxation method depends on the dimension. For the dimension d and for a system with N variables, we obtain a relaxation system with $N * (d + 1)$ variables.

3.1 General principle

We consider the following hyperbolic system with small diffusion.

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}_x(\boldsymbol{U}) + \partial_x \boldsymbol{F}_y(\boldsymbol{U}) = \nu \nabla \cdot (D(\boldsymbol{U}) \nabla \boldsymbol{U}) \tag{23}$$

with

$$D(\boldsymbol{U}) = \begin{pmatrix} D_{xx}(\boldsymbol{U}) & D_{xy}(\boldsymbol{U}) \\ D_{yx}(\boldsymbol{U}) & D_{yy}(\boldsymbol{U}) \end{pmatrix}$$

As previously we apply the relaxation method to the hyperbolic part of the system. As stated before we propose to write a relaxation scheme for the full velocity vector and not for each component separately. We obtain

$$\begin{cases} \partial_t \boldsymbol{U} + \partial_x \boldsymbol{V}_x + \partial_y \boldsymbol{V}_y = 0 \\ \\ \partial_t \boldsymbol{V}_x + \alpha^2 B_{xx} \partial_x \boldsymbol{U} + \alpha^2 B_{xy} \partial_y \boldsymbol{U} = \dfrac{R_{xx}}{\epsilon}(\boldsymbol{F}_x(\boldsymbol{U}) - \boldsymbol{V}_x) + \dfrac{R_{xy}}{\epsilon}(\boldsymbol{F}_y(\boldsymbol{U}) - \boldsymbol{V}_y) \\ \partial_t \boldsymbol{V}_y + \alpha^2 B_{yx} \partial_x \boldsymbol{U} + \alpha^2 B_{yy} \partial_y \boldsymbol{U} = \dfrac{R_{yx}}{\epsilon}(\boldsymbol{F}_x(\boldsymbol{U}) - \boldsymbol{V}_x) + \dfrac{R_{yy}}{\epsilon}(\boldsymbol{F}_y(\boldsymbol{U}) - \boldsymbol{V}_y) \end{cases}, \tag{24}$$

with the relaxation matrix R and the transport matrix B given by

$$R = \begin{pmatrix} R_{xx} & R_{xy} \\ R_{yx} & R_{yy} \end{pmatrix}, \quad B = \begin{pmatrix} B_{xx} & B_{xy} \\ B_{yx} & B_{yy} \end{pmatrix},$$

where each sub-block in a $N \times N$ square matrix. The relaxation matrix R generalizes the relaxation method as was the case in one dimension. The introduction of the matrix B induces an additional degree of freedom which allows to change the structure of the diffusion in space and can be a way to reduce the diffusion or the dispersion at the numerical level. The standard choice is $B_{xx} = B_{yy} = I_d$ and $B_{xy} = B_{yx} = 0$.

Lemma 3.1. *The model (24) is consistent with the following system*

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}_x(\boldsymbol{U}) + \partial_y \boldsymbol{F}_y(\boldsymbol{U}) = \epsilon \nabla \cdot (\Omega^{-1}(\alpha^2 B - A^q) \nabla \boldsymbol{U}) + O(\epsilon^2)$$

Taking $\epsilon = \nu$ *and*

$$R = (\alpha^2 B - A^q) D(\boldsymbol{U})^{-1}$$

with

$$A^q = \begin{pmatrix} A_x^2 & A_x A_y \\ A_y A_x & A_y^2 \end{pmatrix}$$

This model is consistent with the original model (23)

Proof. Taking the first and second equation, we obtain that

$$R_{xx}(\boldsymbol{F}_x(\boldsymbol{U}) - \boldsymbol{V}_x) + R_{xy}(\boldsymbol{F}_y(\boldsymbol{U}) - \boldsymbol{V}_y) = \epsilon(\partial_t \boldsymbol{V}_x + \alpha^2(B_{xx}\partial_x \boldsymbol{U} + B_{xy}\partial_y \boldsymbol{U}))$$
$$R_{yx}(\boldsymbol{F}_x(\boldsymbol{U}) - \boldsymbol{V}_x) + R_{yy}(\boldsymbol{F}_y(\boldsymbol{U}) - \boldsymbol{V}_y) = \epsilon(\partial_t \boldsymbol{V}_y + \alpha^2(B_{yx}\partial_x \boldsymbol{U} + B_{yy}\partial_y \boldsymbol{U}))$$

which is equivalent to

$$\begin{pmatrix} \boldsymbol{V}_x \\ \boldsymbol{V}_y \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}_x \\ \boldsymbol{F}_y \end{pmatrix} - \epsilon R^{-1} \begin{pmatrix} \partial_t \boldsymbol{V}_x + \alpha^2(B_{xx}\partial_x \boldsymbol{U} + B_{xy}\partial_y \boldsymbol{U}) \\ \partial_t \boldsymbol{V}_y + \alpha^2(B_{yx}\partial_x \boldsymbol{U} + B_{yy}\partial_y \boldsymbol{U}) \end{pmatrix}$$

Now we use that $\boldsymbol{V}_x = \boldsymbol{F}_x + O(\epsilon)$ and the same for the y component. We obtain

$$\begin{pmatrix} \partial_t \boldsymbol{V}_x + \alpha^2(B_{xx}\partial_x \boldsymbol{U} + B_{xy}\partial_y \boldsymbol{U}) \\ \partial_t \boldsymbol{V}_y + \alpha^2(B_{yx}\partial_x \boldsymbol{U} + B_{yy}\partial_y \boldsymbol{U}) \end{pmatrix} = \begin{pmatrix} \partial_t \boldsymbol{F}_x + \alpha^2(B_{xx}\partial_x \boldsymbol{U} + B_{xy}\partial_y \boldsymbol{U}) \\ \partial_t \boldsymbol{F}_y + \alpha^2(B_{yx}\partial_x \boldsymbol{U} + B_{yy}\partial_y \boldsymbol{U}) \end{pmatrix}$$

Now we want obtain the equation for $\partial_t \boldsymbol{F}_x$ and $\partial_t \boldsymbol{F}_y$. For this we multiply

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{V}_x + \partial_y \boldsymbol{V}_y = 0$$

by $A_x(\boldsymbol{U}) = \partial \boldsymbol{F}_x(\boldsymbol{U})$ (resp $A_y(\boldsymbol{U})$) to obtain the following expression

$$\begin{pmatrix} \partial_t \boldsymbol{F}_x + A_x(\boldsymbol{U})\partial_x \boldsymbol{V}_x + A_x(\boldsymbol{U})\partial_y \boldsymbol{V}_y = 0 \\ \partial_t \boldsymbol{F}_y + A_y(\boldsymbol{U})\partial_x \boldsymbol{V}_x + A_y(\boldsymbol{U})\partial_y \boldsymbol{V}_y = 0 \end{pmatrix}$$

□

We we use again that $\boldsymbol{V}_x = \boldsymbol{F}_x + O(\epsilon)$ and we obtain that

$$\begin{pmatrix} \partial_t \boldsymbol{F}_x \\ \partial_t \boldsymbol{F}_y \end{pmatrix} + A\nabla \boldsymbol{U} = 0$$

Therefore we have at the end

$$\begin{pmatrix} \boldsymbol{V}_x \\ \boldsymbol{V}_y \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}_x \\ \boldsymbol{F}_y \end{pmatrix} - \epsilon R^{-1}(\alpha^2 B - A^q) \begin{pmatrix} \partial_x \boldsymbol{U} \\ \partial_y \boldsymbol{U} \end{pmatrix}$$

Using the procedure as before we can write an first and second order splitting scheme for the relaxation system. This writing procedure is valid for any system but it might be interesting to adapt parts of it to specific cases. In the following sections we will consider two such examples. Now we introduce briefly the splitting scheme and the consistency result.

Definition 3.1. *The first order splitting scheme for the system (24) is given by*

$$T(\Delta t) \circ Re(\Delta t)\boldsymbol{V}^{n+1} = \boldsymbol{V}^n \tag{25}$$

with

$$T(\Delta t) = \begin{cases} \boldsymbol{U}^{**} + \Delta t(\partial_x \boldsymbol{V}_x^{**} + \partial_y \boldsymbol{V}_y^{**}) = \boldsymbol{U}^* \\ \boldsymbol{V}_x^{**} + \alpha^2 \Delta t \partial_x \boldsymbol{U}^{**} = \boldsymbol{V}_x^* \\ \boldsymbol{V}_y^{**} + \alpha^2 \Delta t \partial_y \boldsymbol{U}^{**} = \boldsymbol{V}_y^* \end{cases} \tag{26}$$

$$Re(\Delta t) = \begin{cases} \boldsymbol{U}^{**} = \boldsymbol{U}^{*} \\ \boldsymbol{V}_x^{**} = \left(\dfrac{\epsilon + \Delta t R_{xx}(\boldsymbol{U}^{*})}{\epsilon}\right)^{-1} \left(\boldsymbol{V}_x^{*} + \Delta t \dfrac{R}{\epsilon} \boldsymbol{F}_x(\boldsymbol{U}^{*})\right) + \left(\dfrac{\epsilon + \Delta t R_{xy}(\boldsymbol{U}^{*})}{\epsilon}\right)^{-1} \left(\boldsymbol{V}_y^{*} + \Delta t \dfrac{R}{\epsilon} \boldsymbol{F}_y(\boldsymbol{U}^{*})\right) \\ \boldsymbol{V}_y^{**} = \left(\dfrac{\epsilon + \Delta t R_{xy}(\boldsymbol{U}^{*})}{\epsilon}\right)^{-1} \left(\boldsymbol{V}_x^{*} + \Delta t \dfrac{R}{\epsilon} \boldsymbol{F}_x(\boldsymbol{U}^{*})\right) + \left(\dfrac{\epsilon + \Delta t R_{yy}(\boldsymbol{U}^{*})}{\epsilon}\right)^{-1} \left(\boldsymbol{V}_y^{*} + \Delta t \dfrac{R}{\epsilon} \boldsymbol{F}_y(\boldsymbol{U}^{*})\right) \end{cases} \tag{27}$$

Lemma 3.2. *The scheme (25) - (26) - (27) is consistent at the limit with the model*

$$\partial_t \boldsymbol{U} + \nabla \cdot \boldsymbol{F}(\boldsymbol{U}) = \nu \nabla \cdot (D(\boldsymbol{U}) \nabla \boldsymbol{U}) + \mathbf{E} + O(\epsilon \Delta t + \Delta t^2 + \epsilon^2) \tag{28}$$

with

$$\mathbf{E} = \frac{\Delta t}{2} \nabla \cdot (\alpha^2 \nabla \boldsymbol{U}) + \frac{\Delta t}{2} \nabla \cdot \left(\left(\alpha^2 B - A(\boldsymbol{U})^q \right) \nabla \boldsymbol{U} \right) \tag{29}$$

Proof. The principle of the proof is exactly the same that in the one dimensional case. Consequently we do not give anymore detail. □

4 Implementation of the scheme and remarks

In this section we propose to discuss to the implementation of the various steps. In this section we will discuss the various steps of the splitting scheme, with a particular insistence on the transport one.

4.1 Implementation of the transport step

The best way to solve this step depends on the choice of the scheme used for the spatial discretization. Here we consider an Isogeometric Finite Element method based on high-order B-Splines. Indeed one of the planned applications of the scheme is to consider MHD flows in Tokamak. Those flows are smooth (low Mach, low β flows with diffusion) and the geometry is complex. This is why we consider this spatial discretization. In 2D or 3D the wave equation is not easy to invert for large values of the time step or large wave speed (driven by α) since the steady wave problem admits a null eigenvalue. Additionally, inverting a first-order problem is not easy with a finite element method. In the IGA/FE context the most simple operator to invert is the Laplacian. Indeed this a well-posed operator and there exist many very efficient solvers for this problem (multi-grids method, specific preconditioning based on tensor product for B-Splines etc).

4.1.1 Simple Boundary condition

First we will consider the simple boundary case. In this case we assume that the boundary conditions do not couple the different variables. In the following, we will present an algorithm which allows to solve the wave equation using Laplacian operators. We consider the transport

step in 2D (the principle is the same for higher numbers of dimensions) which is given by the following model

$$\begin{cases} \partial_t \boldsymbol{U} + \partial_x \boldsymbol{V}_x + \partial_y \boldsymbol{V}_y = 0 \\ \partial_t \boldsymbol{V}_x + \alpha^2 \partial_x \boldsymbol{U} = 0 \\ \partial_t \boldsymbol{V}_y + \alpha^2 \partial_y \boldsymbol{U} = 0 \end{cases} \tag{30}$$

We remark that each variable U^i is only coupled with the variables V_x^i and V_y^i. Therefore the system (30) can be rewritten a N independent systems given by

$$\begin{cases} \partial_t U^i + \partial_x V_x^i + \partial_y V_y^i = 0, \\ \partial_t V_x^i + \alpha^2 \partial_x U^i = 0, \\ \partial_t V_y^i + \alpha^2 \partial_x U^i = 0, \quad \forall i \in [1, .., N] \end{cases} \tag{31}$$

Firstly these N wave systems are independent and can be solved in parallel. For this we discretize each system with a θ scheme in time (which gives either the first order implicit scheme ($\theta = 1$) or the second order Cranck-Nicolson scheme ($\theta = \frac{1}{2}$). We obtain

$$\begin{pmatrix} I_d & \theta\Delta t\partial_x & \theta\Delta t\partial_y \\ \alpha^2\theta\Delta t\partial_x & I_d & 0 \\ \alpha^2\theta\Delta t\partial_y & I_d & 0 \end{pmatrix} \begin{pmatrix} U^i \\ V_x^i \\ V_y^i \end{pmatrix} = \begin{pmatrix} I_d & -(1-\theta)\Delta t\partial_x & -(1-\theta)\Delta t\partial_y \\ -\alpha^2(1-\theta)\Delta t\partial_x & I_d & 0 \\ -\alpha^2(1-\theta)\Delta t\partial_y & 0 & I_d \end{pmatrix} \begin{pmatrix} U^{i,n} \\ V_x^{i,n} \\ V_y^{i,n} \end{pmatrix} \tag{32}$$

The idea is not to solve directly this implicit problem but to use a Schur decomposition which approximates a block by block matrix by a three triangular block matrices. This method is commonly used for wave equations [4] - [11]. Applying a Schur decomposition to the implicit matrix, we obtain

$$\begin{pmatrix} I_d & \theta\Delta t\partial_x & \theta\Delta t\partial_y \\ \alpha^2\theta\Delta t\partial_x & I_d & 0 \\ \alpha^2\theta\Delta t\partial_y & 0 & I_d \end{pmatrix}^{-1}$$
$$= \begin{pmatrix} I_d & 0 & 0 \\ \alpha^2\theta\Delta t\partial_x & I_d & 0 \\ \alpha^2\theta\Delta t\partial_y & 0 & I_d \end{pmatrix} \begin{pmatrix} P^{-1} & 0 & 0 \\ 0 & I_d & 0 \\ 0 & 0 & I_d \end{pmatrix} \begin{pmatrix} I_d & \theta\Delta t\partial_x & \theta\Delta t\partial_y \\ 0 & I_d & 0 \\ 0 & 0 & I_d \end{pmatrix}$$

with the Schur complement $P = I_d - \theta^2\Delta t^2 \nabla\cdot(\nabla I_d)$. Applying this inverse, we obtain that solving the system (32) is equivalent to the algorithm for the i-th wave equation.

$$\begin{cases} \text{Predictor :} & \begin{pmatrix} V_x^{i,*} \\ V_y^{i,*} \end{pmatrix} = \begin{pmatrix} V_x^{i,n} \\ V_y^{i,n} \end{pmatrix} - (1-\theta)\Delta t\alpha \begin{pmatrix} \partial_x U^{i,n} \\ \partial_y U^{i,n} \end{pmatrix} \\ \text{Update :} & \left(I_d - \alpha^2\theta^2\Delta t^2 \nabla\cdot(\nabla\cdot)\right) U_i^{n+1} = -\theta\Delta t\left(\partial_x V_x^{i,*} + \partial_y V_y^{i,*}\right) + \left(U_i^n - (1-\theta)\Delta t(\partial_x V_x^{i,n} + \partial_y V_y^{i,n})\right) \\ \text{Corrector :} & \begin{pmatrix} V_x^{i,n+1} \\ V_y^{i,n+1} \end{pmatrix} = \begin{pmatrix} V_x^{i,*} \\ V_y^{i,*} \end{pmatrix} - \theta\Delta t\alpha \begin{pmatrix} \partial_x U^{i,n+1} \\ \partial_y U^{i,n+1} \end{pmatrix} \end{cases}$$

The Schur decomposition allows to replace the first order wave equation discretized in time by a time discretization of the second order wave equation. In this algorithm there are just

two operators to invert : the identity (which maps to a mass matrix after the finite element discretization and which gives diagonal or diagonal by block for some finite element or DG schemes) and classical diffusion operator (which maps a stiffness matrix after the finite element discretization). Inverting these two operators is easier than inverting the full wave problem when using an adapted algebraic preconditioning (a multi-grid method for instance). All in all, the transport part can be solved by only inverting very well-known and understood problems.
During the simulation the coefficient α can be dynamically adapted: ideally it should stay just high enough to insure stability. To avoid additional costs we do not want to have to reassemble the various matrices whenever the value of α is modified. Since we use iterative solvers to invert the different matrices, we must store only the stiffness matrix and the mass matrix and define a matrix-vector product as a linear combination of the two matrices depending on α and Δt.

4.1.2 Complex boundary conditions

The situation is different for more complex boundary conditions. If the various variables are not coupled by the boundary conditions we can use the previous algorithm, if not, we cannot solve the transport step with the same algorithm. In that case the algorithm is given by

$$\begin{cases} \text{Predictor :} & \begin{pmatrix} V_x^{i,*} \\ V_y^{i,*} \end{pmatrix} = \begin{pmatrix} V_x^{i,n} \\ V_y^{i,n} \end{pmatrix} - (1-\theta)\Delta t\alpha \begin{pmatrix} \partial_x U^{i,n} \\ \partial_y U^{i,n} \end{pmatrix}, \quad \forall i \in \{1, N\} \\ \text{Update :} & \left(I_d - \alpha^2\theta^2\Delta t^2 \nabla\cdot(\nabla I_d)\right)\boldsymbol{U}^{n+1} = -\theta\Delta t\left(\partial_x \boldsymbol{V}_x^* + \partial_y \boldsymbol{V}_y^*\right) + \left(\boldsymbol{U}^n - (1-\theta)\Delta t(\partial_x \boldsymbol{V}_x^n + \partial_y \boldsymbol{V}_y^n)\right) \\ \text{Corrector :} & \begin{pmatrix} V_x^{i,n+1} \\ V_y^{i,n+1} \end{pmatrix} = \begin{pmatrix} V_x^{i,*} \\ V_y^{i,*} \end{pmatrix} - \theta\Delta t\alpha \begin{pmatrix} \partial_x U^{i,n+1} \\ \partial_y U^{i,n+1} \end{pmatrix}, \quad \forall i \in \{1, N\} \end{cases}$$

The difference between this algorithm and the previous one lies in the fact that the diffusion equations (second step) must be solved simultaneously. In the first and third steps, each system (associated with the mass matrix) can be solved independently. Only in the second step do we lose the parallelism at the modeling level (ie. regardless of the possible spatial domain decomposition). Finally we replace a strongly coupled and ill-conditioning hyperbolic problem by a weakly coupled diffusion model of the same size and a large number of independent L^2 projections (mass matrices at the discrete level). Since the main problem to invert (the weakly coupled Laplacian) is easier to precondition and to invert we expect a reduction of the computing cost. In the future it might be beneficial to find an algorithm to solve the weakly coupled Laplacian operators which allows to keep some parallelism.

4.2 Relaxation and Source steps

These two steps of the algorithm are local to the cell. However, in the finite element context, the problem is not local but global and requires inverting the mass matrix. For the relaxation step we can write one system associated with each fictive variable V_i (all the variables can be treated in parallel). Indeed all the non-linearity is in the RHS. Consequently for each fictive variable we construct the RHS and invert the mass matrix. One defect of the current implementation is

that we build the RHS for each variable V_i independently consequently we build the relaxation matrix R at reach gauss point more often than necessary. For three dimensional cases this matrix has a size between 10 and 30 therefore we have an additional cost during the assembly of the RHS (which can be reduced by a partil precomputation) but we can solve the relaxation of each variable independently and in parallel. The parallel computation of the source step is similar as we can solve the mass system independently for each primitive variable. A Picard solver can be used in order to avoid the assembly of the mass matrix.

4.3 Remark on the parallelization

As in all standard finite element based codes domain decomposition can be applied to parallelize the matrices and preconditioning construction and the matrix vector product in the iterative solvers. One of the advantages of the relaxation method developed herein is that we can extract additional parallelism in each step of the algorithm between the models. Indeed for the relaxation and source step the systems associated with the mas matrix for each variable are independent and can be solved in parallel. The situation is similar for the transport part where the wave systems are independent whenever the boundary conditions do not couple variables. As stated before, when the boundary conditions induce a coupling between the different variables we lose parallelism only the "update " step of the transport algorithm where we solve the Laplacian. For this step standard parallelization schemes such as domain decomposition can be used. In the future it will be interesting to find an algorithm which allows to recover the parallelism for the transport step with complex boundary conditions.

5 Numerical results

In this section we validate our approach on a set of sample problems. We will first consider two one dimensional models: the viscid Burgers equation and the compressible Navier-Stokes equations. Then we will validate the method in two dimensions on the compressible isothermal Euler equations.

5.1 1D viscous Burgers

In this section we apply our relaxation scheme to the viscous Burgers equation. The model is given by

$$\partial_t \rho + \partial_x \left(\frac{\rho^2}{2} \right) = \partial_x (\nu \partial_x \rho) + f \tag{33}$$

The relaxation system is given by the following equation

$$\begin{cases} \partial_t \rho + \partial_x u = f \\ \partial_t u + \alpha^2 \partial_x \rho = \dfrac{1}{\epsilon} \left(\dfrac{\rho^2}{2} - u \right) + \rho f \end{cases} \tag{34}$$

with $\epsilon = \frac{\nu}{\alpha^2 - \rho^2}$. Now we will check whether the relaxation method works for this model and show that it is a competitive method.

Test 1:
We choose as a source term $f = g\rho$ in order to obtain a steady solution given by

$$\rho(t,x) = 1.0 + 0.1e^{-\frac{x^2}{\sigma}}, \quad g(t,x) = -\frac{2x}{\sigma}e^{-\frac{x^2}{\sigma}}$$

This test case allows to validate the order of convergence. We consider the final time $T = 0.1$, we consider a finely resolved mesh (10000 cells with third order polynomials). For this test case we consider a negligible viscosity with $\epsilon = 10^{-12}$ and $\alpha = 1.2$. The first and second order scheme are compared for different values of the time step. This test case allows to verify the

	Order 1		Order 2		Order 2 AP	
	Error	Order	Error	Order	Error	Order
$\Delta t = 0.02$	$1.5E^{-2}$	-	$3.6E^{-4}$	-	$2.2E^{-4}$	-
$\Delta t = 0.01$	$7.9E^{-3}$	0.92	$9.2E^{-5}$	1.97	$5.4E^{-5}$	2.02
$\Delta t = 0.005$	$4.1E^{-3}$	0.95	$2.3E^{-5}$	2.00	$1.3E^{-5}$	2.05
$\Delta t = 0.0025$	$2.1E^{-3}$	0.97	$5.8E^{-6}$	1.99	$3.3E^{-6}$	1.98
$\Delta t = 0.00125$	$1.0E^{-3}$	1.07	$1.4E^{-6}$	2.05	$8.3E^{-7}$	1.99

Table 1: Numerical error and order for the test 1 with the relaxation scheme.

time convergence (since we use a very fine discretization in space the time error dominates). We compare the first order (9) - (10) - (11) - (12), the second order splitting scheme (19) - (20) - (21) - (22) and the AP scheme. As expected the first and second order schemes converges with the correct order (tab 1) and the AP scheme also converge with the second order and a better constant which is also expected since we add steps. The fact that we can increase the order is very important since we want to be able large time steps while preserving a reasonable accuracy. Moreover, in this test case we consider a relaxation parameter ϵ close to zero: contrary to the standard time scheme of [9]-[10] for the relaxation, the Strang splitting coupled with the use of implicit time reversible substeps allows to obtain second order accuracy in time.

Test 2:
For this case the source term is set to zero. The spatial domain is $[0, 1]$, discretized over 10000 cells with third order polynomials. The initial condition is given by

$$\rho(0,x) = sin(2\pi x)$$

In this test case, the solution becomes discontinuous for large time values. This is a hard case for an implicit scheme combined with the use of large time steps. For this particular grid the explicit time step is around $\Delta t_{ex} = 10^{-5}$. We will show that we can apply the first and second order spltting schemes of our method with time step values far larger than the explicit one and

that the results are sufficiently accurate compared to the standard implicit scheme.
On (fig. 1) we compare, for one value of the time step $\Delta t = 0.001 \approx 100\Delta t_{ex}$ and $\nu = 10^{-3}$, the solution computed using respectively the first and second order schemes. Those results clearly

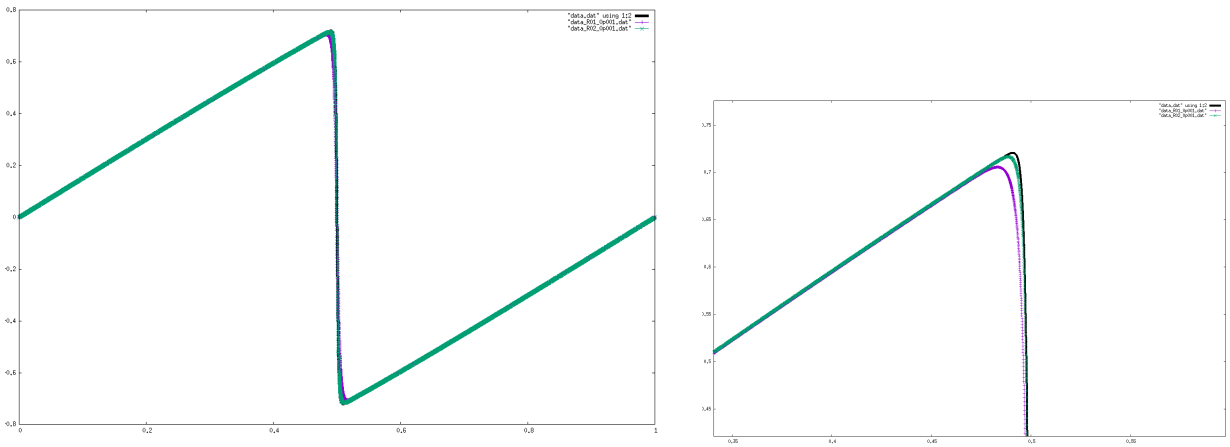

Figure 1: Burgers $1D$ Test 1; Left: reference numerical solution (black), numerical solution for first order (purple),second order (light blue) schemes for $\Delta t = 0.001$, Right: Zoom

show that with a time step far larger than the explicit one our relaxation scheme is able to capture the solution correctly. As expected the second order scheme captures the solution with the highest accuracy ((fig 1 right)).
On (fig 2) we show the results of first, second order and second order AP relaxation schemes for $\nu = 10^{-3}$ and for various values of the time step. On (fig 2 left) we observe that the first order scheme is able to compute the solution correctly for various values of the time step $\Delta t = 0.001$, $\Delta t = 0.005$ and $\Delta t = 0.01$. As expected the numerical diffusion associated with the relaxation method is slightly higher than with the standard implicit scheme. On (fig 2 right) we show the results for the second order scheme for two values of the time step $\Delta t = 0.001$ and $\Delta t = 0.005$. For the first value $\Delta t = 100\Delta t_{ex}$ the solution is computed with high accuracy. For a larger time step $\Delta t = 500\Delta t_{ex}$ the solution can still be captured but we observe an oscillation which comes from the numerical dispersion. For higher values of the time step, the second order method becomes unstable. The cause of this instability lies in the presence of numerical dispersion. When oscillations increase we need to increase α to insure the stability and when we increase α the numerical dispersion associated with the splitting increases as well. We can limit mitigate this effect but it is not possible to avoid the issue altogether.
Now we perform the same comparison for $\nu = 10^{-2}$. First, we observe on (Fig 3) shows that for a smooth problem the second order scheme remains stable for larger time step values (up to $1000\Delta t_{ex}$). Second, the scheme does not converge since the error between the relaxation model and the initial model is homogeneous to ν^2. All in all, this test case shows that the method is able to capture the numerical solution of the viscid Burgers equation using time step values far larger than the explicit one but the value of the time step cannot be increased as much as when using a fully nonlinear implicit scheme due to numerical dispersion. Moreover, as can be expected from the previous analysis, the relaxation scheme is ill-suited for the treatment of large ν cases.

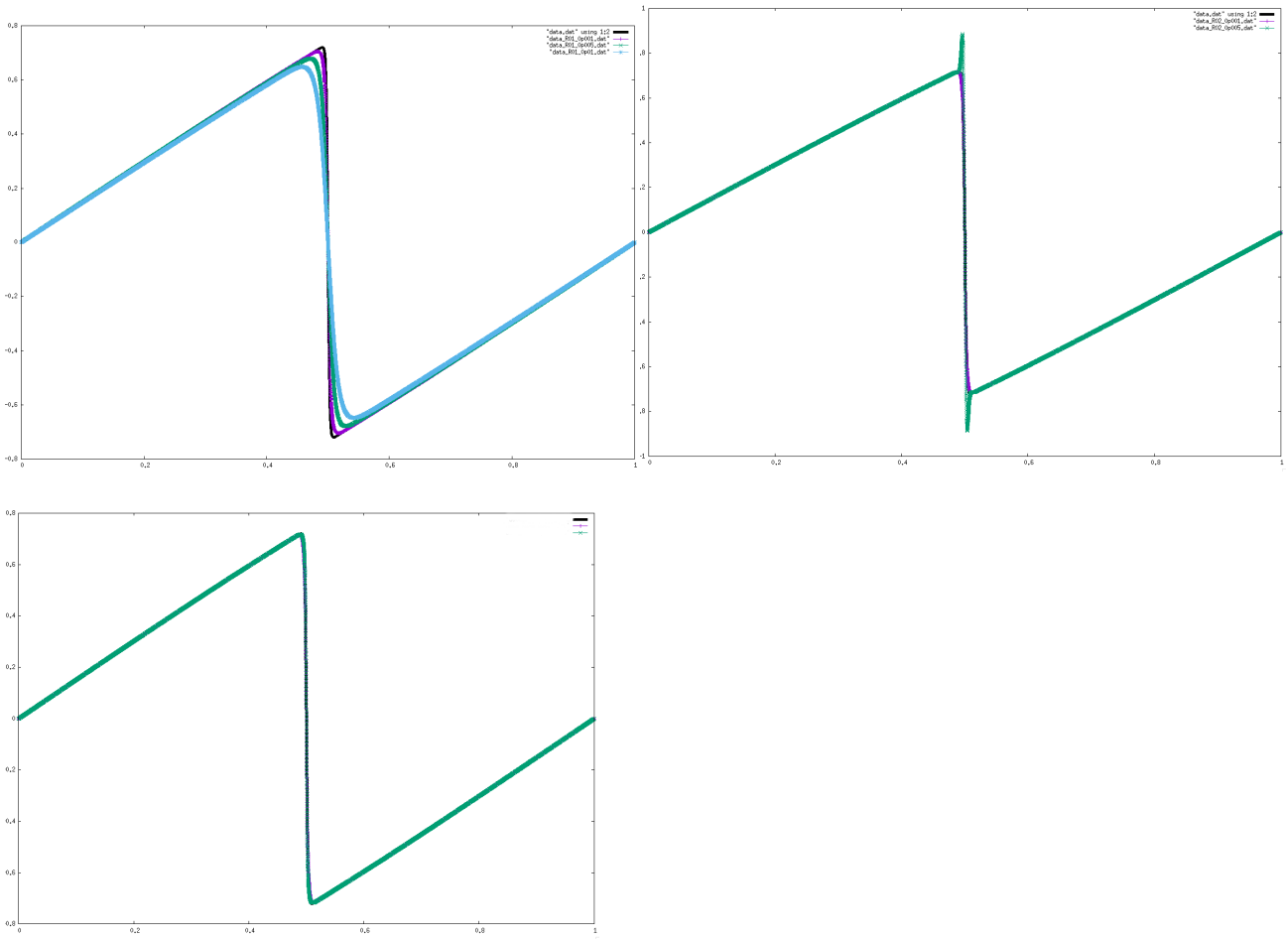

Figure 2: Burgers $1D$ Test 2 ; Top Left: numerical solution for first order scheme, Top Right: numerical solution for second order scheme. Bottom: numerical solution for second order AP scheme. $\nu = 10^{-3}$

Test 3:
We consider now the propagation of a perturbation. For this case the source term is set to zero. The initial condition is given by

$$\rho(0, x) = 1.0 + 0.1e^{-\frac{x^2}{\sigma}}$$

We compare the various solvers for this test case and we choose a viscosity value very close to zero, namely $\nu = 10^{-10}$. As for the previous test case the explicit time step is given by $\Delta t_{ex} = 10^{-5}$.

On (Fig. 4-5-6) we first note that the relaxation scheme is able to treat the propagation of the wave with large time step value (ranging from $200\Delta t_{ex}$ to $1000\Delta t_{ex}$). Second, we remark that both the standard method (Cranck-Nicolson scheme) and the relaxation scheme exhibit

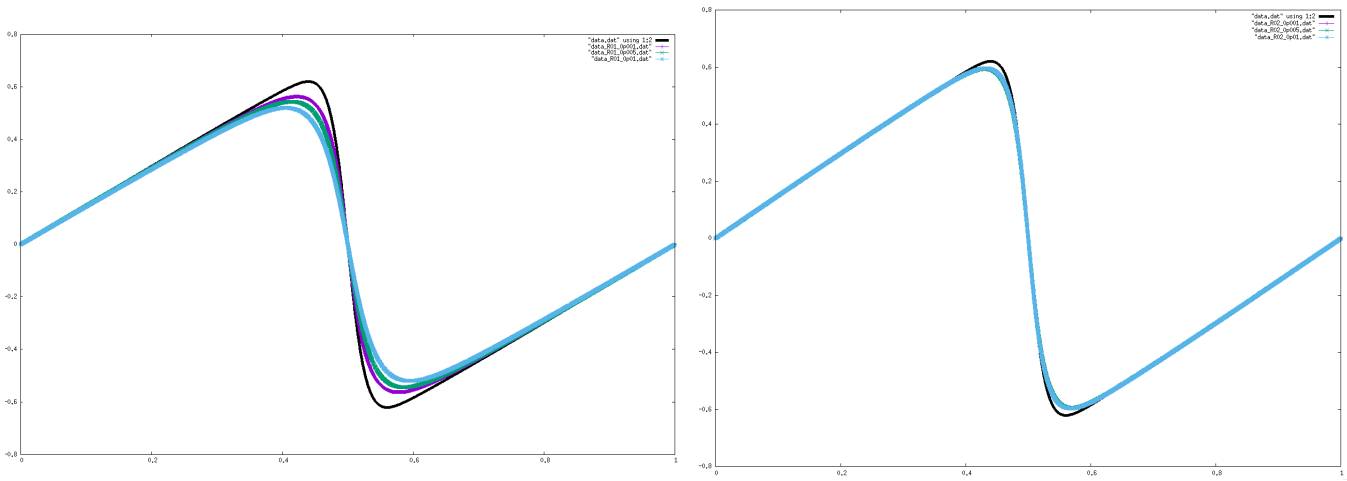

Figure 3: Burgers $1D$ Test 2; Left: numerical solution for first order scheme, Right: numerical solution for second order scheme. $\nu = 10^{-2}$

numerical dispersion as expected, the relaxation method being slightly more dispersive. Those results confirm the previous conclusions. This test and the previous one show that both the relaxation scheme and the standard implicit methods are able to solve the Burgers equation with a time step far larger than the explicit one. All in all, both methods allow to avoid the correlation between spatial discretization and time. Let us now compare the CPU time for a run to verify if the relaxation method is competitive from a computational point of view, despite the increase in the number of unknowns. The comparison is done between the standard linearized Cranck-Nicolson method and the relaxation method. In table (tab 5.1) we observe that for small

	CN method			Relaxation method		
Δt cells	5.10^3	10^4	2.10^4	5.10^3	10^4	2.10^4
$\Delta t = 0.005$	67	217.5	980	75.5	240	1100
$\Delta t = 0.01$	35	114	518	41	122.5	561
$\Delta t = 0.02$	18	61	280	20	63	294
$\Delta t = 0.05$	9.5	32.5	144	8	29	126

Table 2: CPU time (in seconds) for the $1D$ Burgers test-case (test 3)

.

time steps where the matrix associated with the different problem is close to the mass matrix, as expected, the standard method is more efficient since we solve a smaller problem. When the time step increase the conditioning of the Burgers matrix deteriorates and in this case the relaxation method becomes more efficient. This can be explained by the fact that we solve small and simple problems in the relaxation method for which the condition number does not depend on the time step or the physical coefficients contrary to the Cranck-Nicolson method for which the condition number increases with the time step. While the cost of a matrix assembly (at each

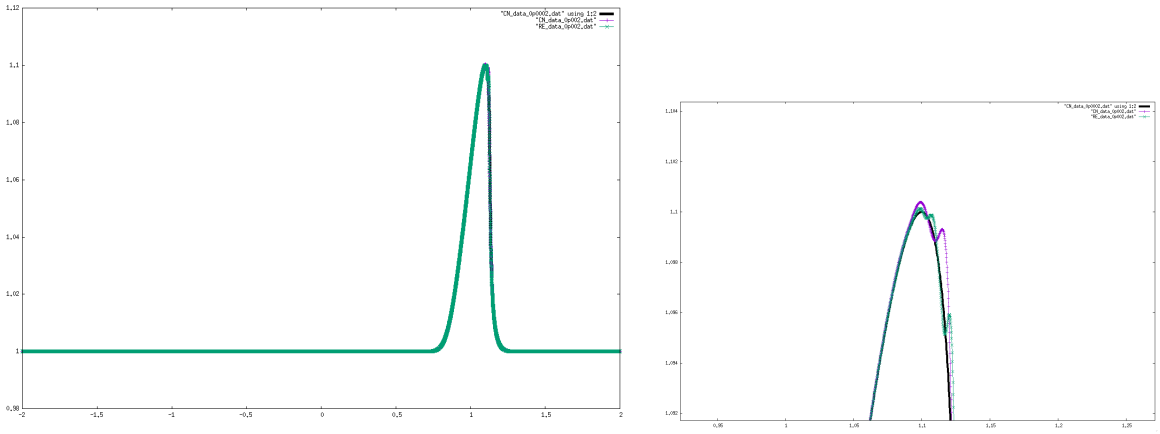

Figure 4: Burgers $1D$ Test 3 ; Left: Comparison between a fine solution (black), the Cranck-Nicolson solution (green) and the relaxation solution (purple) for $\Delta t = 0.002$, Right: Zoom

step for Cranck-Nicolson, at the beginning for the relaxation method) is negligible in the one dimensional case, it will have a non-negligible impact for $2D$ or $3D$ problems. For those cases, we expect the relaxation method to be more efficient compared to the classical Cranck-Nicolson method. Moreover the one-dimensional Burgers equation is not ill-conditioned (no physical eigenvalues close to zero, no problem linked the frequencies orthogonal to the transport, etc.) and thus for more complex systems (for instance the 2D Burgers equation) the relaxation method should also be more efficient compared to the standard method.

5.2 1D compressible Navier-Stokes equation

The second model proposed to validate the implicit relaxation method is the compressible Navier-Stokes equation given by

$$\begin{cases} \partial_t \rho + \partial_x(\rho v) = S_\rho \\ \partial_t \rho v + \partial_x(\rho v^2 + p) = \partial_x(\mu(\rho)\partial_x v) - \rho g + S_{\rho v} \\ \partial_t \rho E + \partial_x(\rho v E + pv) = \partial_x(\mu(\rho)\partial_x \frac{v^2}{2}) + \partial_x(\eta \partial_x T) - \rho v g + S_E \end{cases}$$

with ρ the density, v the velocity, E the total energy and T the temperature. The coefficient μ and η are the viscous and thermal viscosity and g the (constant) gravity field magnitude. The relaxation system associated with this model reads

$$\begin{cases} \partial_t \boldsymbol{U} + \partial_x \boldsymbol{V} = \boldsymbol{G}(\boldsymbol{U}) \\ \partial_t \boldsymbol{V} + \alpha^2 \partial_x \boldsymbol{U} = \dfrac{R}{\epsilon}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) + \boldsymbol{H}(\boldsymbol{U}) \end{cases}$$

with $\boldsymbol{H}(\boldsymbol{U}) = A(\boldsymbol{U})\boldsymbol{G}(\boldsymbol{U})$, $A(\boldsymbol{U})$ the Jacobian of $\boldsymbol{F}(\boldsymbol{U})$,

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho v \\ \rho E \end{pmatrix}, \quad \boldsymbol{F}(\boldsymbol{U}) = \begin{pmatrix} \rho v \\ \rho v^2 + p \\ \rho v E + pv \end{pmatrix}, \quad \boldsymbol{G}(\boldsymbol{U}) = \begin{pmatrix} S_\rho \\ -g\rho + S_{\rho v} \\ -g\rho v + S_E \end{pmatrix}$$

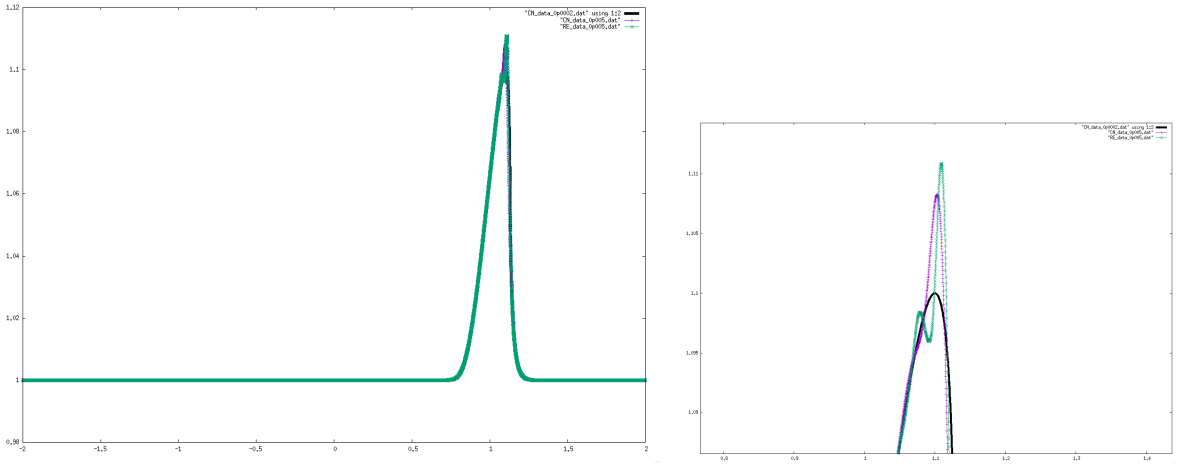

Figure 5: Test 3 ; Left: Comparison between a fine solution (black), the Cranck-Nicolson solution (green) and the relaxation solution (purple) for $\Delta t = 0.005$, Right: Zoom

and

$$R = \left(\alpha^2 - A(\boldsymbol{U})^2\right) \begin{pmatrix} \delta & 0 & 0 \\ -\frac{\mu(\rho)u}{\eta\rho} & \frac{\mu(\rho)}{\eta\rho} & 0 \\ -\frac{\eta T + \mu(\rho)u^2}{\eta\rho} & \frac{(\mu(\rho) - (\gamma-1)\eta)u}{\eta\rho} & \frac{(\gamma-1)\eta}{\eta\rho} \end{pmatrix}^{-1}$$

with $\epsilon = \eta$, and $\delta > 0$ a small parameter which allows to obtain a invertible matrix adding a very small diffusion on the density. As for the Burgers equation, we will first validate the convergence on a steady solution and then compare the various solvers.

Test 1:
We validate the order of convergence with a steady solution, assuming periodic boundary conditions. We consider the initial solution

$$\begin{cases} \rho(t=0,x) = 1.0 + 0.1e^{-\frac{x^2}{\sigma}} \\ u(t=0,x) = 2.0 \\ T(t=0,x) = \frac{1.0}{\rho(t=0,x)} \end{cases}$$

with $\sigma = 0.01$. We also choose $\mu = \eta = 0$, $\gamma = 1.4$ and $g = 0$. In table (tab 3) we compare the

Scheme Δt	$\Delta t = 1.0E^{-2}$	$\Delta t = 5.0E^{-3}$	$\Delta t = 2.5E^{-3}$	$\Delta t = 1.25E^{-3}$
CN scheme	$8.8E^{-3}$	$2.25E^{-3}$	$5.7E^{-3}$	$1.4E^{-3}$
Relaxation scheme	$2.25E^{-3}$	$5.7E^{-4}$	$1.4E^{-4}$	$3.6E^{-5}$

Table 3: Convergence results for the compressible Navier-Stokes equation (Test 1)

numerical error for the standard Cranck-Nicolson scheme with a Newton solver and the second

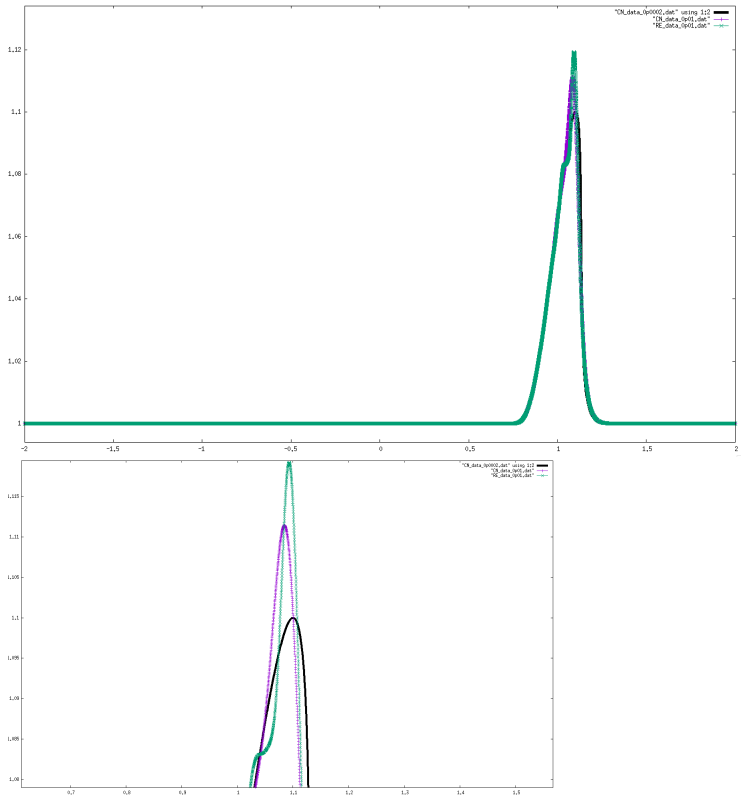

Figure 6: Test 3 ; Left: Comparison between a fine solution (black), the Cranck-Nicolson solution (green) and the relaxation solution (purple) for $\Delta t = 0.01$, Right: Zoom

order relaxation method. The results exhibit the expected orders of convergence: both schemes converge with the second order. We also notice that the relaxation method converges with a better constant. Now we perform another test where we will compare first the results of both methods qualitatively and then the CPU time.

Test 2:
For this case the source term is set to zero. We apply the various schemes on the propagation of an acoustic wave. To that end we set the initial data to

$$\begin{cases} \rho(t=0,x) = 1.0 + 0.1e^{-\frac{x^2}{\sigma}} \\ u(t=0,x) = M \\ T(t=0,x) = \frac{1}{3} \end{cases}$$

The value of α is allowed to change following the Mach number associated to the case. We first consider a low Mach regime with $\alpha = 0$. In table (tab 4) we observe that (i) contrarily to the Burgers case the relaxation method is more efficient for all scanned values of the time step and (ii) similarly as for the Burgers case, the performance difference between the two methods increases with the time step value. Those results can be explained by the fact that the steady

	CN method			Relaxation method		
Δt / cells	5.10^3	10^4	2.10^4	5.10^3	10^4	2.10^4
$\Delta t = 0.005$	160	540	2350	135	430	1920
$\Delta t = 0.01$	90	315	1550	70	220	1000
$\Delta t = 0.02$	55	175	765	40	125	530
$\Delta t = 0.05$	30	100	420	20	65	270

Table 4: CPU time for the test case 2 with $\alpha = 0$

Euler equation are ill-conditioned and consequently the Gmres solvers converge slowly. As for the Burgers case, the implementation used for the test does not take advantage of the inherent parallelism of the relaxation and transport problems and the relative cost of the various matrix assemblies is very small. We can expect better results in 2D or 3D with the model parallelization. We now compare the standard method and the relaxation method for various time step and α values. We first consider a low Mach regime with $M = 0$.

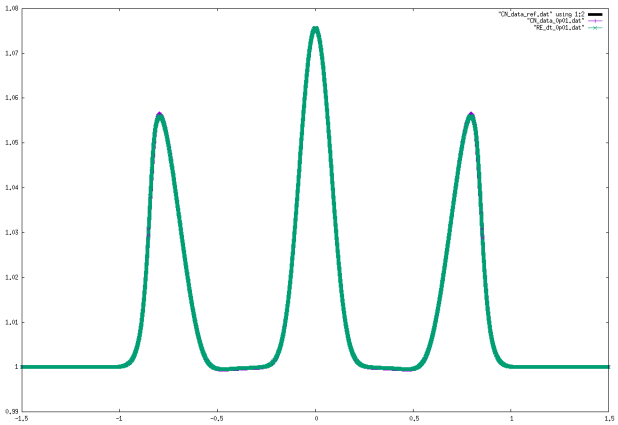

Figure 7: Comparison between a fine solution (black), the Cranck-Nicolson solution (green) and the relaxation solution (purple) for $\Delta t = 0.01$

In (Figs (7)-(8)-(9)) we observe that both methods capture the fine solution (in black) with high accuracy for a time step approximately one thousand larger than the explicit time step. For larger time steps values (between $5000\Delta t_{ex}$ and $10000\Delta t_{ex}$) we notice as previously that the relaxation remain accurate but is slightly more dispersive than the standard implicit method. Those observations are in accordance with the previous ones.
We now consider the case where $M = 0.5$. As previously we first compare the two methods on a qualitative basis and then compare the CPU time for a full run. As for the low Mach case, we observe that the CPU time is smaller for the Relaxation method compared to the Cranck-Nicolson method. The difference between the two schemes is less important than for the previous case. This difference comes from that for this test case the Jacobian of the Euler equations is less

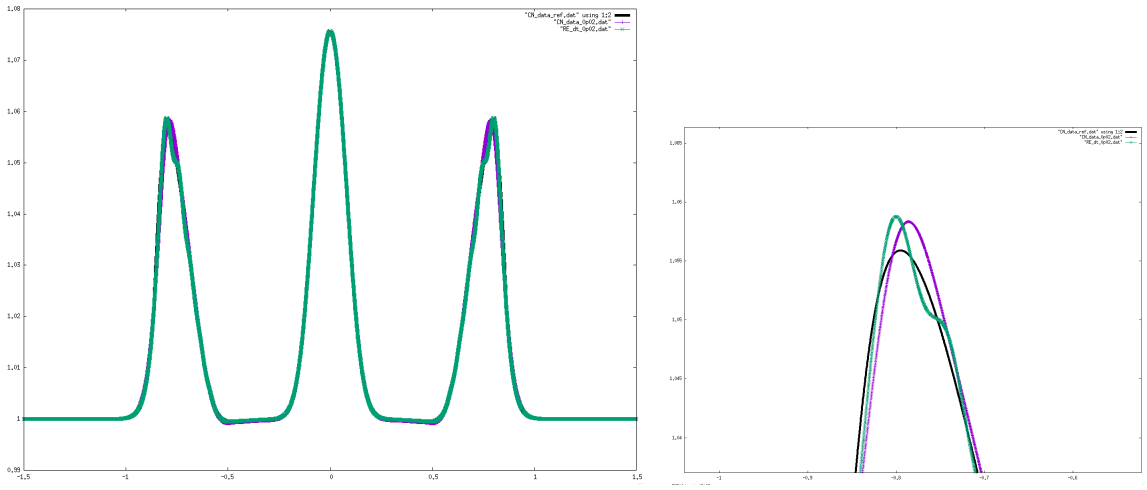

Figure 8: Comparison between a fine solution (black), the Cranck-Nicolson solution (green) and the relaxation solution (purple) for $\Delta t = 0.02$, Right: Zoom

ill-conditioned that for the low Mach case. In this test we consider less bigger time step since the maximal velocity is more important but the time step $\Delta t = 0.01$ and $\Delta t = 0.02$ corresponds to around one and two hundred times the explicit time step. Comparing the figures (fig 10-11) we observe as before that the Relaxation method is a little more dispersive than the Cranck-Nicolson method. Finally, at the end this test case shows that the relaxation method is slightly more

	CN method			Relaxation method		
Δt / cells	5.10^3	10^4	2.10^4	5.10^3	10^4	2.10^4
$\Delta t = 0.01$	145	480	2150	100	320	1470
$\Delta t = 0.02$	80	290	1200	60	200	970

Table 5: CPU time for the test case 2 with $\alpha = 0.5$

efficient than the standard implicit one even with a fully sequential implementation. Two main reasons explain this difference: first, matrix assemblies are performed only once at initialization for the relaxation mathod while they have to be performed at each time step in the standard implicit scheme. Second, though more problems are to be solved when using the relaxation method, their resolution is more efficient than the resolution of the global system in the standard method due to the fact that they are well-posed problems. We thus obtain a competitive method without even taking advantage of the additional parallelism brought forth by the method. The price to pay for this efficiency lies in the slightly higher dispersion of the method.

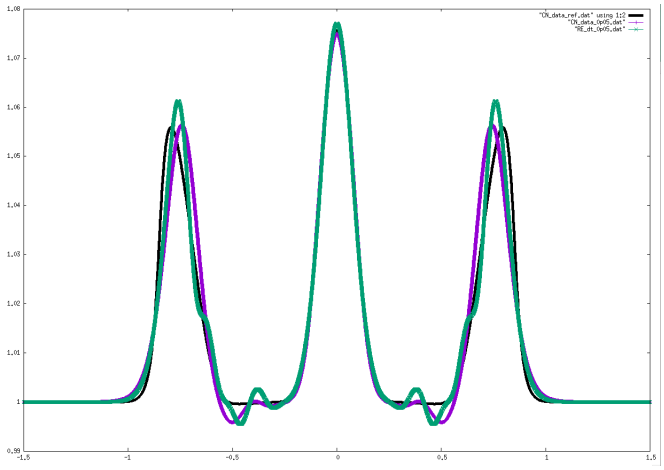

Figure 9: Comparison between a fine solution (black), the Cranck-Nicolson solution (green) and the relaxation solution (purple) for $\Delta t = 0.05$

5.3 Isothermal Euler 2D

We will now consider the case of the 2D isothermal Euler equation

$$\begin{cases} \partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = S_r \rho \\ \partial_t (\rho \boldsymbol{u}) + \nabla \cdot \left(\rho \boldsymbol{u} \otimes \boldsymbol{u} + c^2 \rho I_d \right) = \mu \Delta \boldsymbol{u} + (\mu + \lambda) \nabla (\nabla \cdot \boldsymbol{u}) + S_r \boldsymbol{u} \end{cases}$$

which corresponds to the previous Euler equation with a constant temperature. The fluxes are given by

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \end{pmatrix}, \quad \boldsymbol{F}_x(\boldsymbol{U}) = \begin{pmatrix} \rho u_1 \\ \rho u_1^2 + c\rho \\ \rho u_1 u_2 \end{pmatrix}, \quad \boldsymbol{F}_x(\boldsymbol{U}) = \begin{pmatrix} \rho u_2 \\ \rho u_1 u_2 \\ \rho u_2^2 + c^2 \rho \end{pmatrix}$$

and the diffusion matrices by

$$D_{xx}(\boldsymbol{U}) = \begin{pmatrix} 0 & 0 & 0 \\ -\frac{2\mu+\lambda}{\rho} & \frac{2\mu+\lambda}{\rho} & 0 \\ -\frac{\mu}{\rho} & 0 & \frac{\mu}{\rho} \end{pmatrix}, \qquad D_{xy}(\boldsymbol{U}) = \begin{pmatrix} 0 & 0 & 0 \\ -\frac{\mu+\lambda}{\rho} & 0 & \frac{\mu+\lambda}{\rho} \\ 0 & 0 & 0 \end{pmatrix} \tag{35}$$

and

$$D_{xy}(\boldsymbol{U}) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\frac{\mu+\lambda}{\rho} & \frac{\mu+\lambda}{\rho} & 0 \end{pmatrix}, \qquad D_{yy}(\boldsymbol{U}) = \begin{pmatrix} 0 & 0 & 0 \\ -\frac{\mu}{\rho} & \frac{\mu}{\rho} & 0 \\ -\frac{2\mu+\lambda}{\rho} & 0 & \frac{2\mu+\lambda}{\rho} \end{pmatrix} \tag{36}$$

We do not rewrite here the full time scheme as it is the same splitting scheme as before. Exactly as in 1D we will first check the convergence in time and then compare the standard and relaxation solvers.

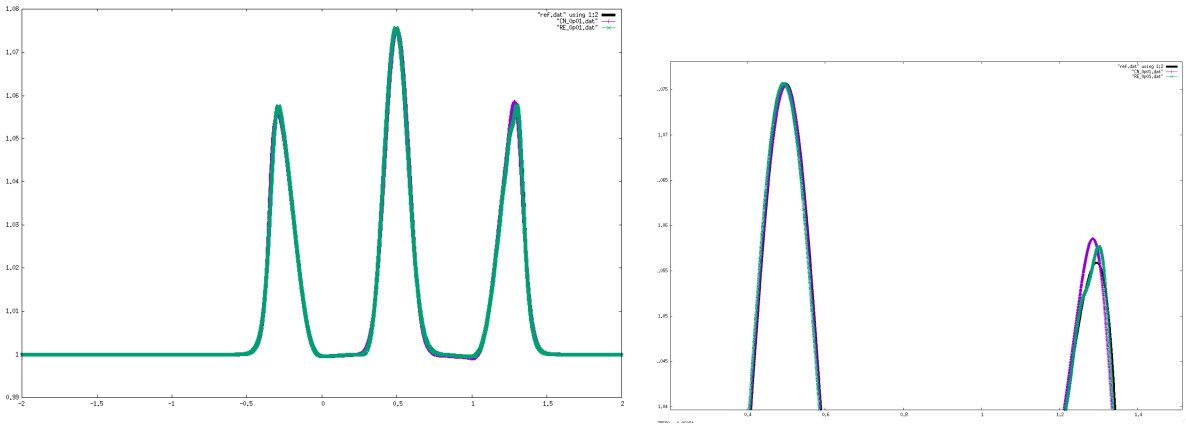

Figure 10: Comparison between a fine solution (black), the Cranck-Nicolson solution (green) and the relaxation solution (purple) for $\Delta t = 0.01$, Right: Zoom

Test 1: In order to validate the splitting scheme we use analytic test case with a source term (to obtain a steady state). We consider

$$\rho(t, \mathbf{x}) = 1.0 + 0.2e^{-\frac{\|\mathbf{x}\|^2}{\sigma^2}}, \quad \boldsymbol{u}(t, \mathbf{x}) = 0.2 + 0.5e^{-\frac{\|\mathbf{x}\|^2}{\sigma^2}} I_d$$

Let us first check the convergence of the second order splitting scheme. The final time is $t_f = 0.2$ and the grids is given $100 * 100$. The results are given in table (6). The results of table (6) show

	Error	Order
$\Delta t = 0.05$	$7.9E^{-2}$	-
$\Delta t = 0.025$	$1.6E^{-2}$	2.3
$\Delta t = 0.0125$	$3.8E^{-3}$	2.05
$\Delta t = 0.00625$	$9.3E^{-4}$	2.03
$\Delta t = 0.003125$	$2.3E^{-4}$	2.02

Table 6: Error and order for the test 1 one with the relaxation scheme.

that the scheme converge as expected with second order accuracy.
Test 2 For this test case we consider the propagation of an acoustic with initial data given by

$$\begin{cases} \rho(t = 0, \mathbf{x}) = 1.0 + 0.1e^{-\frac{\|\mathbf{x}\|^2}{\sigma}} \\ \boldsymbol{u}(t = 0, \mathbf{x}) = 0 \end{cases}$$

with $\sigma = 0.01$. The final time is $T_f = 0.4$. The CPU time need to solve the problem using the various schemes is given in table (7) In addition to the CPU results we compare qualitatively the solutions given by both methods. We consider a $400 * 400$ mesh, and use 3^{rd}-order B-Splines with maximal regularity. We compare various methods for the time step values $\Delta t = 0.01$, $\Delta t = 0.05$ and $\Delta t = 0.1$ On figures (12)-(13)-(14) we observe that the three methods yield

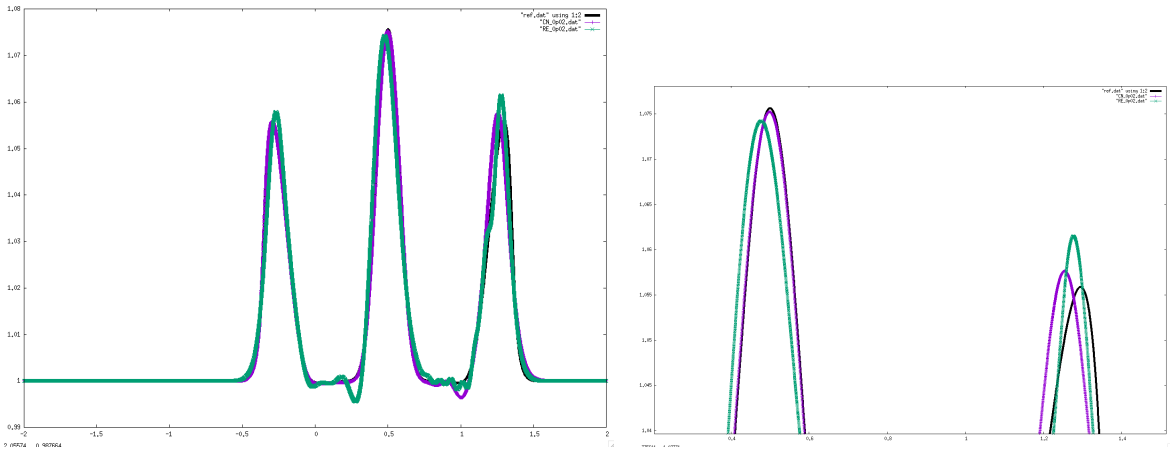

Figure 11: Comparison between a fine solution (black), the Cranck-Nicolson solution (green) and the relaxation solution (purple) for $\Delta t = 0.02$, Right: Zoom

	CN method			CN Newton			Relaxation method		
Δt / cells	100^2	200^2	400^2	100^2	200^2	400^2	100^2	200^2	400^2
$\Delta t = 0.01$	340	1320	5650	610	2410	9800	330	1260	5040
$\Delta t = 0.02$	170	670	3060	310	1250	6850	165	650	2555
$\Delta t = 0.05$	75	300	1290	140	555	3080	70	275	1115
$\Delta t = 0.1$	45	170	760	100	380	2190	40	155	625

Table 7: CPU time for the test case 2

qualitatively similar results. For larger time step value we notice that the standard Cranck-Nicolson with first order linearization does not preserve the symmetry contrary to the Newton scheme. We also notice that the relaxation seems less dispersive than the other methods in that case: the solution obtained with the relaxation method is closer to the numerical reference solution computed with a small time step value. This is confirmed by the results shown on figure (15) where we compare a 1D cut for the numerical reference solution and the solutions computed with the various methods with a large time step value.

6 Conclusion

In this work we proposed an implicit extension of the relaxation scheme proposed by [9]. This method is based on a time splitting scheme coupled with Cranck-Nicolson discretization and allows to obtain a second order time scheme with two steps: a linear transport step and a local relaxation step. Using a Schur decomposition the transport step can be rewritten as a Laplacian inversion and two L^2 projections. Both steps require solving a set of independent linear problems associated with well-known matrices (the mass and stiffness ones). Due to both the inherent parallelism and the fact that robust and efficient solvers exist for the stiffness and mass matrices,

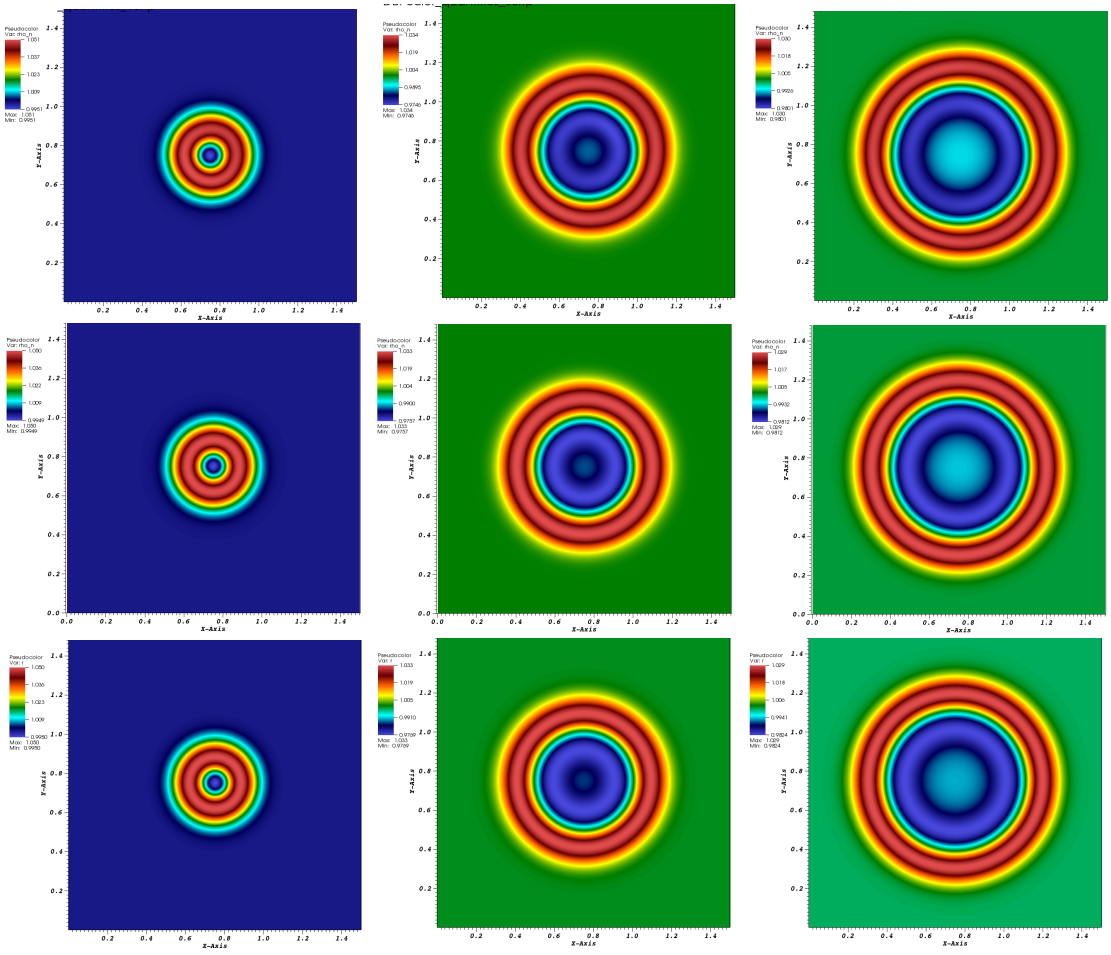

Figure 12: Comparison between the Cranck-Nicolson method (top), the CN coupled with Newton method (middle) and the relaxation (bottom) for a time step $\Delta t = 0.01$.

we obtain a method that is more robust and scalable than the standard implicit one for which a full nonlinear hyperbolic problem has to be solved implicitly. Numerical experiments have shown that the price to pay for the efficiency of the relaxation method lies in increased dispersive effected when compared with the standard implicit scheme. In future works we plan to follow three directions. First, since one of our problems of interest is to build an efficient solver for compressible MHD in magnetic fusion devices, the method must be extended to the treatment hyperbolic problems with divergence free constraints and for high diffusion regimes. Second, we will design and experiment procedures to avoid oscillations arising for discontinuous and strong gradient solutions using limiting or filtering techniques. Third, we will implement a parallel version of the method in order to take full advantage of its inherent parallel structure and evaluate its efficiency and scalability for large problem sizes.

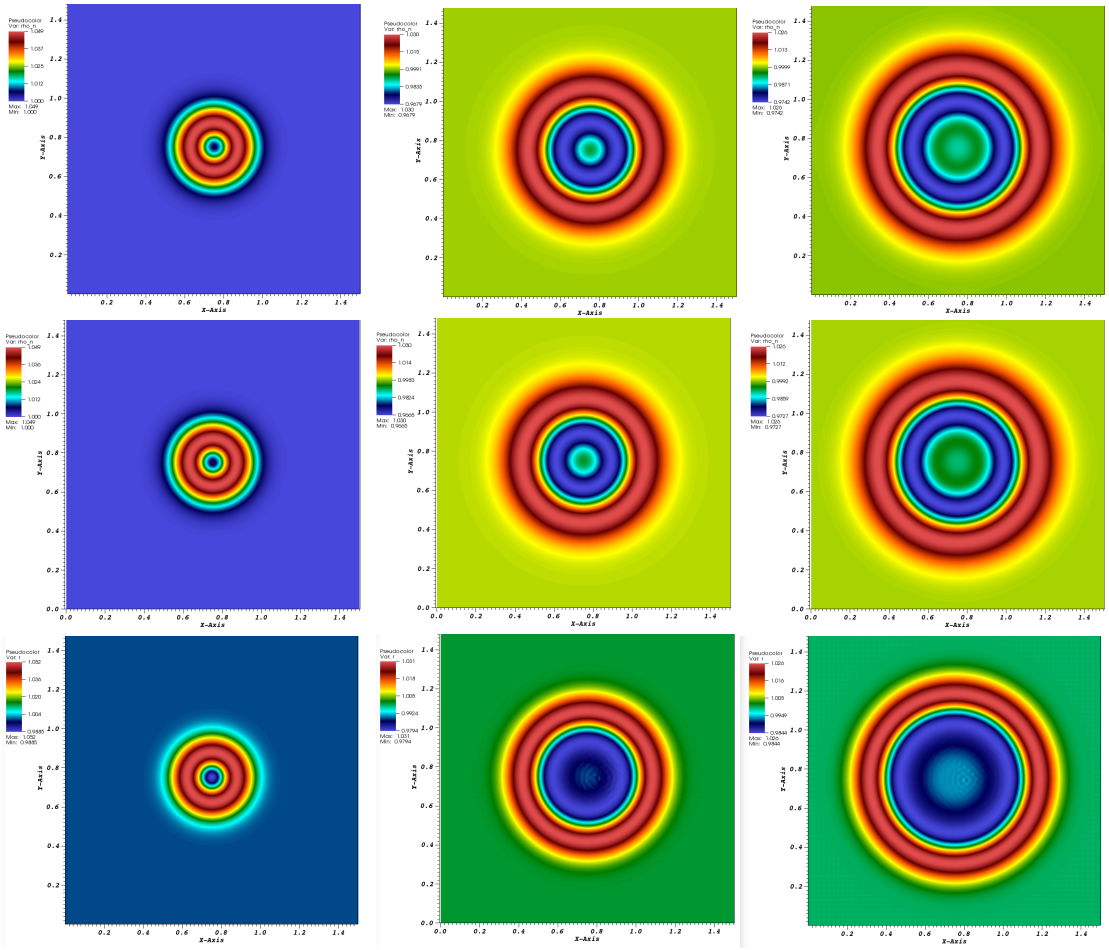

Figure 13: Comparison between the Cranck-Nicolson method (top), the CN coupled with Newton method (middle) and the relaxation (bottom) for a time step $\Delta t = 0.05$.

References

[1] M.K. Banda, M. Sead, A. Klar, and L. Pareschi. Compressible and incompressible limits for hyperbolic systems with relaxation. *Journal of Computational and Applied Mathematics*, 168(12):41 – 52, 2004. Selected Papers from the Second International Conference on Advanced Computational Methods in Engineering (ACOMEN 2002).

[2] E. Bertolazzi and G. Manzini. *High-order IMEX-RK finite volume methods for multidimensional hyperbolic systems*, pages 35–44. Springer Milan, Milano, 2003.

[3] L. Chacón. An optimal, parallel, fully implicit newton krylov solver for three-dimensional viscoresistive magnetohydrodynamics). *Physics of Plasmas*, 15(5), 2008.

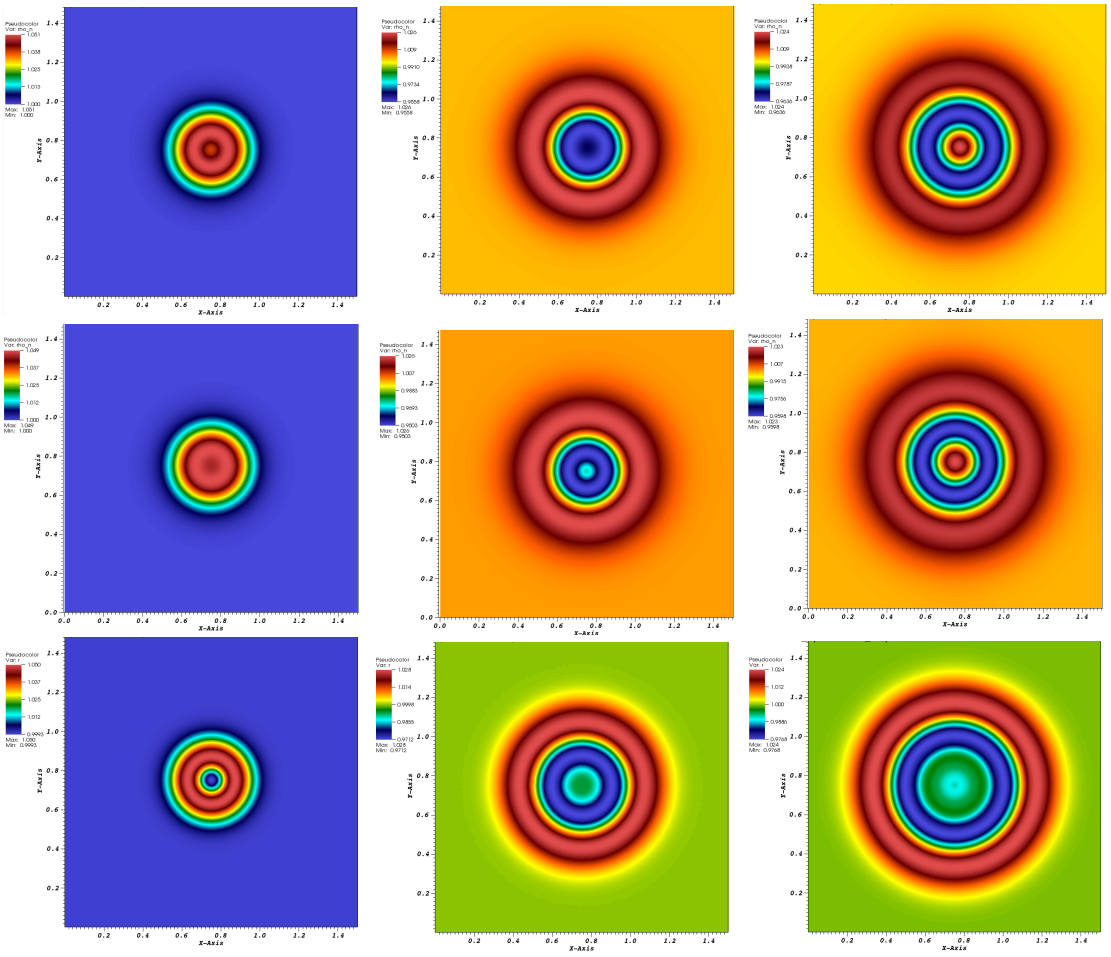

Figure 14: Comparison between the Cranck-Nicolson method (top), the CN coupled with Newton method (middle) and the relaxation (bottom) for a time step $\Delta t = 0.1$.

[4] L. Chacón, D.A. Knoll, and J.M. Finn. An implicit, nonlinear reduced resistive mhd solver. *Journal of Computational Physics*, 178(1):15 – 36, 2002.

[5] Courtès, Clémentine, Franck, Emmanuel, Helluy, Philippe, and Oberlin, Herbert. Study of physics-based preconditioning with high-order galerkin discretization for hyperbolic wave problems. *ESAIM: ProcS*, 55:61–82, 2016.

[6] P. Helluy L. Navoret D. Coulette, E. Franck and M. Mehrenberger. Palindromic discontinuous galerkin method for kinetic equations with stiff relaxation. 2017.

[7] B. Graille. Approximation of mono-dimensional hyperbolic systems: A lattice boltzmann scheme as a relaxation method. *Journal of Computational Physics*, 266:74 – 88, 2014.

[8] S. C. Jardin. Review of implicit methods for the magnetohydrodynamic description of magnetically confined plasmas. *J. Comput. Phys.*, 231(3):822–838, February 2012.

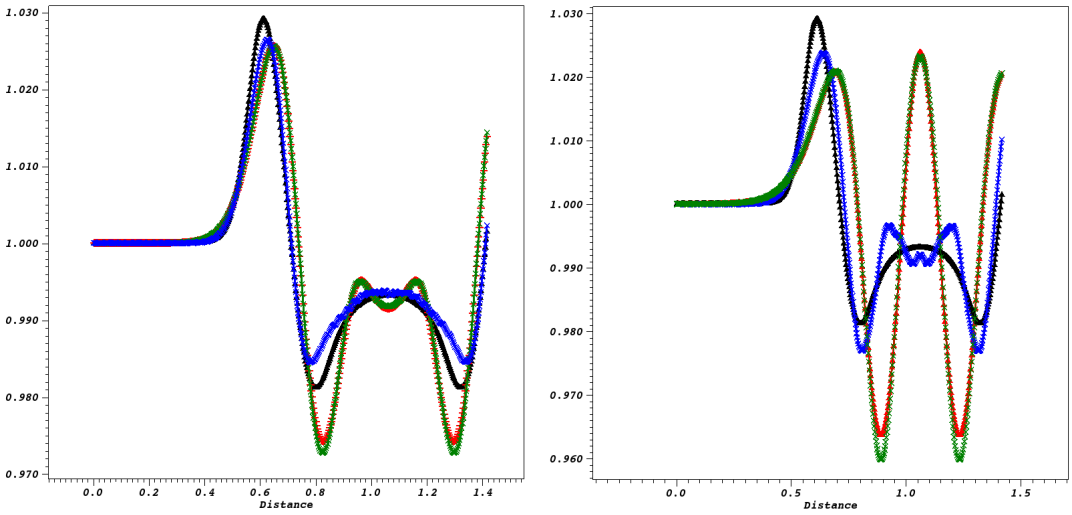

Figure 15: 1D cut of the solutions. In black the solution given by Newton with $Dt = 0.01$. In red, green and blue the solutions given by the CN method, the Newton and relaxation for $\Delta t = 0.05$ (left) and for $\Delta t = 0.1$ (right)

[9] S. Jin, Z. Xin, Shi Jin, and Zhouping Xin. The relaxation schemes for systems of conservation laws in arbitrary space dimensions. *Comm. Pure Appl. Math*, 48:235–277, 1995.

[10] Shi Jin. Runge-kutta methods for hyperbolic conservation laws with stiff relaxation terms. *Journal of Computational Physics*, 122(1):51 – 67, 1995.

[11] D. A. Knoll, V. A. Mousseau, L. Chacón, and J. Reisner. Jacobian–free newton–krylov methods for the accurate time integration of stiff wave systems. *Journal of Scientific Computing*, 25(1):213–230, 2005.

[12] Daniel R. Reynolds, Ravi Samtaney, and Carol S. Woodward. Operator-based preconditioning of stiff hyperbolic systems. *SIAM J. Scientific Computing*, 32(1):150–170, 2010.