MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2019/04.02.23.51-TDI

NUMERICAL METHODS APPLIED TO SPACE MAGNETOHYDRODYNAMICS FOR HIGH PERFORMANCE COMPUTING

Müller Moreira Souza Lopes

Doctorate Thesis of the Graduate Course in Applied Computing, guided by Drs. Margarete Oliveira Domingues, and Odim Mendes Junior, approved in May 02, 2019.

URL of the original document:
<http://urlib.net/8JMKD3MGP3W34R/3T3K8C2>

INPE
São José dos Campos
2019

PUBLISHED BY:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GBDIR)
Serviço de Informação e Documentação (SESID)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2019/04.02.23.51-TDI

NUMERICAL METHODS APPLIED TO SPACE MAGNETOHYDRODYNAMICS FOR HIGH PERFORMANCE COMPUTING

Müller Moreira Souza Lopes

Doctorate Thesis of the Graduate Course in Applied Computing, guided by Drs. Margarete Oliveira Domingues, and Odim Mendes Junior, approved in May 02, 2019.

URL of the original document:
<http://urlib.net/8JMKD3MGP3W34R/3T3K8C2>

INPE
São José dos Campos
2019

Cataloging in Publication Data

Lopes, Müller Moreira Souza.

L881n Numerical methods applied to space magnetohydrodynamics for high performance computing / Müller Moreira Souza Lopes. – São José dos Campos : INPE, 2019.

xxiv + 203 p. ; (sid.inpe.br/mtc-m21c/2019/04.02.23.51-TDI)

Thesis (Doctorate in Applied Computing) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2019.

Guiding : Drs. Margarete Oliveira Domingues, and Odim Mendes Junior.

1. Adaptive mesh refinement. 2. Magnetohydrodynamics. 3. High performance computing. 4. Divergence cleaning. 5. Magnetosphere. I.Title.

CDU 519.6:537.84

Aluno (a): *Müller Moreira Souza Lopes*

Título: "NUMERICAL METHODS APPLIED TO SPACE MAGNETOHYDRODYNAMICS FOR HIGH PERFORMANCE COMPUTING"

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de *Doutor(a)* em *Computação Aplicada*

Dr. Elbert Einstein Nehrer Macau

Presidente / INPE / São José dos Campos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Dra. Margarete Oliveira Domingues

Orientador(a) / INPE / SJCampos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Dr. Odim Mendes Junior

Orientador(a) / INPE / SJCampos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Dr. Stephan Stephany

Membro da Banca / INPE / SJCampos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 02 de maio de 2019

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de *Doutor(a)* em

Computação Aplicada

Dr. Oswaldo Duarte Miranda

Membro da Banca / INPE / SJCampos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Dr. Cayo Prado Fernandes Francisco

Convidado(a) / IAE / São José dos Campos - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Dr. Iberê Luiz Caldas

Convidado(a) / USP / São Paulo - SP

( ) Participação por Video - Conferência

(X) Aprovado *( ) Reprovado*

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 02 de maio de 2019

To my mother.

ACKNOWLEDGEMENTS

I thank the National Institute for Space Research (INPE) and the Graduation program in Applied computing (CAP) for the necessary structure for the realization of this work.

This thesis was supported by CNPq (PhD scholarship, process number $140626/2014-0$).

I also thank to CAPES for the sandwich doctorade grant via PDSE program (process number $88881.132489/2016-01$).

I thank the Aix-Marseille University for the hospitality during the CAPES sandwich doctorade visit in April 2017.

I thank the University of Southampton for the hospitality during the CAPES sandwich doctorade visit from May to July 2017 and the CNPq/FAPESP Sprint/SOU visit from January to March 2018.

I thank the supervisors Margarete Domingues and Odim Mendes for suggesting and planning this project and for the many fruitful scientific discussions, suggestion, opportunity, material and patience which allowed this work to be developed.

I thank Dr. Ralf Deiterding for receiving me in the University of Southampton in my two visits, for the fruitfull discussions which colaborates for the progress of this work and for his assitance with the AMROC framework.

I thank Dr. Kai Schneider for receiving me in the Aix-Marseille Univesity, for the fruitfull discussions regarding to local time stepping methods, which colaborates for the progress of this work.

I thank also Varlei Menconi and Marcelo Banik for the technical support and Anna Gomes for the fruitful discussions.

This work has the finantial/technical support related to the cluster Orion (FINEP Grant: 0112052700).

ABSTRACT

The study of physical systems related to the space sciences presents several challenges regarding the great variety of phenomena and scales. In particular, magnetohydrodynamic models are applied in space weather to study phenomena that reach the Earth's atmosphere, affecting infrastructure services provided to society. In this context, many fields of study complement each other to obtain a better understanding of these systems. Among these approaches are the numerical simulations, which provide an approximated prediction of the system behaviour from a predefined setup. However, in order for this simulations to be viable, they must be performed in a realistic time, which is a challenge for complex models such as the magnetohydrodynamic equations. Thus, the use of adaptive computational meshs that present higher refinements in regions of interest is an effective strategy to reduce the computational time required for simulations. In particular, the simulations of the magnetohydrodynamic equations present a fundamental challenge that is the emergence of a non-realistic divergence over the magnetic field caused by numerical errors, which requires special techniques to be treated in order to maintain the correctness and the numerical stability. This work presents a code for solving magnetohydrodynamic equations using a high-performance environment that allows the use of adaptive meshs and parallel algorithms. Also, a new technique is proposed to overcome the divergence problem. The code is applied in several test problems in order to verify its performance. Then it is applied to a Earth magnetosphere model. As a product of this thesis, an innovative, high-performance tool for the future space weather research conducted at INPE.

Keywords: Adaptive mesh refinement. Magnetohydrodynamics. High performance computing. Divergence cleaning. Magnetosphere.

MÉTODOS NUMÉRICOS APLICADOS A MAGNETOHIDRODINÂMICA ESPACIAL PARA COMPUTAÇÃO DE ALTO DESEMPENHO

RESUMO

O estudo de sistemas físicos relacionados às ciências espaciais apresentam diversos desafios devido à grande variedade de fenômenos e escalas envolvidos. Em particular, modelos magnetohidrodinâmicos são aplicados em clima espacial para estudar fenômenos que atingem a atmosfera terrestre, os quais podem afetar serviços de infraestrutura oferecidos à sociedade. Neste contexto, diversas áreas de estudo se complementam visando obter um melhor entendimento destes sistemas. Dentre estas abordagens encontram-se as simulações numéricas, que fornecem uma previsão aproximada do comportamento do sistema a partir de uma configuração predeterminada. Porém, para que as simulações sejam viáveis, elas devem ser realizadas dentro de um intervalo de tempo realístico, o que é um desafio para modelos complexos como as equações magnetohidrodinâmicas. Desta forma, o uso de malhas computacionais adaptativas que apresentam maior refinamento em regiões de interesse é uma estratégia efetiva para reduzir o custo computacional requerido por estas simulações. Em particular, as simulações das equações magnetohidrodinâmicas apresentam um desafio inerente associado a emersão de uma divergência não realística sobre o campo magnético causada por erros numéricos, sendo necessário o uso de técnicas especiais para manter a exatidão e a estabilidade numérica. Este trabalho apresenta um código para simular as equações magnetohidrodinâmicas utilizando um ambiente de alto desempenho que permite o uso de malhas adaptativas e algoritmos paralelizados. Além disso, é proposta uma nova técnica para lidar com o problema da divergência. Este código é aplicado em diversos problemas de teste para verificar sua performance, incluindo um modelo de magnetosfera terrestre. Como produto desta tese, obtém-se uma ferramenta inovadora e de alta performance para futuras pesquisas em clima espacial a serem conduzidas pelo INPE.

Palavras-chave: Refinamento de malha adaptativo. Magnetohidrodinâmica. Processamento de alto desempenho. Correção de divergência. Magnetosfera.

LIST OF FIGURES

Page

4.1 Numerical extrapolation to the interfaces. 25
4.2 Full Riemann fan for the MHD equations. 28
4.3 Example of a slope s dividing two arbitrary states. 32
4.4 HLL Flux Riemann's fan 34
4.5 Schematic structure of the HLLD solver Riemann's fan. 35

5.1 Red/Black Gauss-Seidel colour distribution in a 2D mesh. 56
5.2 V-cycle multigrid algorithm. 59

6.1 Example of hierarchy of retangular submeshes. The clustered regions marked in grey are overlayed by a finer submesh. 62
6.2 Adaptive mesh and corresponding tree structure to store the patches. . . 64
6.3 Example of a patch containing two layers of ghost cells. 64
6.4 Patch with ghost cells placed over the mesh hierarchy. 65
6.5 Cell clustering algorithm. 67

7.1 AMROC folder hierarchy. 70

8.1 Riemann type IC: 1 direction. Cuts over the x-axis for the 2D and 3D solutions using meshes with 2048^2 and 512^3 cells, alongside with the exact solution. 79
8.2 Riemann type IC: 2 directions. Results for ρ, p, $\mathcal{E}$ and u_x obtained by the 2D and 3D simulations from the AMROC and CARMEN codes. . . . 81
8.3 Riemann type IC: 2 directions. Results for u_y, u_z, B_x, B_y and B_z obtained by the 2D and 3D simulations from the AMROC and CARMEN codes. . 82
8.4 2D-OTV: solution for ρ using different flux limiters in a base mesh with 256^2 cells. 84
8.5 2D-OTV: solution for ρ using different flux limiters in a base mesh with 2048^2 cells. 85
8.6 GLM parabolic-hyperbolic correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the 2D problems. 90
8.7 GLM parabolic-hyperbolic correction: Solutions obtained for the 2D problems using the mesh $\mathcal{G}_4$. 93
8.8 GLM parabolic-hyperbolic correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the 3D-OTV problem. 94

8.9 GLM parabolic-hyperbolic correction: Solutions obtained for the 3D-OTV problem using the mesh $\mathcal{G}_2$. 95
8.10 GLM elliptic correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the 2D problems. 97
8.11 GLM elliptic correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the 3D-OTV problem. . . . 98
8.12 GLM elliptic correction: Solutions obtained for the 2D problems using the mesh $\mathcal{G}_4$. 99
8.13 GLM elliptic correction: Solutions obtained for the 3D-OTV problem using the mesh $\mathcal{G}_2$. 100
8.14 GLM triple correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the ROT, 2D-OTV and BWV problems. 102
8.15 GLM triple correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the 3D-OTV problem. . . . 103
8.16 GLM triple correction: Solutions obtained for the 2D problems using the mesh $\mathcal{G}_4$. 104
8.17 GLM triple correction: Solutions obtained for the 3D-OTV problem using the mesh $\mathcal{G}_2$. 105
8.18 Parameter $\mathcal{D}_{\mathbf{B}}$, obtained for the 2D problems using the discussed divergence cleaning approaches in the base mesh $\mathcal{G}_4$. 108
8.19 Parameter $\mathcal{D}_{\mathbf{B}}$, obtained for the 3D-OTV problem using the discussed divergence cleaning approaches in the base mesh $\mathcal{G}_2$. 109
8.20 $\|\mathbf{B}\|$ obtained for the base mesh $\mathcal{G}_4$ the ADV, ROT, 2D-OTV and BWV problems using the discussed divergence cleaning approaches. 109
8.21 REC: Solution for p, the adaptive mesh and mesh distribution per processors. This simulation was performed using 4 refinement levels with threshold value $\epsilon = 0.001$ and 24 processors. 113
8.22 Results for p, the adaptive mesh and mesh distribution per processors. This simulation was performed using 4 refinement levels with threshold value $\epsilon = 0.001$ and 24 processors. 116
8.23 SCI: Results for p, the adaptive mesh and mesh distribution per processors. This simulation was performed using 4 refinement levels with threshold value $\epsilon = 0.001$ and 24 processors. 119
8.24 Magnetosphere setup after the hydrodynamic step. 124
8.25 Magnetosphere setup before the inclusion of the real solar wind data. . . 125

LIST OF TABLES

Page

4.1 Best convergence order obtainable with a s stages RK method. 39

8.1 Riemann type IC: 2 directions. Initial values for the each quadrant. . . . 80
8.2 GLM parabolic-hyperbolic correction: Elapsed time, in hours, for the parallel computations for the 2D problems, using the mesh $\mathcal{G}_4$, in function of the number of processors. 91
8.3 GLM parabolic-hyperbolic correction: Elapsed time, in hours, for the computations of the problems using 16 processors in function of the base mesh. 91
8.4 GLM parabolic-hyperbolic correction: Elapsed time, in hours, for the parallel computations for the 3D-OTV problem, using the mesh $\mathcal{G}_2$, in function of the number of processors. 91
8.5 GLM parabolic-hyperbolic correction: Breakdown of the CPU time in hours spent in the main tasks computations for different numbers of processors in the ADV problem. 92
8.6 GLM elliptic correction: Elapsed time, in hours, for the parallel computations for the 2D problems, using the mesh $\mathcal{G}_4$, in function of the number of processors. 96
8.7 GLM elliptic correction: Elapsed time, in hours, for the parallel computations for the 3D-OTV problem, using the mesh $\mathcal{G}_2$, in function of the number of processors. 97
8.8 GLM elliptic correction: Elapsed time, in hours, for the computations of the 2D and 3D problems using 16 processors in function of the base mesh used. 97
8.9 GLM triple correction: Elapsed time, in hours, for the parallel computations for the 2D problems, using the mesh $\mathcal{G}_4$, in function of the number of processors. 101
8.10 GLM triple correction: Elapsed time, in hours, for the parallel computations for the 3D-OTV problem using the mesh $\mathcal{G}_2$, in function of the number of processors. 102
8.11 GLM triple correction: Elapsed time, in hours, for the computations of the 2D and 3D problems using 16 processors in function of the base mesh used. 102

8.12 Divergence and CPU time, in hours, obtained by the discussed divergence cleaning methods. 107
8.13 ADV problem: Errors for the B_x and B_y components in the $\mathbb{L}_1$ and $\mathbb{L}_\infty$ norms obtained by the studied divergence cleaning approaches in the mesh $\mathcal{G}_4$ using 16 processors. 107
8.14 REC: Errors in p, memory consumption and CPU time obtained by using several refinement levels. 112
8.15 REC: Breakdown of the CPU time in % spent in the main tasks computations for different numbers of refinement levels using $\epsilon = 0.001$. 112
8.16 KHI: Errors in p, memory consumption and CPU time obtained by using several refinement levels with $\epsilon = 0.001$. 115
8.17 SCI: Errors in p, memory consumption and CPU time obtained by using several refinement levels. 118

LIST OF ABBREVIATIONS

ADV	–	Magnetic field loop advection
AMROC	–	Adaptive Mesh Refinement in Object-oriented C++
BWV	–	Spherical Blast Wave
CFL	–	Courant-Friedrich-Lewy Condition
FD	–	Finite Differences method
FE	–	Finite Elements method
FV	–	Finite Volumes method
GLM	–	Generalised Lagrange Multiplier
HDF	–	Hierarchical Data Format
HLL	–	Harten–Lax–van Leer
HLLD	–	Harten–Lax–van Leer Discontinuities
KHI	–	Kelvin-Helmholtz instability
MHD	–	Magnetohydrodynamic model
MPI	–	Message Passing Interface
MR	–	Multiresolution method
MUSCL	–	Monotone Upstream-Centered Schemes for Conservation Laws
OTV	–	Orszag-Tang vortex
RK	–	Runge–Kutta method
REC	–	Magnetic Reconnection
ROT	–	Rotor Problem
SAMR	–	Structured Adaptive Mesh Refinement
SCI	–	Shock-Cloud Iteration
SG	–	Scaled Gradient
VTK	–	Visualization ToolKit
WENO	–	Weighted Essentially Non-Oscillatory

LIST OF SYMBOLS

A	–	scalar value notation
$\mathbf{A}$	–	vector notation
$\mathbf{A}_1\mathbf{A}_2$	–	tensor product between the vectors $\mathbf{A}_1$ and $\mathbf{A}_2$
$\mathbb{A}$	–	matrix/tensor notation
$\mathbb{I}$	–	identity matrix
λ_D	–	Debye lenght
N_D	–	number of particles inside the Debye sphere
ϵ_0	–	vacuum premittivity constant
k	–	Boltzmann constant
e	–	electron charge
$\mathcal{L}$	–	plasma lenght scale
f_{pe}	–	plasma frequency
ν_{en}	–	electron-neutron collision rate
t	–	time
$\mathbf{r}$	–	physical space coordinates
$\mathbf{v}$	–	velocity space coordinates
α	–	generic type of particles
n_α	–	number of particles of type α per unit of volume
m_α	–	mass of particle of type α
q_α	–	electric charge of particles α
T_α	–	temperature of the particles α
f_α	–	distribution function of the particles α
$\mathbf{u}_\alpha$	–	fluid velocity of the particles α
$\mathbf{w}_\alpha$	–	thermal velocity of the particles α
$\mathbf{h}_\alpha$	–	heat flow of particles α
$\mathbb{P}_\alpha$	–	stress tensor of particles α
p_α	–	isotropic pressure component of the stress tensor of particles α
π_α	–	anisotropic component of the stress tensor of particles α
N_d	–	number of degrees of freedom
γ	–	adiabatic index
$\mathbf{A}_\alpha$	–	momentum rate of change of the particles α due to collisions
Q_α	–	generated heat per unit of volume of the particles α due to collisions
$\mathbf{B}$	–	magnetic field
$\mathbf{E}$	–	electric field
$\mathbf{J}$	–	current density
η	–	resistivity
ρ_C	–	single fluid charge density
ρ	–	single fluid mass density
$\mathbf{u}$	–	single fluid velocity
p	–	single fluid isotropic pressure

$\mathcal{E}$ – plasma internal energy
$\mathfrak{q}$ – vector of conservative variables
$\mathbf{w}$ – vector of primitive variables
$\lambda^{+/-}_{f/A/s/e}$ – MHD model eigenvalues
c_f – fast magnetosonic velocity
c_s – slow magnetosonic velocity
c_A – Alfvén velocity
R_m – magnetic Reynolds number
$\mathcal{U}$ – plasma velocity scale
t_e – final instant of the simulation
$\mathbb{F}$ – flux components
$\mathbf{S}$ – source terms
Ω – physical domain
d – number of dimensions of the physical domain
$\mathcal{C}_{i,j,k}$ – cell centred at the point (x_i, y_j, z_k)
Δx – length of the cell $\mathcal{C}$ in the x direction
$\Delta^3 r$ – volume of the cell $\mathcal{C}$
$\mathbf{q}_{i,j,k}$ – average value of $\mathfrak{q}$ in the cell $\mathcal{C}_{i,j,k}$
n – time evolution counter
$\mathbf{a}^n$ – solution of the variable $\mathbf{a}$ after n iterations of the time evolution
(m) – iterative method counter
$\mathbf{a}^{(m)}$ – solution of the variable $\mathbf{a}$ after m iterations of a iterative method
$\mathcal{S}$ – surface surrounding an integration domain
$\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$ – numerical fluxes in the x, y and z directions, respectively
ϕ – slope limiter function
$q_{\theta,\,i}$ – variable number θ of the solution vector $\mathbf{q}_i$
$\mathfrak{v}$ – vector of characterist variables
v_θ – variable number θ of the characteristic variables vector
$\mathfrak{R}_\theta$ – eigenvector associated with the eigenvalue λ_θ
σ – Courant number
$\mathcal{D}_{\mathbf{B}}$ – ratio between the divergence and magnitude of the magnetic field per unit of volum
ψ – divergence cleaning parameter
$\boldsymbol{\psi}$ – vector containing every solution ψ of the mesh
$\mathfrak{r}$ – residual from iterative method
$\mathfrak{r}_*$ – projected residual from iterative method
$\mathbf{e}_*$ – error from iterative method
ν_{pre} ν_{post} – number of iteration per level in the multigrid method
L – highest refinement level of the mesh
ϵ – mesh refinement thresholding value
ν – ration between flagged cells and total cells in the clustering algorithm
$\mathcal{G}^\ell$ – computational mesh discretised in the refinement level ℓ
$\mathcal{G}^\ell_m$ – patch number m of the mesh $\mathcal{G}^\ell$

ξ	–	distance to the origin
t_{CPU}	–	CPU time
$\mathbf{g}$	–	gravity field
$\mathbf{B_d}$	–	dipole magnetic field of the Earth

CONTENTS

Page

1 INTRODUCTION 1

2 PLASMA PHYSICS 5

2.1 Plasma characteristics 5

2.2 Mathematical description of plasma phenomena 7

2.2.1 Kinetic model 7

2.2.2 Fluid model 9

3 MAGNETOHYDRODYNAMICS 13

3.1 Ideal MHD 14

3.1.1 System eigenvalues 15

3.2 Resistive MHD 17

4 NUMERICAL FORMULATION 21

4.1 Finite Volumes method 22

4.2 Numerical Fluxes 24

4.2.1 High-Resolution Schemes 25

4.3 Riemann problem 27

4.3.1 Rankine–Hugoniot relations 31

4.3.2 HLL Riemann solver 33

4.3.3 HLLD Riemann solver 35

4.4 Runge-Kutta methods 39

4.5 Courant-Friedrich-Lewy Condition 40

5 DIVERGENCE CLEANING 43

5.1 Generalised Lagrange Multipliers approach 44

5.1.1 Splitting methods 45

5.1.2 Parabolic correction 46

5.1.3 Hyperbolic correction 46

5.1.4 Parabolic-hyperbolic correction 49

5.1.5 Elliptic correction 50

5.2 Relaxation Methods 51

5.3 Multigrid methods 56

5.4 Combining the elliptic and parabolic-hyperbolic corrections 58

6 PATCH-STRUCTURED ADAPTIVE MESH REFINEMENT . 61
6.1 Mesh Hierarchy 63
6.1.1 Patch boundaries 63
6.2 Flagging algorithm 65
6.3 Time evolution 67

7 COMPUTATIONAL ASPECTS 69
7.1 Generic SAMR solver 71
7.2 Base MHD module 71
7.3 Specific MHD module 71
7.4 Multigrid solver module 72
7.5 Running test cases 72
7.6 Building new test cases 74

8 RESULTS: VERIFICATION 77
8.1 Accuracy experiments 77
8.2 Divergence cleaning performance experiments 86
8.2.1 Results: GLM parabolic-hyperbolic correction 89
8.2.2 Results: GLM elliptic correction 96
8.2.3 Results: GLM triple correction 101
8.2.4 Comparison between the GLM methods 106
8.3 Adaptive mesh experiments 110
8.4 Magnethosphere simulation 120

9 LOCAL TIME STEPPING 127

10 CONCLUSIONS 129

REFERENCES 131

ANNEX A - PROOFS OF THE THEOREMS 143

ANNEX B - NERK LOCAL TIME STEPPING 167

ANNEX C - AMROC MHD SOLVER DISCUSSIONS 197

1 INTRODUCTION

Phenomena associated with the interplanetary environment can affect some aspects of modern society directly. Many of these phenomena are associated with solar activity, which presents a cycle in which those events present maximums and minimums in its frequencies and intensity (KIVELSON; RUSSELL, 1995). Several physical processes connect those phenomena to Earth's electrodynamics responses requiring the proposal and understanding of physical models (MENDES et al., 2005). Among the significant consequences of those phenomena are the risks of damage in systems that depends on satellite technologies, energy distribution systems, *etc.,* as discussed for instance in Cucinotta et al. (2015) and Schrijver et al. (2015). Hapgood (2011) discussed some of the many impacts that solar activity has over our society and highlighted the consequences of some extreme events registered in history.

In this context, the importance of the surveillance of solar activity and the study of its impacts on the Earth's atmosphere and the objects in space has increased in recent years. To deal with space environment risks, many research centres have increasingly dedicated to the space weather field development. The studies depend on the analysis of the data originated from several instruments, such as magnetometers, radio-telescopes, space probes, and others. To complement the analysis and the possible forecast of space-Earth coupling phenomena, there is a high expectation that numerical models will be used as the nowadays low atmosphere weather forecasting ones. In particular, most of the physics of the Earth's magnetosphere can be described using a magnetohydrodynamic (MHD) model. Early attempts to numerical space weather forecasting are discussed in Tóth et al. (2012), which uses an MHD model based on the seminal work of Powell (POWELL et al., 1999; POWELL, 1997). Among the necessities of the space weather field, the numerical modelling of the phenomena in this area is challenging in several aspects. On the one hand, in the physics scope, the space plasmas contain a wide variety of phenomena and scales, which varies from the order of the fraction of the Earth's radius to the order of solar radius. On the other hand, from the computational scope, the simulation of the MHD equations presents a high computational cost because it requires accurate results due to the instabilities it involves. Therefore, adaptive meshs and high performance methodologies have a vital role in this space context.

The adaptive methodologies were introduced in the 1980's to improve the performance of numerical simulations in the hydrodynamical context. For instance, one can mention the adaptive mesh refinement (BERGER, 1982; BERGER; CORELLA, 1989;

BERGER; OLIGER, 1984) and later ideas such as the seminal work on adaptive multiresolution (HARTEN, 1995), as described in the review papers of Deiterding (2011) and Domingues et al. (2011), respectively. More recently, this context was compared (DEITERDING et al., 2016) and integrated (DEITERDING; DOMINGUES, 2017; DEITERDING et al., 2018) into the Adaptive Mesh Refinement in Object oriented C^{++} (AMROC) framework reviewed in Deiterding (2011) and references therein for hydrodynamics. That framework uses a message passing interface (MPI) to perform parallel computations in a patch-structured adaptive mesh refinement environment for finite volumes methods. Moreover, AMROC has a well balanced workload distribution, implying in an excellent computational performance. Nowadays very few frameworks have those kinds of qualities. In this context, this thesis develops and implements a new ideal and resistive MHD solver for multidimensional problems using the AMROC framework, taking advantage of both the adaptive and parallel implementations provided by this framework. The developed and implemented solver was based on the CARMEN MHD module described in Gomes (2017), Gomes et al. (2015), and Domingues et al. (2013).

The MHD simulations present an inherent challenge associated with the formation of non-physical divergence components in the magnetic field, requiring a methodology capable of correcting the solution after every iteration of the time evolution process in order to maintain the Gauss's law for magnetism valid for the solution (BRACKBILL; BARNES, 1980). In this context, it is possible to propose several strategies (DEDNER et al., 2002; MUNZ et al., 2000; KAWAI, 2013; DERIGS et al., 2018) named divergence cleaning methods. Some approaches present lower computational cost. However, they do not eliminate the divergence components effectively, although they do not compromise the solution of the physical phenomena. On the other hand, other approaches mostly eliminate these divergence components effectively, with the price of a high computational cost and a slight diffusion in the solution. Several studies addressed and presented comparisons among those methods (HOPKINS, 2016; MIYOSHI; KUSANO, 2011; BALSARA; KIM, 2003; TÓTH, 2000). The construction of a perfect divergence cleaning approach is still an open problem in the high performance computing scope. In this context, to improve the development in this area, this thesis also makes a contribution by proposing a new divergence cleaning approach that combines both approaches in the Generalised Lagrange Multiplier (GLM) context introduced in Dedner et al. (2002) and Mignone et al. (2010).

Another contribution of this thesis is an adaptive wavelet-based methodology for local time stepping. This methodological approach is general and it can be applied in

both hydrodynamic and magnetohydrodynamic cases to increase the computational performance of adaptive meshes by fixing the Courant number for every cell and obtaining a time step proportional to the cell refinement. The proposed numerical solution innovates by allowing higher order methods in time. This approach was implemented in the CARMEN code developed in Roussel et al. (2003), extended to local time stepping in Domingues et al. (2008) and optimised in (DEITERDING et al., 2016).

The thesis highlights are:

- A multi-dimensional MHD solver for ideal and resistive cases which is developed and implemented in a high performance environment using adaptive mesh refinement strategies with an MPI formulation, which was published in Moreira Lopes et al. (2018a).

- A successful approach to overcome the magnetic field numerical divergence problem. This method combines the techniques presented in Dedner et al. (2002) and Munz et al. (2000). This approach uses a multigrid solver to overcome the performance limitations of the operator defined in Munz et al. (2000). It will be presented in Domingues et al. (2019a).

- A mesh refinement criteria based on the wavelet formulation proposed in Harten (1995) which is implemented in the context of patch structured adaptive mesh refinement methods and is under revision in Domingues et al. (2019b).

- An improvement to the local time stepping strategy proposed in Moreira Lopes (2014), which was published in Moreira Lopes et al. (2018b) for the second order accuracy and recently, extended to third order with a detailed stability analysis in Moreira Lopes et al. (2019).

Outline

Chapter 2 introduces the concept of plasma and the mathematical formulation that allows describing space science problems of interest and concludes with the mathematical approach for fluids obtained from the Boltzmann equation. Chapter 3 presents some simplification assumptions to describe the plasma as a single fluid, which allows obtaining a basis for magnetohydrodynamical models as the one applied for magnetosphere and Sun-Earth electrodynamical coupling.

Once the physical problem is characterised, Chapter 4 presents the numerical methods applied in the simulations of the studied equations. Chapter 5 presents different approaches to overcome the divergence problem inherent to magnetohydrodynamic simulations, including the proposed one, which requires several numerical methods, also discussed in this chapter. Chapter 6 describes the adaptive mesh refinement techniques, used to perform the mesh adaptation. Chapter 7 discusses the implementation of those methods in the proposed magnetohydrodynamic solver.

Chapter 8 presents the solvers verification tests, the comparisons among the GLM corrections, the adaptive simulations, and the magnetospheric experiment. Chapter 9 contains an article developed in parallel with this thesis, concerning the implementation of a new time adaptation strategy, which is annexed. The conclusions and future works are presented in Chapter 10. Additionally, in annex A, the proof of the theorems presented in Chapter 2 and 3 are included. Annex B and C consist of two articles published during this thesis, which contains an extension of the results from the NERK local time stepping initially developed during the Author's Master program and extended during this PhD program; and the AMROC MHD solver verification in its earlier stages, respectively.

2 PLASMA PHYSICS

Plasma is the most common state which known matter can be found in the universe. It is found in stars, the solar wind, the interplanetary medium, earth's magnetosphere and in many other structures in the universe. This state of matter is characterised by the electrons disattaching from the atoms, which allows them to roam freely through the medium, interacting with another ions and molecules. A collective behaviour is established involving the the electrically charged particles (CHEN, 2016). However, not every ionised gas is considered to be in the plasma state.

2.1 Plasma characteristics

As presented in Bittencourt (2004), to be considered a plasma, an ionised gas should present three physical factors to ensure the plasma behaviour. These characteristics are described as follows.

Macroscopic Neutrality

One of the major characteristics of the plasma is maintaining the quasi-neutrality condition, which is described as an apparent overall charge neutrality through the medium. However, in the small scales, the medium may contain charged regions and electric fields.

Under this condition, the electric field created by every charged particle must be screened by their surrounding particles of opposite charge. The range of the effects due to the electric field generated by a particle is estimated by the Debye length, calculated as:

$$\lambda_D = \left(\frac{\epsilon_0 k T_e}{n_e e^2} \right)^{\frac{1}{2}}, \tag{2.1}$$

where ϵ_0 is the permittivity constant, k is the Boltzmann constant, T_e is the electron's temperature, n_e is the density of electrons in the medium and e is the electron charge.

In this context, the quasi-neutrality condition is assured when the Debye length is considered very small when compared with the length scale $\mathcal{L}$. Thus:

$$\mathcal{L} >> \lambda_D. \tag{2.2}$$

Physically, this means that the range of the medium $\mathcal{L}$ must be much higher than the radius of action of any coulombian force inserted in the medium.

Debye screening

The quasi-neutrality condition states that the plasma may presents electric fields around charged particles in small scales. These fields should occur inside a region delimited by the Debye length, denominated Debye sphere. In order to avoid these fields to have a higher influence radius, the electrons move themselves in order to screen the electric fields at a distance around λ_D.

In this context, a high number of particles around the perturbed particle are required in order to properly screen any electrostatic field around the particle within a distance near the Debye length.

The number of particles N_D inside the Debye sphere is estimated as its volume times the density of electrons. Therefore, if

$$N_D = \frac{4}{3}\pi\lambda_D^3 n_e \approx \lambda_D^3 n_e >> 1 \tag{2.3}$$

is satisfied, the charge potential inside the Debye sphere is considered to be properly screened.

Plasma frequency

When the electric field inside the plasma is perturbed, its electrons presents a collective behaviour in order to restore the macroscopic neutrality. However, due to inertial effects, the electrons moves beyond of the equilibrium point. This effect produces a new electric field in the opposite direction, requiring a new collective motion to restore the macroscopic neutrality.

This recursive effect causes the electrons to move around the equilibrium point periodically. This cycle has an oscillation frequency, called plasma frequency, given by:

$$f_{pe} = \frac{1}{2\pi}\left(\frac{n_e e^2}{\epsilon_0 m_e}\right)^{\frac{1}{2}}, \tag{2.4}$$

where m_e is the electron mass. However, this oscillation is meddled by collisions with neutrons, disturbing the movement of the electrons inside the Debye sphere. Consequently, this effect may compromise the screening process, causing the loss of the quasi-neutrality.

In this context, the electrons are considered to move freely around the Debye sphere when their plasma frequency f_{pe} is higher than the electron-neutron collision rate

ν_{en}. Therefore, the last condition to characterise a plasma is:

$$f_{pe} > \nu_{en}, \tag{2.5}$$

where the electron-neutron collision rate is given by:

$$\nu_{en} \approx 10^{-19} n_n \left(\frac{kT_e}{m_e}\right)^{\frac{1}{2}} \tag{2.6}$$

and n_n is the neutron density inside the medium.

2.2 Mathematical description of plasma phenomena

The plasma dynamo can be described by several mathematical formulations. The most accurate is the particle model, which considers the motion of each particle inside the plasma individually, obtaining a momentum equation for each particle. However, besides being accurate, this model is computationally impractical due to the immense number of particles inside a real plasma. Therefore, some considerations have to be done in order to obtain a more practical formulation, such as the kinetic model, which maps particles with similar position and velocity collectively using analytical mechanics techniques.

2.2.1 Kinetic model

The kinetic model extends the Newtonian formulation by also mapping the particles according a velocity space, creating a 6-dimensional phase space $(\mathbf{r}, \mathbf{v})$, where $\mathbf{r} = (x, y, z)$ is the physical space and $\mathbf{v} = (v_x, v_y, v_z)$ is the velocity space.

Instead of following the particles around the medium, this formulation is focused in controlling the number of particles, indicated by $\mathcal{N}_\alpha$, inside the space between $(\mathbf{r}, \mathbf{v})$ and $(\mathbf{r} + d\mathbf{r}, \mathbf{v} + d\mathbf{v})$, where α can represent electrons, neutrons or ions, according a distribution function f_α. Thus, the number of particles inside this space is determined as:

$$d^6\mathcal{N}_\alpha(\mathbf{r}, \mathbf{v}, t) = f_\alpha(\mathbf{r}, \mathbf{v}, t)\, d^3r\; d^3v, \tag{2.7}$$

where $d^3r = dx\, dy\, dz$ and $d^3v = dv_x\, dv_y\, dv_z$. The dynamo of the plasma inside the phase space is governed by the evolution in time of the distribution function f_α, which is given by the Boltzmann Equation:

$$\frac{\partial f_\alpha}{\partial t} + \mathbf{v} \cdot \nabla_r f_\alpha + \mathbf{a} \cdot \nabla_v f_\alpha = \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll}, \tag{2.8}$$

where $\nabla_r = \mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} + \mathbf{k}\frac{\partial}{\partial z}$, $\nabla_v = \mathbf{i}\frac{\partial}{\partial v_x} + \mathbf{j}\frac{\partial}{\partial v_y} + \mathbf{k}\frac{\partial}{\partial v_z}$. The right side term is associated to plasma collisions, which can be described using several models such as the Boltzmann, Fokker-Plank and Krook collision terms (BITTENCOURT, 2004).

The Boltzmann equation describes the microscopic behaviour of the plasma effectively. However, for many applications, much of the microscopic effects can be neglected due to large scale of the problem. In these cases, the problem can be simplified by variables which describes the plasma dynamo macroscopically.

The macroscopic variables are defined as an average value of a physical quantity inside a unit of volume. Supposing an arbitrary physical quantity $\mathbf{g}$, its average value inside a unit of volume of the physical space is defined by:

$$< \mathbf{g} >_\alpha = \frac{1}{n_\alpha(\mathbf{r}, t)} \int_{\mathbf{v}} \mathbf{g} f_\alpha(\mathbf{r}, \mathbf{v}, t)\, d^3v, \tag{2.9}$$

where n_α represents the number of particles of the type α in the space between $\mathbf{r}$ and $\mathbf{r} + d\mathbf{r}$, independently of their individual velocities. This value is obtained by integrating $d^6\mathcal{N}_\alpha$ over the velocity space, thus:

$$n_\alpha(\mathbf{r}, t) = \frac{1}{d^3r} \int_{\mathbf{v}} d^6\mathcal{N}_\alpha(\mathbf{r}, \mathbf{v}, t) = \int_{\mathbf{v}} f_\alpha(\mathbf{r}, \mathbf{v}, t)\, d^3v. \tag{2.10}$$

In special, the velocity component of the particles of a certain type α can be decomposed as the sum between the fluid velocity $\mathbf{u}_\alpha$, defined as the average velocity of all particles α within a unit of volume; and the thermal velocity $\mathbf{w}_\alpha$ which describes the dispersion in the velocity component of a particle of type α in relation to the average velocity. Therefore:

$$\mathbf{v} = \mathbf{u}_\alpha + \mathbf{w}_\alpha. \tag{2.11}$$

By definition, the fluid velocity $\mathbf{u}_\alpha$ is calculated as the average value of velocity $\mathbf{v}$. Using the Equation 2.9, this value is defined as:

$$\mathbf{u}_\alpha(\mathbf{r}, t) = < \mathbf{v} >_\alpha = \frac{1}{n_\alpha(\mathbf{r}, t)} \int_{\mathbf{v}} \mathbf{v} f_\alpha(\mathbf{r}, \mathbf{v}, t)\, d^3v. \tag{2.12}$$

Consequently, the average value for the random thermal velocity is:

$$< \mathbf{w}_\alpha(\mathbf{r}, t) >_\alpha = < \mathbf{v} - \mathbf{u}_\alpha >_\alpha = < \mathbf{v} >_\alpha - < \mathbf{u}_\alpha >_\alpha = \mathbf{0}, \tag{2.13}$$

since $< \mathbf{u}_\alpha >_\alpha = \mathbf{u}_\alpha$ is already an average value. In Goedbloed and Poedts (2004), the

thermal component $\mathbf{w}_\alpha$ allows the definition of the thermal macroscopic quantities:

$$T_\alpha(\mathbf{r},t) = \frac{m_\alpha}{N_d k} < \|\mathbf{w}\|^2 >_\alpha \qquad \text{(Temperature)} \tag{2.14a}$$

$$\mathbf{h}_\alpha(\mathbf{r},t) = \frac{1}{2} n_\alpha m_\alpha < \|\mathbf{w}\|^2 \mathbf{w} >_\alpha \qquad \text{(Heat flow)} \tag{2.14b}$$

$$\mathbb{P}_\alpha(\mathbf{r},t) = n_\alpha m_\alpha < \mathbf{w}\mathbf{w} >_\alpha \qquad \text{(Stress tensor)} \tag{2.14c}$$

where N_d is the number of degrees of freedom available to the particles. This number is used to define the adiabatic index γ of the gas:

$$\gamma = \frac{C_p}{C_V} = \frac{N_d + 2}{N_d}, \tag{2.15}$$

where C_p and C_V are the specific heats of constant pressure and volume, respectively.

The stress tensor $\mathbb{P}_\alpha$ can be decomposed as the sum between its isotropic and anisotropic contributions as:

$$\mathbb{P}_\alpha = p_\alpha \mathbb{I} + \pi_\alpha, \tag{2.16}$$

where the isotropic part p_α is given in function of the temperature and particle density:

$$p_\alpha = n_\alpha k T_\alpha, \tag{2.17}$$

and the anisotropic part π_α is a tensor with null elements in its diagonal.

2.2.2 Fluid model

Plasmas that presents a high frequency of particle collisions, so that each particle of type α maintains a local equilibrium distribution function, can be described as a conducting fluid. Mathematically, this formulation describes the plasma dynamo using macroscopic variables such as density, macroscopic velocity and temperature.

In particular, considered as particles in thermal equilibrium, the particles with velocity $\mathbf{v}$ are distributed over the phase space according to the Maxwellian distribution:

$$f_\alpha(\mathbf{v}) = \left(\frac{m_\alpha}{2\pi k T_\alpha}\right)^{\frac{3}{2}} e^{\frac{-m_\alpha \|\mathbf{v}\|^2}{2kT_\alpha}} \tag{2.18}$$

The fluid model treats the plasma as a mixture of many interpenetrating fluids so that each fluid is composed by a type of particles α. Therefore, in this formulation, a set of equations is obtained to describe the dynamo of each fluid composed by particle of the type α. In special, the single fluid formulation presented in Chapter

3, also know as magnetohydrodynamics, considers the mixture of ions and electrons as a single fluid unit.

The system of equations that describes the fluid approximation is obtained by taking as many moments of the Boltzmann Equation as desirable, then converting the microscopic variables into macroscopic variables. The fluid formulation studied in this work, farther presented in Chapter 3, approximates the plasma dynamo until the second moment. These moments are taken by applying the following integral operators to the Boltzmann Equation:

$$\underbrace{m_\alpha \int_{\mathbf{v}} d^3v}_{0^{th}\text{ moment}} \qquad \underbrace{m_\alpha \int_{\mathbf{v}} \mathbf{v} d^3v}_{1^{st}\text{ moment}} \qquad \underbrace{m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 d^3v}_{2^{nd}\text{ moment}} \tag{2.19}$$

This process results in the system of equations obtained in the Theorem 1, proven is in Annex A.

Theorem 1. *The moments of the Boltzmann Equation produces the following set of equations:*

$$\frac{\partial n_\alpha m_\alpha}{\partial t} + \nabla \cdot (n_\alpha m_\alpha \mathbf{u}_\alpha) = S_\alpha \tag{2.20a}$$

$$\frac{\partial \left(n_\alpha m_\alpha \mathbf{u}_\alpha\right)}{\partial t} + \nabla \cdot \left[n_\alpha m_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha + \mathbb{P}_\alpha\right] - n_\alpha q_\alpha \left(\mathbf{E} + \mathbf{u}_\alpha \times \mathbf{B}\right) = \mathbf{A}_\alpha. \tag{2.20b}$$

$$\frac{1}{2}\frac{\partial n_\alpha m_\alpha \|\mathbf{u}_\alpha\|^2}{\partial t} + \frac{\partial N_d k n_\alpha T_\alpha}{\partial t} + \nabla \cdot \left(\frac{1}{2} n_\alpha m_\alpha \|\mathbf{u}_\alpha\|^2 \mathbf{u}_\alpha + \frac{1}{2} N_d k n_\alpha T_\alpha \mathbf{u}_\alpha + \mathbf{u}_\alpha \cdot \mathbb{P}_\alpha + \mathbf{h}_\alpha\right) - n_\alpha q_\alpha \mathbf{E} \cdot \mathbf{u}_\alpha = \mathbf{u}_\alpha \cdot \mathbf{A}_\alpha + Q_\alpha \tag{2.20c}$$

where q_α is the electric charge of a particle α, S_α is rate per unit of volume which the particles α are produced or lost due to collision effects, $\mathbf{A}$ is the rate of change of the momentum of the particles α due to collisions with particles of another types, Q_α is generated heat per unit of volume of the particles α due to collisions with another particles, $\mathbf{B}$ is the magnetic field and $\mathbf{E}$ is the electric field.

Each moment of the Boltzmann Equations produces an equation that describes the evolution of one physical quantity. However, every equation contains a term that is described by the equation in the next momentum. In particular, these moments produces the following equations:

- The 0^{th} moment describes the evolution of the term n_α, but it introduces the variable $\mathbf{u}_\alpha$, described by the next moment, into the system.

- The 1^{st} moment describes the evolution of the term $\mathbf{u}_\alpha$, but it introduces the variables $\mathbf{B}$, $\mathbf{E}$ and $\mathbb{P}$, which can be associated with the variable T_α, described in the next moment.
- The 2^{nd} moment describes the evolution of the term T_α, but it introduces the variable $\mathbf{h}_\alpha$ to the system.

As noted, the procedure of taking moments of the Boltzmann equation always presents more parameters than equations, making the system with non-unique solutions. To bypass this problem, the system should be completed by another set of equations which uses the same parameters, and some terms might be neglected. In special, for the fluid formulation presented in Chapter 3, the system is completed by the Maxwell Equations, which describes the behaviour of magnetic and electric fields as:

$$\nabla \cdot \mathbf{B} = 0 \qquad \text{(Gauss' law - Magnetism)} \tag{2.21a}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \qquad \text{(Faraday's law)} \tag{2.21b}$$

$$\nabla \times \mathbf{B} = \mathbf{J} \qquad \text{(Ampère's law)} \tag{2.21c}$$

$$\nabla \cdot \mathbf{E} = \frac{\rho_C}{\epsilon_0} \qquad \text{(Gauss' law)} \tag{2.21d}$$

where $\mathbf{J}$ is the current density and ρ_C is the charge density. Alongside with the Maxwell equations, the system of equations is also completed with the Ohm's law:

$$\mathbf{E} + \mathbf{u} \times \mathbf{B} = \eta \mathbf{J}, \tag{2.22}$$

where η is the resistivity. In the next chapter, the relations presented in this chapter are used to describe the magnetohydrodynamic model, which considers the plasma as a single fluid, instead of a mixture of particles α.

3 MAGNETOHYDRODYNAMICS

The Magnetohydrodynamic (MHD) model describes the plasma dynamo as a single conductor fluid under a magnetic field. For that, it neglects the particularity of the motion of each particle type so that the model considers the group of ions and electrons as a single fluid element.

Mathematically, the MHD model describes the plasma by the set of single fluid macroscopic variables $\mathfrak{q}(\mathbf{r},t) = (\rho, \rho\mathbf{u}, \mathcal{E}, \mathbf{B})$, corresponding to density, momentum, internal energy and magnetic field, respectively. The single fluid variables are defined as the sum of terms associated to one type of particle, such as:

$$\rho(\mathbf{r},t) = \sum_\alpha n_\alpha m_\alpha \qquad \text{(Density)} \tag{3.1a}$$

$$\rho_C(\mathbf{r},t) = \sum_\alpha n_\alpha q_\alpha \qquad \text{(Charge density)} \tag{3.1b}$$

$$\rho\mathbf{u}(\mathbf{r},t) = \sum_\alpha n_\alpha m_\alpha \mathbf{u}_\alpha \qquad \text{(Momentum)} \tag{3.1c}$$

$$\mathbf{J}(\mathbf{r},t) = \sum_\alpha n_\alpha q_\alpha \mathbf{u}_\alpha \qquad \text{(Current density)} \tag{3.1d}$$

$$p(\mathbf{r},t) = \sum_\alpha p_\alpha \qquad \text{(Isotropic pressure)} \tag{3.1e}$$

The MHD equations are obtained by summing the fluid equations obtained in the previous chapter for electrons and ions. Then, after applying the MHD model assumptions, the sum of variables associated to one type of particle are converted to the single fluid macroscopic variables according to the Equations 3.1.

This model is particularly useful to describe problems which the plasma has macroscopic force balance, equilibrium and dynamic. In special, for space sciences, phenomena like the magnetosphere, the solar wind and heliosphere are well represented by the MHD formulation. In the context of laboratory plasmas, the MHD model poorly represents their dynamo. However, it is a good stability predictor for those plasmas.

In order to reduce the mathematical complexity of the phenomena, the following assumptions are made in addiction to the approximations presented in the previous chapter:

Electron mass neglected: considering that $m_i >> m_e$, where m_i and m_e corre-

sponds to the mass of the ions and electrons, respectively, the terms with m_e are neglected of the formulation. Thus the plasma density became $\rho = n_i m_i$, characterising a low-frequency plasma.

Mass and momentum conservation: the system is assumed to neither produce or loss particles. Therefore the collision term $S_\alpha = 0$. Furthermore, the collision between particles of different types are assumed to conserve the momentum in the system, thus $\sum_\alpha \mathbf{A}_\alpha = 0$.

Isotropic pressure: the anisotripic component of stress tensor is neglected. Therefore $\mathbb{P}_\alpha = p_\alpha \mathbb{I}$.

Adiabatic process: the system is assumed to not transfer heat. Therefore, the heat flux $\mathbf{h}_\alpha$ and the generated heat Q_α are neglected.

3.1 Ideal MHD

The ideal MHD is the simplest representation of the dynamo of plasma as a conducting fluid. This formulation is an adiabatic process that neglects dissipative terms such as η, π_α and $\mathbf{h}_\alpha$, corresponding to resistivity, viscosity and heat conductivity effects, respectively. Thus, under the ideal MHD formulation, the plasma conserves the physical properties of mass, momentum, energy, helicity and the magnetic field topology due to the absence of dissipation. Considering these assumptions, the Theorem 2, proven in the Annex A, presents the MHD equations.

Theorem 2. *Applying the Ideal MHD properties to the fluid formulation obtained in Section 2.2.2 produces the following equations:*

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{3.2a}$$

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot \left[\rho \mathbf{u}\mathbf{u} + \left(p + \frac{\mathbf{B} \cdot \mathbf{B}}{2} \right) \mathbb{I} - \mathbf{B}\mathbf{B} \right] = \mathbf{0} \tag{3.2b}$$

$$\frac{\partial \mathcal{E}}{\partial t} + \nabla \cdot \left[\left(\mathcal{E} + p + \frac{\mathbf{B} \cdot \mathbf{B}}{2} \right) \mathbf{u} - (\mathbf{u} \cdot \mathbf{B}) \mathbf{B} \right] = 0 \tag{3.2c}$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot [\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u}] = \mathbf{0}, \tag{3.2d}$$

where this system is completed with the internal energy equation, given by the combination of the hydrodynamic and magnetic energies:

$$\mathcal{E} = \frac{p}{\gamma - 1} + \rho \frac{\mathbf{u} \cdot \mathbf{u}}{2} + \frac{\mathbf{B} \cdot \mathbf{B}}{2}, \tag{3.3}$$

and γ is the adiabatic index.

This formulation, called conservative form, describes the evolution of physical quantities that in the absence of dissipative effects, should be conserved. Thus:

$$\frac{\partial}{\partial t}\int \rho d^3r = 0 \qquad \frac{\partial}{\partial t}\int \rho \mathbf{u} d^3r = \mathbf{0} \qquad \frac{\partial}{\partial t}\int \mathcal{E} d^3r = 0 \qquad \frac{\partial}{\partial t}\int \mathbf{B} d^3r = \mathbf{0}. \quad (3.4)$$

The numerical simulations of the MHD equations performed in this work are based in this conservative form due to the formulation of the numerical methods chosen for those simulations, presented in Chapter 4.

3.1.1 System eigenvalues

The numerical methods for the simulation of the MHD model, presented in Chapter 4, requires the eigenvalues associated with each spacial derivative of the system. These eigenvalues are obtained from the matrices multiplying their respective derivatives when the system is written in the form of an advection equation:

$$\frac{\partial \mathbf{w}}{\partial t} + \mathbb{A}_x \frac{\partial \mathbf{w}}{\partial x} + \mathbb{A}_y \frac{\partial \mathbf{w}}{\partial y} + \mathbb{A}_z \frac{\partial \mathbf{w}}{\partial z} = \mathbf{0}, \quad (3.5)$$

where $\mathbb{A}_x$, $\mathbb{A}_y$ and $\mathbb{A}_z$ are matrices associated with the transport in the **i**, **j** and **k** directions, respectively, and **w** is a vector of variables. However, the terms of the conservative MHD equations can not be expressed in this formulation. Thus, the system of equations is rewritten as the primitive formulation.

The primitive formulation describes the MHD equations based in the vector of variables $\mathbf{w} = (\rho, p, v_x, v_y, v_z, B_x, B_y, B_z)$. The system of equations that rules the MHD model using the primitive variables is obtained by reordering the terms of the conservative formulation, except for the energy equation. This equation is substituted by the pressure equation deducted in the Theorem 3, proven in the Annex A. This equations requires the alternative formulation of the second moment of the Boltzmann Equation, presented in Equation A.82.

Theorem 3. *The pressure equation for Ideal MHD is given by:*

$$\frac{\partial p}{\partial t} + (\mathbf{u}\cdot\nabla)\, p = -\gamma p \nabla\cdot\mathbf{u} \quad (3.6)$$

Thus, the system of Ideal MHD equations in the primitive form is written as:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{3.7a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\, \mathbf{u} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\,(\nabla \times \mathbf{B}) \times \mathbf{B} \tag{3.7b}$$

$$\frac{\partial p}{\partial t} + (\mathbf{u} \cdot \nabla)\, p = -\gamma p \nabla \cdot \mathbf{u} \tag{3.7c}$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot [\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u}] = 0 \tag{3.7d}$$

This system of equations is expressed in the form of an advection equation, conforming Equation 3.5, using the matrices $\mathbb{A}_a$, where a stands for the axes x, y or z, calculated as:

$$\mathbb{A}_x = \begin{bmatrix} u_x & 0 & \rho & 0 & 0 & 0 & 0 & 0 \\ 0 & u_x & \gamma p & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\rho} & u_x & 0 & 0 & -\frac{B_x}{\rho} & \frac{B_y}{\rho} & \frac{B_z}{\rho} \\ 0 & 0 & 0 & u_x & 0 & -\frac{B_y}{\rho} & -\frac{B_x}{\rho} & 0 \\ 0 & 0 & 0 & 0 & u_x & -\frac{B_z}{\rho} & 0 & -\frac{B_x}{\rho} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & B_y & -B_x & 0 & -u_y & u_x & 0 \\ 0 & 0 & B_z & 0 & -B_x & -u_z & 0 & u_x \end{bmatrix} \tag{3.8a}$$

$$\mathbb{A}_y = \begin{bmatrix} u_y & 0 & 0 & \rho & 0 & 0 & 0 & 0 \\ 0 & u_y & 0 & \gamma p & 0 & 0 & 0 & 0 \\ 0 & 0 & u_y & 0 & 0 & -\frac{B_y}{\rho} & -\frac{B_x}{\rho} & 0 \\ 0 & \frac{1}{\rho} & 0 & u_y & 0 & \frac{B_x}{\rho} & -\frac{B_y}{\rho} & \frac{B_z}{\rho} \\ 0 & 0 & 0 & 0 & u_y & 0 & -\frac{B_z}{\rho} & -\frac{B_y}{\rho} \\ 0 & 0 & -B_y & B_x & 0 & u_y & -u_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & B_z & -B_y & 0 & -u_z & u_y \end{bmatrix} \tag{3.8b}$$

$$\mathbb{A}_z = \begin{bmatrix} u_z & 0 & 0 & 0 & \rho & 0 & 0 & 0 \\ 0 & u_z & 0 & 0 & \gamma p & 0 & 0 & 0 \\ 0 & 0 & u_z & 0 & 0 & -\frac{B_z}{\rho} & 0 & -\frac{B_x}{\rho} \\ 0 & 0 & 0 & u_z & 0 & 0 & -\frac{B_z}{\rho} & -\frac{B_y}{\rho} \\ 0 & \frac{1}{\rho} & 0 & 0 & u_z & \frac{B_x}{\rho} & \frac{B_y}{\rho} & -\frac{B_z}{\rho} \\ 0 & 0 & -B_z & 0 & B_x & u_z & 0 & -u_x \\ 0 & 0 & 0 & -B_z & B_y & 0 & u_z & -u_y \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.8c}$$

Each one of these matrices presents seven non-zero eigenvalues, which can be ordered in ascending order as:

$$\lambda_f^- \leq \lambda_A^- \leq \lambda_s^- \leq \lambda_e \leq \lambda_s^+ \leq \lambda_A^+ \leq \lambda_f^+. \tag{3.9}$$

These eigenvalues are calculated for every axis a in Gomes (2012), obtaining:

$$\lambda_e = u_a, \qquad \lambda_f^{\pm} = u_a \pm c_f, \qquad \lambda_s^{\pm} = u_a \pm c_s, \qquad \lambda_A^{\pm} = u_a \pm c_A \tag{3.10}$$

where the velocities c_f, c_s and c_A are defined as:

$$c_f = \sqrt{\frac{1}{2}\left[\kappa_1^2 + \kappa_2^2 + \sqrt{\left(\kappa_1^2 + \kappa_2^2\right)^2 - 4\kappa_1^2\kappa_a^2}\right]} \quad \text{(Fast magnetosonic velocity)} \tag{3.11a}$$

$$c_s = \sqrt{\frac{1}{2}\left[\kappa_1^2 + \kappa_2^2 - \sqrt{\left(\kappa_1^2 + \kappa_2^2\right)^2 - 4\kappa_1^2\kappa_a^2}\right]} \quad \text{(Slow magnetosonic velocity)} \tag{3.11b}$$

$$c_A = \|\kappa_a\| \quad \text{(Alfvén velocity)} \tag{3.11c}$$

with the values κ given as:

$$\kappa_a = \sqrt{\frac{B_a^2}{\rho}}, \qquad \kappa_1 = \sqrt{\frac{\gamma p}{\rho}}, \qquad \kappa_2 = \frac{\|\mathbf{B}\cdot\mathbf{B}\|}{\rho}. \tag{3.12}$$

3.2 Resistive MHD

This formulation is a more realistic description of the plasma dynamo that is obtained by considering the resistive effects obtained in the energy and induction equation, which are neglected in the Ideal MHD model. The resistive effects causes a diffusion in the magnetic field lines, allowing the study of a new range of phenomena beyond of the described by the Ideal MHD.

Physically, the resistive formulation does not preserve the topology of the magnetic field lines, allowing magnetic reconnections, which are responsible for many phenomena, such as Coronal Mass Ejections (KIVELSON; RUSSELL, 1995).

However, besides being more realistic than the Ideal MHD, the resistive MHD formulation is not applied in every situation due to the increasing of both the theoretical and computational efforts to study the plasma. Therefore, the resistive formulation is applied when his effects are significant to the physics of the problem.

The relevance of the resistive effects over the problem studied is measured by the magnetic Reynolds number R_m defined by:

$$R_m := \frac{\mathcal{U}\mathcal{L}}{\eta}, \tag{3.13}$$

where $\mathcal{U}$ and $\mathcal{L}$ are the typical velocity and length scale of the flow and η is the resistivity. The magnetic Reynolds number is a dimensionless number that represents the ratio between the magnitude of the advective and diffusive effects. Thus, a low magnetic Reynolds number ($R_m << 1$) implies in a medium considerably influenced by resistive effects, requiring the usage of the resistive MHD model. In contrast, a high magnetic Reynolds number ($R_m >> 1$) implies in a medium dominated by advective effects, which the Ideal MHD formulation is more suitable.

The equations that describes the resistive MHD model are obtained analogously as the Ideal MHD equations, but without dropping the terms associated with the resistivity η. In special, the continuity and the momentum equations in the resistive model are identical to the obtained for the ideal model, whilst the resistive energy and induction equations are obtained by applying the Ampère's law into the Equations A.114 and A.117. However, the resistive induction equation requires some modifications to be written in the conservation form, which is performed in the Theorem 4, proven in the Annex A.

Theorem 4. *The resistive induction equation, obtained in Equation A.117, can be rewritten in the conservative form as:*

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot \left[\mathbf{uB} - \mathbf{Bu} + \eta \left((\nabla \mathbf{B})^T - \nabla \mathbf{B} \right)\right] = 0 \tag{3.14}$$

Therefore the resistive MHD model is governed by the following equations in the

conservation form:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{3.15a}$$

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot \left[\rho \mathbf{u}\mathbf{u} + \left(p + \frac{\mathbf{B} \cdot \mathbf{B}}{2} \right) \mathbb{I} - \mathbf{B}\mathbf{B} \right] = \mathbf{0} \tag{3.15b}$$

$$\frac{\partial \mathcal{E}}{\partial t} + \nabla \cdot \left[\left(\mathcal{E} + p + \frac{\|\mathbf{B}\|^2}{2} \right) \mathbf{u} - (\mathbf{u} \cdot \mathbf{B})\, \mathbf{B} + (\eta \mathbf{J}) \times \mathbf{B} \right] = 0 \tag{3.15c}$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot \left[\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u} + \eta \left((\nabla \mathbf{B})^T - \nabla \mathbf{B} \right) \right] = 0 \tag{3.15d}$$

Defined the studied equations, the next chapter presents the numerical strategies applied to perform the MHD simulations.

4 NUMERICAL FORMULATION

This chapter presents the numerical schemes used in the simulations of the MHD model. The first step to perform these simulations is to discretise the problem in space and, in case of evolutive PDEs, time. The spatial discretisation consists into representing the physical domain of the problem by using a set of elements, such as mesh points or cells, so that each element is associated with a value representing the solution locally.

The major approaches for the spatial discretisation are the Finite Volumes (FV) (LEVEQUE, 1990), Finite Differences (FD) (LEVEQUE, 2007) and the Finite Elements (FE) (DHATT et al., 2012) methods. The FV method has the advantages of allowing an explicit approach, which requires considerably less computational efforts. Moreover, it is constructed under an integral formulation, obtaining good results in solutions with discontinuities and being suitable for complex geometries. However, this approach has the disadvantage of being difficult to obtain high order solutions. In the context of MHD equations, the FV formulation is used in works such as Felker and Stone (2018), Balsara and Dumbser (2015) and Shakeri and Dehghan (2011). The FD method is both easier to implement and to obtain high order solutions, also allowing the explicit approach. However, its differential formulation does not present good results in solutions with discontinuities and it may require complex approximations to handle complex geometries. This method is applied to MHD equations in Do et al. (2017), Christlieb et al. (2014) and Mignone et al. (2010). Lastly, the FE method has the advantages of both FV and FD methods, such as being good for complex geometries, the integral formulation and the simplicity to obtain high orders. However, it does not support an explicit approach in time. The FE method is applied for MHD in works such as Yang et al. (2018), Li and Zheng (2017) and Basting and Kuzmin (2017).

The time discretisation is defined as a finite enumeration of the time instants which the solution is represented during the evolutive PDE simulation. These instants are determined during the simulations so that the time step, defined as the difference between two consecutive instants of the time discretisation, follows the CFL condition, which is a constraint that delimits the size of the time step based on the spatial discretisation and the eigenvalues of the problem, as presented in Section 4.5.

Defined the discretisation methods, the numerical simulation consists in evolving the value of each element of the spatial discretisation from one instant of the time discretisation to the next successively until the predetermined final time t_e is reached.

For that is required an initial condition, which is a description of the initial state of the solution, and the boundary conditions, which describes the behaviour of the system at the edges of the physical domain.

In this work, the spatial discretisation is done by the FV method, presented in Section 4.1. This choice is justified by the necessity, in the scope of space weather, for numerical schemes that handles shocks and can be executed in a reasonable time. The FV method depends of numerical schemes to compute the flux of the physical quantities in the interfaces between adjacent cells. The implemented MHD code uses the schemes presented in the Section 4.2 to compute those fluxes. The Section 4.2.1 presents a numerical technique to increase the accuracy of these schemes while avoiding unwanted oscillations caused by strong shocks. The time evolution is performed using Runge–Kutta methods (BURDEN; FAIRES, 1989), as discussed in Section 4.4.

4.1 Finite Volumes method

The FV method is an adequate approach for solving problems concerning about conservation laws, such as transport of energy, mass and heat (LEVEQUE, 1990). Considering a vector of physical quantities $\mathfrak{q}$, the laws which rules these quantities can be written in the form:

$$\frac{\partial \mathfrak{q}}{\partial t} + \nabla \cdot \mathbb{F}\left(\mathfrak{q}\right) = \mathbf{S}\left(\mathfrak{q}\right), \tag{4.1}$$

where $\mathbb{F}$ is a matrix with the flux components and $\mathbf{S}$ is the source term. Considering the MHD model presented in Chapter 3, this formulation can be obtained using the conservative variables $\mathfrak{q} = (\rho, \rho\mathbf{u}, \mathcal{E}, \mathbf{B})$ as presented in the Equations 3.2.

This formulation is considered fully conservative when $\mathbf{S}\left(\mathfrak{q}\right) = \mathbf{0}$, which is valid for both Ideal and resistive MHD models. The Chapter 8.4 presents a non-ideal MHD model containing some physical terms, such as gravity, that does not have a formulation which can be included in the flux vector, causing the loss of conservation in $\mathfrak{q}$.

Considering a three dimensional problem in a Cartesian mesh, the discretisation of a physical domain $\Omega = [x_s, x_e] \times [y_s, y_e] \times [z_s, z_e]$ using the FV method is done by dividing this domain into cells $\mathcal{C}_{i,j,k}$ centred at points (x_i, y_j, z_k), with $x_i = x_s + \left(i + \frac{1}{2}\right)\Delta x$, where $\Delta x = \dfrac{x_e - x_s}{N_x}$ is the length of the cell in the x direction and N_x is the number of cells in this direction. The positions y_j and z_k have analogous

definitions using the y and z axis, respectively.

For each cell $\mathcal{C}_{i,j,k}$, the solution $\mathfrak{q}$ is represented by an average value $\mathbf{q}_{i,j,k}$ given by:

$$\mathbf{q}_{i,j,k}(t) = \frac{1}{\Delta^3 r} \int_{\mathcal{C}_{i,j,k}} \mathfrak{q}(x, y, z, t) d^3 r, \tag{4.2}$$

where $\Delta^3 r = \Delta x \Delta y \Delta z$ is the volume of $\mathcal{C}_{i,j,k}$ and $d^3 r = dxdydz$ is the differential of the volume defined by the space between $\mathbf{r}$ and $\mathbf{r} + d\mathbf{r}$.

The equation that describe the evolution of the average value $\mathbf{q}$ in every cell is obtained by integrating the Equation 4.1 over a cell $\mathcal{C}_{i,j,k}$:

$$\int_{\mathcal{C}_{i,j,k}} \frac{\partial \mathfrak{q}}{\partial t} d^3 r = -\int_{\mathcal{C}_{i,j,k}} \nabla \cdot [\mathbb{F}(\mathfrak{q})] \, d^3 r + \int_{\mathcal{C}_{i,j,k}} \mathbf{S}(\mathfrak{q}) \, d^3 r \tag{4.3}$$

Using the Leibniz' rule in the time derivative, the Gauss' rule at the divergence term and dividing by the volume $\Delta^3 r$, the Equation 4.3 became:

$$\frac{d}{dt}\left(\frac{1}{\Delta^3 r} \int_{\mathcal{C}_{i,j,k}} \mathfrak{q} d^3 r\right) = -\frac{1}{\Delta^3 r} \oint_{\partial \mathcal{C}_{i,j,k}} \mathbb{F}(\mathfrak{q}) \cdot \mathbf{n} \, d\mathcal{S} + \frac{1}{\Delta^3 r} \int_{\mathcal{C}_{i,j,k}} \mathbf{S}(\mathfrak{q}) \, d^3 r, \tag{4.4}$$

where $\mathbf{n}$ is vector normal to the surface $\mathcal{S}$ which delimits the boundaries of the cell $\mathcal{C}$. Applying the Equation 4.2 and the average function value theorem for the source term integral:

$$\frac{d}{dt}\mathbf{q}_{i,j,k} = -\frac{1}{\Delta^3 r} \oint_{\partial \mathcal{C}_{i,j,k}} \mathbb{F}(\mathfrak{q}) \cdot \mathbf{n} \, d\mathcal{S} + \mathbf{S}(\mathbf{q}_{i,j,k}) \tag{4.5}$$

Splitting the surface integrals for each face of the cell $\mathcal{C}_{i,j,k}$ and applying the average function value theorem in these integrals, the formulation of the FV method is obtained as:

$$\frac{d}{dt}\mathbf{q}_{i,j,k} = \frac{\mathcal{F}^-_{i,j,k} - \mathcal{F}^+_{i,j,k}}{\Delta x} + \frac{\mathcal{G}^-_{i,j,k} - \mathcal{G}^+_{i,j,k}}{\Delta y} + \frac{\mathcal{H}^-_{i,j,k} - \mathcal{H}^+_{i,j,k}}{\Delta z} + \mathbf{S}(\mathbf{q}_{i,j,k}), \tag{4.6}$$

where $\mathcal{F}^-_{i,j,k}$ is an average value representing the flux between the cells centred at (x_{i-1}, y_j, z_k) and (x_i, y_j, z_k). The flux $\mathcal{F}^+_{i,j,k}$ is defined between the cells (x_i, y_j, z_k) and (x_{i+1}, y_j, z_k). Therefore, by definition $\mathcal{F}^+_{i,j,k} = \mathcal{F}^-_{i+1,j,k}$. This property guarantee the conservation in $\mathfrak{q}$ for the FV method if $\mathbf{S}(\mathfrak{q}) = 0$, since every quantity which is entering in one cell is being subtracted from another cell.

The fluxes $\mathcal{G}$ and $\mathcal{H}$ have analogous definitions as $\mathcal{F}$, but for the y and z axis, respectively. The fluxes $\mathcal{F}^-_{i,j,k}$ and $\mathcal{F}^+_{i,j,k}$ are defined by the respective following integrals

over the faces in $x_i - \frac{\Delta x}{2}$ and $x_i + \frac{\Delta x}{2}$:

$$\mathcal{F}^{\mp}_{i,j,k} = \iint_{\partial \mathcal{C}_{i\mp\frac{1}{2},j,k}} \mathbf{F}\left[\mathfrak{q}\left(x_i \mp \frac{\Delta x}{2}, y, z, t\right)\right] dydz \tag{4.7}$$

These values are computed by techniques called numerical fluxes, which are discussed in the Section 4.2. Considering that the flux values $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ have analogous definitions and therefore are computed by analogous techniques, from this point the notations are simplified in the following sections by treating the problem as one-dimensional, returning to the three-dimensional notation when necessary.

4.2 Numerical Fluxes

The next step to perform the FV method resides on the computation of the fluxes $\mathcal{F}$ defined in Equation 4.7. Each flux could be interpreted as an evaluation of $\mathbf{F}$ in the interface between two adjacent cells. However, due to these cells having different average values, there is a discontinuity in the solution located in the interface. Thus, the interface region can be described as the following piece-wise constant function containing a single discontinuity:

$$\mathbf{q} = \begin{cases} \mathbf{q}^{+}_{i-1}, & \text{if } x < x_{i-1} + \frac{1}{2}\Delta x \\ \mathbf{q}^{-}_{i}, & \text{if } x > x_{i-1} + \frac{1}{2}\Delta x \end{cases}, \tag{4.8}$$

where $\mathbf{q}^{+}_{i-1}$ is a reconstruction of the solution $\mathbf{q}_{i-1}$ in the right side interface of the cell $\mathcal{C}_{i-1}$, while $\mathbf{q}^{-}_{i}$ is a reconstruction of the solution $\mathbf{q}_i$ in the left side interface of the cell $\mathcal{C}_i$. Further details of these reconstructions are given in Section 4.2.1.

This configuration as an initial condition combined with a conservation law is called a Riemann problem (TORO, 1999). For sake of simplicity, this initial condition is rewritten as:

$$\mathbf{q} = \begin{cases} \mathbf{q}^{L}, & \text{if } x' < 0 \\ \mathbf{q}^{R}, & \text{if } x' > 0 \end{cases}, \tag{4.9}$$

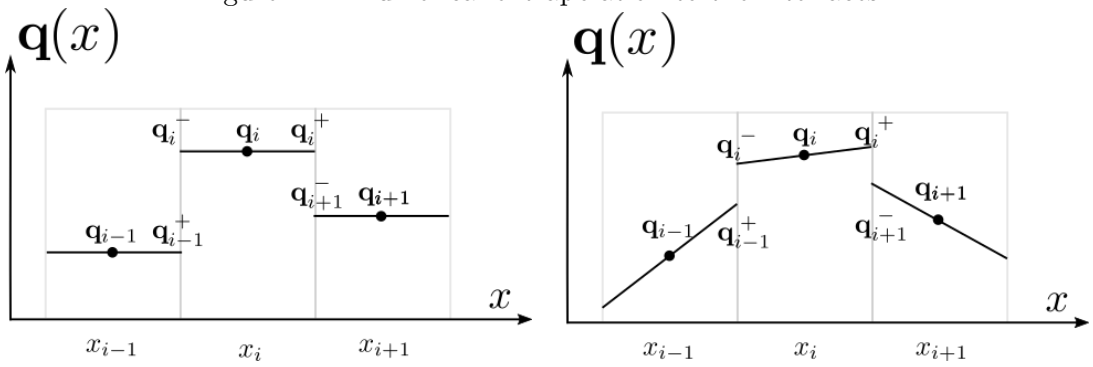
where x' is an auxiliary axis, parallel to x, centred at $x_{i-1} + \frac{1}{2}\Delta x$. The solution of the Riemann problem provides an average value at the studied interface, $x' = 0$, during the time step from t^n to $t^n + \Delta t$. Therefore, an average value for the flux $\mathcal{F}$ is also obtained from this solution.

4.2.1 High-Resolution Schemes

In the context of FV methods, the earlier first order Godunov's schemes considers the solution in the entire cell as the constant average value $\mathbf{q}_i$. Therefore, the interface values for a cell $\mathcal{C}_i$ are $\mathbf{q}_i^- = \mathbf{q}_i^+ = \mathbf{q}_i$, as illustrated in Figure 4.1a. These schemes were proven unable to provide high order solutions without unwanted spurious oscillations (GODUNOV, 1959), causing numerical instabilities or incorrect solutions.

These oscillations in high order solutions are avoided by the usage of high-resolution schemes, which are characterised by obtaining a high accuracy around shocks and discontinuities and maintaining the solution free from spurious oscillations, while obtaining a spatial accuracy of at least second order around smooth regions. Furthermore, a high-resolution scheme requires fewer mesh elements than a first order scheme to obtain a similar accuracy.

Figure 4.1 - Numerical extrapolation to the interfaces.



a) Godunov constant approximation. b) MUSCL reconstruction.
Source: Author's production.

Among the high-resolution schemes for FV methods, popular approaches are the Monotone Upstream-Centered Schemes for Conservation Laws (MUSCL) (van LEER, 1979), applied in MHD equations in works such as Moreira Lopes et al. (2018a), and the Weighted Essentially Non-Oscillatory (WENO) schemes (LIU et al., 1994), applied in Titarev and Toro (2004) for 3D Euler equations and in Xu et al. (2016) for ideal MHD equations.

The WENO schemes are able to obtain higher resolutions than MUSCL schemes. However, they may require more computational cost and larger stencils. In this context, the MUSCL schemes are chosen for the proposed MHD solver due to its computational efficiency and good resolution around discontinuities.

The MUSCL approach is based in the use of slope limiters, consisting in the use of a piece-wise linear reconstruction to extrapolate each variable $q_{\theta,\,i}$ of the solution $\mathbf{q}_i$ to the left and right boundaries of $\mathcal{C}_i$. The left and right extrapolations, denoted by $q^{-}_{\theta,\,i}$ and $q^{+}_{\theta,\,i}$, are used to reconstruct the respective interface solutions $\mathbf{q}_i{}^{-}$ and $\mathbf{q}_i{}^{+}$. These extrapolations are illustrated in the Figure 4.1b.

To perform those extrapolations, Toro (1999) suggested the following relations:

$$q^{+}_{\theta,\,i} = q_{\theta,\,i} + \frac{1}{4}\left[(1-\omega)\,\Delta^{L}_{\theta,\,i}\,\phi\left(\frac{\Delta^{R}_{\theta,\,i}}{\Delta^{L}_{\theta,\,i}}\right) + (1+\omega)\,\Delta^{R}_{\theta,\,i}\,\phi\left(\frac{\Delta^{L}_{\theta,\,i}}{\Delta^{R}_{\theta,\,i}}\right)\right] \tag{4.10a}$$

$$q^{-}_{\theta,\,i} = q_{\theta,\,i} - \frac{1}{4}\left[(1+\omega)\,\Delta^{L}_{\theta,\,i}\,\phi\left(\frac{\Delta^{R}_{\theta,\,i}}{\Delta^{L}_{\theta,\,i}}\right) + (1-\omega)\,\Delta^{R}_{\theta,\,i}\,\phi\left(\frac{\Delta^{L}_{\theta,\,i}}{\Delta^{R}_{\theta,\,i}}\right)\right] \tag{4.10b}$$

where $\omega \in [-1, 1]$ is a weight value, the slope values $\Delta^{L}_{\theta,\,i}$ and $\Delta^{R}_{\theta,\,i}$ are given by:

$$\Delta^{R}_{\theta,\,i} = q_{\theta,\,i+1} - q_{\theta,\,i}, \qquad \Delta^{L}_{\theta,\,i} = q_{\theta,\,i} - q_{\theta,\,i-1}, \tag{4.11}$$

and $\phi(r)$ is a slope limiter function that ensures the Total Variation Diminishing property of the solution. This property ensures that the solution will not develop spurious oscillations around discontinuities or shocks. In the proposed MHD code, the following slope limiter functions are available:

- Minmod (ROE, 1986):

$$\phi(r) = \max\left[0, \min\left(1, r\right)\right]; \tag{4.12}$$

- Monotonized Central (MC) (van LEER, 1977):

$$\phi(r) = \max\left[0, \min\left(2r, \frac{r+1}{2}, 2\right)\right]; \tag{4.13}$$

- Superbee (ROE, 1986):

$$\phi(r) = \max\left[0, \min\left(1, 2r\right), \min\left(r, 2\right)\right]; \tag{4.14}$$

- van Albada (van ALBADA et al., 1997):

$$\phi(r) = \frac{r^2 + r}{r^2 + 1}; \tag{4.15}$$

- van Leer (van LEER, 1974):

$$\phi(r) = \frac{\|r\| + r}{\|r\| + 1}; \tag{4.16}$$

- Koren (KOREN, 1993):

$$\phi(r) = \max\left[0, \min\left(2r, \frac{2r+1}{3}, 2\right)\right]; \tag{4.17}$$

The reconstructed interface solutions $\mathbf{q}_i^R$ and $\mathbf{q}_{i+1}^L$ from the adjacent cells $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$, replaces the initial condition of the Riemann problem given in Equation 4.9, obtaining:

$$\mathbf{q} = \begin{cases} \mathbf{q}_i^R, & \text{if } x' < 0 \\ \mathbf{q}_{i+1}^L, & \text{if } x' > 0 \end{cases}. \tag{4.18}$$

This Riemann problem has an exact solution calculated through a characteristic variable representation, provided in the next section. However, for practical reasons, the solution of the Riemann problem is approximated by using numerical methods called approximate Riemann solvers. In the proposed MHD solver, the Riemann problems are approximated by the methods presented in the Sections 4.3.2 and 4.3.3.

4.3 Riemann problem

The most common approach to solve the Riemann problem is the method of characteristics. This method is based in a change of variables which transforms the studied problem into an advection equation. This so-called characteristic formulation have an exact solution, which is converted back to the original variables.

To construct the characteristic formulation of the MHD equations, consider its primitive formulation:

$$\frac{\partial \mathbf{w}}{\partial t} + \mathbb{A}_x \frac{\partial \mathbf{w}}{\partial x} = \mathbf{0}, \tag{4.19}$$

where $\mathbf{w} = (\rho, p, \mathbf{u}, \mathbf{B})$ is the vector of the primitive variables and $\mathbb{A}_x$ is the Jacobian matrix of the system. For the ideal MHD problem, these values are presented in Section 3.1.1.

Considering the MHD equations as a hyperbolic system, the matrix $\mathbb{A}_x$ has real eigenvalues and can be diagonalised. Hence, $\mathbb{A}_x = \mathbb{P}_x \mathbb{V}_x \mathbb{P}_x^{-1}$, where $\mathbb{P}_x$ is a matrix which its columns are the eigenvectors of $\mathbb{A}_x$ and $\mathbb{V}_x$ is a diagonal matrix which its

elements are the eigenvalues of $\mathbb{A}_x$ in ascending order. Multiplying $\mathbb{P}_x^{-1}$ in Equation 4.19:

$$\mathbb{P}_x^{-1}\frac{\partial \mathbf{w}}{\partial t} + \mathbb{P}_x^{-1}\mathbb{A}_x\frac{\partial \mathbf{w}}{\partial x} = \mathbf{0}. \tag{4.20}$$

Then, defining the characteristics variables $\mathfrak{v} = \mathbb{P}_x^{-1}\mathbf{w}$:

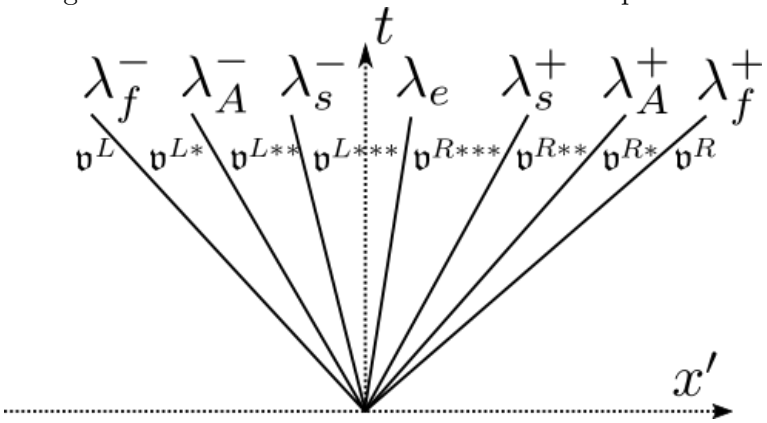
$$\frac{\partial \mathfrak{v}}{\partial t} + \mathbb{P}_x^{-1}\mathbb{A}_x\mathbb{P}_x\frac{\partial \mathfrak{v}}{\partial x} = \mathbf{0}. \tag{4.21}$$

Thus, the characteristic formulation is given as:

$$\frac{\partial \mathfrak{v}}{\partial t} + \mathbb{V}_x\frac{\partial \mathfrak{v}}{\partial x} = \mathbf{0}. \tag{4.22}$$

This formulation implies in a set of advection equations for each characteristic variable v_θ from the vector $\mathfrak{v}$. Therefore, considering a Riemann problem as initial condition, the characteristics v_θ of the discontinuity formed in $x' = 0$ propagates with a velocity correspondent to its eigenvalue $\lambda_\theta = \frac{dx'}{dt}$ over the $x't$ plane, creating the Riemann fan as described in Figure 4.2.

Figure 4.2 - Full Riemann fan for the MHD equations.



Source: Author's production.

The space between each pair of adjacent propagated characteristic defines an intermediary star state. In the case of the MHD equations, there are the states $\mathfrak{v}^{L*}$, $\mathfrak{v}^{L**}$, $\mathfrak{v}^{L***}$, $\mathfrak{v}^{R***}$, $\mathfrak{v}^{R**}$ and $\mathfrak{v}^{R*}$. For each star state, the solution $\mathfrak{v}$ assumes different values.

In order to calculate each variable v_θ from $\mathfrak{v}$, consider that each characteristic λ_θ

separates the solution inside the Riemann fan so that the value v_θ from any arbitrary star state is given as:

$$v_\theta\left(\frac{x'}{t}\right) = \begin{cases} v_\theta^L, \text{if } \frac{x'}{t} < \lambda_\theta \\ v_\theta^R, \text{if } \frac{x'}{t} > \lambda_\theta \end{cases} . \tag{4.23}$$

This implies that the difference between two adjacent states, separated by a characteristic λ_θ, is equal to the correspondent jump value $\Delta\mathfrak{v}_\theta$ where:

$$\begin{aligned} \Delta\mathfrak{v}_1 &= \left[v_1^R - v_1^L, 0, 0, 0, 0, 0, 0\right]^T \\ \Delta\mathfrak{v}_2 &= \left[0, v_2^R - v_2^L, 0, 0, 0, 0, 0\right]^T \\ &\vdots \\ \Delta\mathfrak{v}_7 &= \left[0, 0, 0, 0, 0, 0, v_7^R - v_7^L\right]^T \end{aligned} . \tag{4.24}$$

Therefore, any star state in the left side of the Riemann fan can be calculated by adding to the left state $\mathfrak{v}^L$ the jump conditions required to reach that star state. While the star states in the right side of the Riemann fan can be calculated by subtracting the jump conditions from the right state $\mathfrak{v}^R$. Thus, the exact solution for every point with an inclination $\frac{x'}{t}$ inside the Riemann fan is calculated as:

$$\mathfrak{v}\left(\frac{x'}{t}\right) = \begin{cases} \mathfrak{v}^L, & \text{if } \dfrac{x'}{t} < \lambda_f^- \\ \mathfrak{v}^{L*} = \mathfrak{v}^L + \Delta\mathfrak{v}_1, & \text{if } \lambda_f^- \leq \dfrac{x'}{t} < \lambda_A^- \\ \mathfrak{v}^{L**} = \mathfrak{v}^L + \Delta\mathfrak{v}_1 + \Delta\mathfrak{v}_2, & \text{if } \lambda_A^- < \dfrac{x'}{t} < \lambda_s^- \\ \mathfrak{v}^{L***} = \mathfrak{v}^L + \Delta\mathfrak{v}_1 + \Delta\mathfrak{v}_2 + \Delta\mathfrak{v}_3, & \text{if } \lambda_s^- < \dfrac{x'}{t} < \lambda_e \\ \mathfrak{v}^{R***} = \mathfrak{v}^R - \Delta\mathfrak{v}_7 - \Delta\mathfrak{v}_6 - \Delta\mathfrak{v}_5, & \text{if } \lambda_e < \dfrac{x'}{t} < \lambda_s^+ \\ \mathfrak{v}^{R**} = \mathfrak{v}^R - \Delta\mathfrak{v}_7 - \Delta\mathfrak{v}_6, & \text{if } \lambda_s^+ < \dfrac{x'}{t} < \lambda_A^+ \\ \mathfrak{v}^{R*} = \mathfrak{v}^R - \Delta\mathfrak{v}_7, & \text{if } \lambda_A^+ < \dfrac{x'}{t} < \lambda_f^+ \\ \mathfrak{v}^R, & \text{if } \dfrac{x'}{t} > \lambda_f^+ \end{cases} \tag{4.25}$$

Multiplying this solution by $\mathbb{P}_x$, then using the definition $\mathbb{P}_x\mathfrak{v} = \mathbf{w}$ and the relation $\mathbb{P}_x\Delta\mathfrak{v}_\theta = \mathfrak{R}_\theta\Delta v_\theta$, where $\mathfrak{R}_\theta$ is the eigenvector associated with the eigenvalue λ_θ and $\Delta v_\theta = v_\theta^R - v_\theta^L$. Thus, the exact solution of the Riemann problem is described in the

primitive variables as:

$$\mathbf{w}\left(\frac{x'}{t}\right) = \begin{cases} \mathbf{w}^{L}, & \text{if } \dfrac{x'}{t} < \lambda_f^- \\ \mathbf{w}^{L*} = \mathbf{w}^{L} + \mathfrak{R}_1 \Delta v_1, & \text{if } \lambda_f^- \leq \dfrac{x'}{t} < \lambda_A^- \\ \mathbf{w}^{L**} = \mathbf{w}^{L} + \mathfrak{R}_1 \Delta v_1 + \mathfrak{R}_2 \Delta v_2, & \text{if } \lambda_A^- < \dfrac{x'}{t} < \lambda_s^- \\ \mathbf{w}^{L***} = \mathbf{w}^{L} + \mathfrak{R}_1 \Delta v_1 + \mathfrak{R}_2 \Delta v_2 + \mathfrak{R}_3 \Delta v_3, & \text{if } \lambda_s^- < \dfrac{x'}{t} < \lambda_e \\ \mathbf{w}^{R***} = \mathbf{w}^{R} - \mathfrak{R}_7 \Delta v_7 - \mathfrak{R}_6 \Delta v_6 - \mathfrak{R}_5 \Delta v_5, & \text{if } \lambda_e < \dfrac{x'}{t} < \lambda_s^+ \\ \mathbf{w}^{R**} = \mathbf{w}^{R} - \mathfrak{R}_7 \Delta v_7 - \mathfrak{R}_6 \Delta v_6, & \text{if } \lambda_s^+ < \dfrac{x'}{t} < \lambda_A^+ \\ \mathbf{w}^{R*} = \mathbf{w}^{R} - \mathfrak{R}_7 \Delta v_7, & \text{if } \lambda_A^+ < \dfrac{x'}{t} < \lambda_f^+ \\ \mathbf{w}^{R}, & \text{if } \dfrac{x'}{t} > \lambda_f^+ \end{cases} \tag{4.26}$$

Obtained the solution for the Riemann problem, the flux inside every state is calculated by multiplying the primitive solution by the Jacobian matrix $\mathbb{A}_x$ and using the relations $\mathbb{A}_x \dfrac{\partial \mathbf{w}}{\partial x} = \dfrac{\partial \mathbf{F}(\mathfrak{q})}{\partial x}$ and $\mathbb{A}_x \mathfrak{R}_\theta = \lambda_\theta \mathfrak{R}_\theta$. Thus, the flux in the interface $x' = 0$ is obtained as:

$$\mathbb{A}_x \mathbf{w}(0,t) = \mathbf{F}[\mathfrak{q}(0,t)] = \mathcal{F}_{\text{exact}}, \tag{4.27}$$

where

$$\mathcal{F}_{\text{exact}}\left(\mathbf{q}^L, \mathbf{q}^R\right) = \begin{cases} \mathbf{F}\left(\mathbf{q}^L\right), & \text{if } 0 < \lambda_f^- \\ \mathbf{F}^{L*} = \mathbf{F}\left(\mathbf{q}^L\right) + \mathfrak{R}_1 \lambda_1 \Delta v_1, & \text{if } \lambda_f^- < 0 < \lambda_A^- \\ \mathbf{F}^{L**} = \mathbf{F}\left(\mathbf{q}^L\right) + \mathfrak{R}_1 \lambda_1 \Delta v_1 + \mathfrak{R}_2 \lambda_2 \Delta v_2, & \text{if } \lambda_A^- < 0 < \lambda_s^- \\ \mathbf{F}^{L***} = \mathbf{F}\left(\mathbf{q}^L\right) + \mathfrak{R}_1 \lambda_1 \Delta v_1 + \mathfrak{R}_2 \lambda_2 \Delta v_2 + \mathfrak{R}_3 \lambda_3 \Delta v_3, & \text{if } \lambda_s^- < 0 < \lambda_e \\ \mathbf{F}^{R***} = \mathbf{F}\left(\mathbf{q}^R\right) - \mathfrak{R}_7 \lambda_7 \Delta v_7 - \mathfrak{R}_6 \lambda_6 \Delta v_6 - \mathfrak{R}_5 \lambda_5 \Delta v_5, & \text{if } \lambda_e < 0 < \lambda_s^+ \\ \mathbf{F}^{R**} = \mathbf{F}\left(\mathbf{q}^R\right) - \mathfrak{R}_7 \lambda_7 \Delta v_7 - \mathfrak{R}_6 \lambda_6 \Delta v_6, & \text{if } \lambda_s^+ < 0 < \lambda_A^+ \\ \mathbf{F}^{R*} = \mathbf{F}\left(\mathbf{q}^R\right) - \mathfrak{R}_7 \lambda_7 \Delta v_7, & \text{if } \lambda_A^+ < 0 < \lambda_f^+ \\ \mathbf{F}\left(\mathbf{q}^R\right), & \text{if } \lambda_f^+ < 0 \end{cases} \tag{4.28}$$

Besides the Riemann problem having a known exact solution, its usage in real applications are impracticable due to the high computational cost required for the eigenvector computations, which are associated with the usage of iterative methods.

For ideal MHD, an exact solver is applied in Takahashi and Yamada (2014), and in Giacomazzo and Rezzolla (2006) an exact solver is also proposed for relativistic MHD.

The alternative to solve the Riemann problem under a reasonable computational time is the usage of the approximated Riemann solvers. One of the most popular approaches is the Roe solver (ROE, 1981), which is applied for MHD equations in the works of Brio and Wu (1988). This approach replaces the Jacobian matrix $\mathbb{A}_x$ for an approximated constant coefficient linear matrix $\bar{\mathbb{A}}_x\left(\mathbf{q}^L, \mathbf{q}^R\right)$. Then, the Riemann problem is solved analogously using this linearised Jacobian matrix, resulting in a problem with simpler eigenvectors.

Another class of successful approximated Riemann solvers are the HLL-type Riemann solvers (HARTEN et al., 1983). In the proposed MHD code, described in Chapter 7, are implemented the HLL and HLLD Riemann solvers described in Sections 4.3.2 and 4.3.3, respectively. The choice for the HLL-type over the Roe-type solvers in this work is due to the its robustness and lower computational cost, since the HLL-type does not require eigenvector computations.

The construction of the HLL-type solvers is based on the approximation of the jump conditions. This is done through the Rankine–Hugoniot relations, derived in the next section.

4.3.1 Rankine–Hugoniot relations

The Rankine–Hugoniot relations associates the solutions of two adjacent states separated by a shock wave and their respective fluxes. To obtain these relations, the 1D Ideal MHD equations in its conservative formulation are considered:

$$\frac{\partial \mathfrak{q}}{\partial t} = -\frac{\partial \mathbf{F}\left(\mathfrak{q}\right)}{\partial x}, \tag{4.29}$$

integrating over the domain $[x',\ x'+dx'] \times [t^n,\ t^n+dt]$:

$$\int_{x'}^{x'+dx'} \int_{t^n}^{t^n+dt} \frac{\partial \mathfrak{q}}{\partial t} dt dx = -\int_{x'}^{x'+dx'} \int_{t^n}^{t^n+dt} \frac{\partial \mathbf{F}\left(\mathfrak{q}\right)}{\partial x} dt dx. \tag{4.30}$$

These integrals can be solved using the Leibniz's rule for t in the left side and for x in the right side term, obtaining:

$$\int_{x'}^{x'+dx'} \left[\mathfrak{q}\left(x, t^n + dt\right) - \mathfrak{q}\left(x, t^n\right)\right] dx = -\int_{t^n}^{t^n+dt} \left[\mathbf{F}\left(\mathfrak{q}(x' + dx', t)\right) - \mathbf{F}\left(\mathfrak{q}(x', t)\right)\right] dt. \tag{4.31}$$

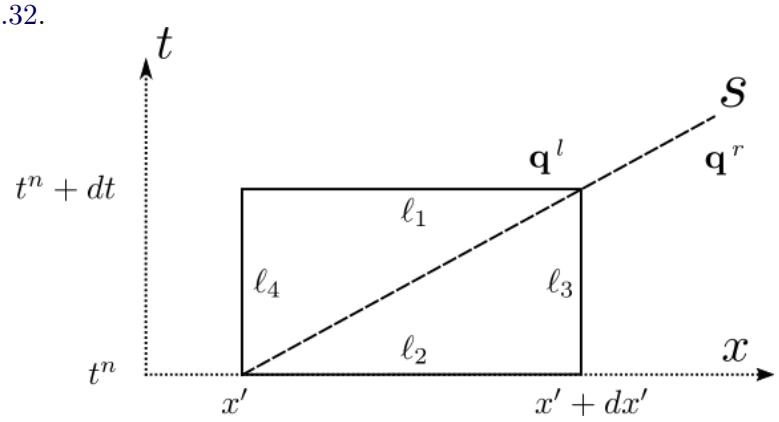
These integrals can be splitted into:

$$\begin{aligned} \underbrace{\int_{x'}^{x'+dx'} \mathfrak{q}\left(x, t^n + dt\right) dx}_{\ell_1} - \underbrace{\int_{x'}^{x'+dx'} \mathfrak{q}\left(x, t^n\right) dx}_{\ell_2} = &- \underbrace{\int_{t^n}^{t^n+dt} \mathbf{F}\left(\mathfrak{q}(x' + dx', t)\right) dt}_{\ell_3} \\ &+ \underbrace{\int_{t^n}^{t^n+dt} \mathbf{F}\left(\mathfrak{q}(x', t)\right) dt}_{\ell_4}. \end{aligned} \tag{4.32}$$

Considering the schematic presented in Figure 4.3, the integration interval of each integral presented in Equation 4.32 corresponds to a side of the rectangle over the $x't$ plane. Taking into account that any of them crosses the slope s, they are all integrations over constant functions. Therefore, the exact solution of these integrals can be obtained through the mean value theorem:

$$\left(\mathbf{q}^{\,l} - \mathbf{q}^{\,r}\right) dx' = -\left[\mathbf{F}\left(\mathbf{q}^{\,r}\right) - \mathbf{F}\left(\mathbf{q}^{\,l}\right)\right] dt \tag{4.33}$$

Figure 4.3 - Example of a slope s dividing two arbitrary states $\mathbf{q}^{\,l}$ and $\mathbf{q}^{\,r}$. The sides ℓ_1, ℓ_2, ℓ_3 and ℓ_4 correspond to the integration intervals of the terms in Equation 4.32.



Source: Author's production.

Considering $s = \dfrac{dx'}{dt}$ as a slope value that approximates the characteristic velocity,

the Rankine–Hugoniot relations are obtained as:

$$s\left(\mathbf{q}^r - \mathbf{q}^l\right) = \mathbf{F}\left(\mathbf{q}^r\right) - \mathbf{F}\left(\mathbf{q}^l\right) \tag{4.34}$$

This relation is applied in HLL-type approximate Riemann solvers as the jump conditions to calculate the next most internal star states using the data known from the external states. Then, the next more internal star states are computed recursively.

4.3.2 HLL Riemann solver

The HLL-type (Harten-Lax-van Leer) is a class of approximate Riemann solvers which simplifies the Riemann fan by considering only some of the characteristic waves. These waves are approximated by slope values, then the intermediary star states and their respective fluxes are obtained through the Rankine–Hugoniot relations.

The simplest case, the HLL Riemann solver, as proposed in Harten et al. (1983), is constructed under the assumption of a constant state $\mathbf{q}^*$ that covers from the slowest to the fastest wave, λ_f^- and λ_f^+ respectively, forming the Riemann fan presented in Figure 4.4. Therefore, the states of the solution $\mathbf{q}$ over the plane $x't$ are described as:

$$\mathbf{q}\left(x', t\right) = \begin{cases} \mathbf{q}^L, \text{if } \frac{x'}{t} < s^L \\ \mathbf{q}^*, \text{if } s^L \leq \frac{x'}{t} \leq s^R \\ \mathbf{q}^R, \text{if } \frac{x'}{t} > s^R \end{cases} \tag{4.35}$$

where s^L and s^R are approximations of the wave velocities λ_f^- and λ_f^+, respectively. In Davis (1988), the following values are suggested:

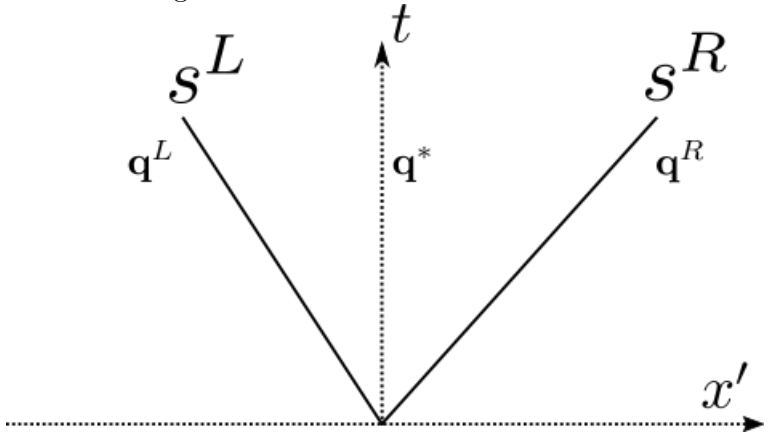
$$s^L = \min\left(u_x^L - c_f^L,\ u_x^R - c_f^R\right) \qquad s^R = \max\left(u_x^L + c_f^L,\ u_x^R + c_f^R\right) \tag{4.36}$$

where c_f^L and c_f^R are the fast magnetosonic velocities, as defined in Equation 3.11a in Section 3.1.1, for the left and right states respectively.

Using the Rankine–Hugoniot relations given in Equation 4.34 to describe the jump conditions over the waves s^L and s^R, the following system is obtained:

$$\begin{cases} s^L\left(\mathbf{q}^* - \mathbf{q}^L\right) = \mathbf{F}^*\left(\mathbf{q}^L, \mathbf{q}^R\right) - \mathbf{F}\left(\mathbf{q}^L\right) \\ s^R\left(\mathbf{q}^R - \mathbf{q}^*\right) = \mathbf{F}\left(\mathbf{q}^R\right) - \mathbf{F}^*\left(\mathbf{q}^L, \mathbf{q}^R\right) \end{cases} \tag{4.37}$$

Figure 4.4 - HLL Flux Riemann's fan



Source: Author's production.

where $\mathbf{F}^*\left(\mathbf{q}^L, \mathbf{q}^R\right)$ is the value of the flux inside the $\mathbf{q}^*$ region. The solution for this system in the variables $\mathbf{q}^*$ and $\mathbf{F}^*\left(\mathbf{q}^L, \mathbf{q}^R\right)$ is given by:

$$\mathbf{q}^* = \frac{s^R\mathbf{q}^R - s^L\mathbf{q}^L + \mathbf{F}\left(\mathbf{q}^L\right) - \mathbf{F}\left(\mathbf{q}^R\right)}{s^R - s^L}, \tag{4.38a}$$

$$\mathcal{F}^*\left(\mathbf{q}^L, \mathbf{q}^R\right) = \frac{s^R\mathbf{F}\left(\mathbf{q}^L\right) - s^L\mathbf{F}\left(\mathbf{q}^R\right) + s^L s^R\left(\mathbf{q}^R - \mathbf{q}^L\right)}{s^R - s^L}. \tag{4.38b}$$

Therefore, the HLL flux approximates the solution for the Riemann problem as:

$$\mathcal{F}_{\text{HLL}}\left(\mathbf{q}^L, \mathbf{q}^R\right) = \begin{cases} \mathbf{F}\left(\mathbf{q}^L\right), & \text{if } s^L > 0 \\ \mathcal{F}^*\left(\mathbf{q}^L, \mathbf{q}^R\right), & \text{if } s^L \leq 0 \leq s^R \\ \mathbf{F}\left(\mathbf{q}^R\right), & \text{if } s^R \leq 0 \end{cases}. \tag{4.39}$$

This can be simplified by:

$$\mathcal{F}_{\text{HLL}}\left(\mathbf{q}^L, \mathbf{q}^R\right) = \frac{\bar{s}^R\mathbf{F}\left(\mathbf{q}^L\right) - \bar{s}^L\mathbf{F}\left(\mathbf{q}^R\right) + \bar{s}^L\bar{s}^R\left(\mathbf{q}^R - \mathbf{q}^L\right)}{\bar{s}^R - \bar{s}^L}, \tag{4.40}$$

where the new slopes $\bar{s}^L$ and $\bar{s}^R$ are:

$$\bar{s}^L = \min\left(s^L,\ 0\right) \qquad \bar{s}^R = \max\left(s^R,\ 0\right). \tag{4.41}$$

Despite being a simple methodology, the HLL flux is not suitable to solve problems with isolated discontinuities, presenting a significant dissipation in those cases.

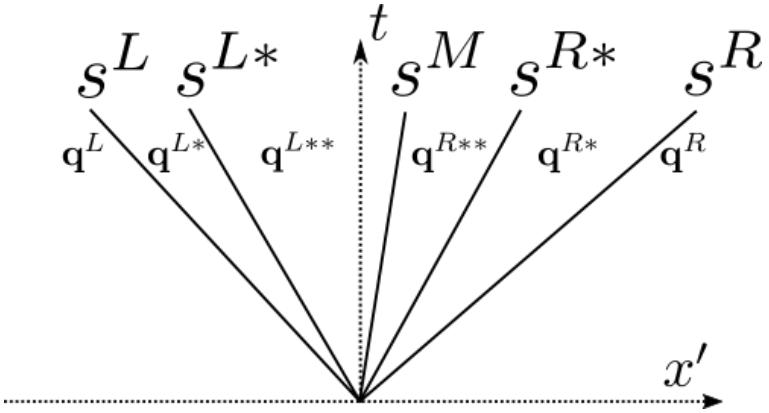
This effect is caused because all the intermediary states of the MHD problem are approximated as only one state.

4.3.3 HLLD Riemann solver

The HLLD (Harten-Lax-van Leer Discontinuities) Riemann solver is proposed in Miyoshi and Kusano (2005) as an extension of the HLL solver based on the multi-state HLLC-type solvers (TORO, 1999). This method splits the Riemann fan into 4 intermediary states $\mathbf{q}^{L*}$, $\mathbf{q}^{L**}$, $\mathbf{q}^{R**}$ and $\mathbf{q}^{R*}$ obtained between the 5 waves s^L, s^{L*}, s^M, s^{R*} and s^R. Therefore, as illustrated in Figure 4.5, the solution inside the Riemann fan are described as:

$$\mathbf{q}\left(x',t\right) = \begin{cases} \mathbf{q}^{L}, & \text{if } \frac{x'}{t} < s^{L} \\ \mathbf{q}^{L*}, & \text{if } s^{L} < \frac{x'}{t} < s^{L*} \\ \mathbf{q}^{L**}, & \text{if } s^{L*} < \frac{x'}{t} < s^{M} \\ \mathbf{q}^{R**}, & \text{if } s^{M} < \frac{x'}{t} < s^{R*} \\ \mathbf{q}^{R*}, & \text{if } s^{R*} < \frac{x'}{t} < s^{R} \\ \mathbf{q}^{R}, & \text{if } 0 > s^{R} \end{cases} \tag{4.42}$$

Figure 4.5 - Schematic structure of the HLLD solver Riemann's fan.



Source: Author's production.

where the slopes s^L and s^R approximates the fast magnetosonic waves λ_f^- and λ_f^+,

respectively, using the following relations:

$$\begin{aligned} s^L &= \min\left[\min\left(u_x^L, u_x^R\right) - \max\left(c_f^L, c_f^R\right)\ , 0\right] \\ s^R &= \max\left[\max\left(u_x^L, u_x^R\right) + \max\left(c_f^L, c_f^R\right)\ , 0\right] \end{aligned} \ . \tag{4.43}$$

The middle slope s^M approximates the entropy wave λ_e by:

$$s^M = \frac{\left(s^R - u_x^R\right)\rho^R u_x^R - \left(s^L - u_x^L\right)\rho^L u_x^L - p^R + p^L}{\left(s^R - u_x^R\right)\rho^R - \left(s^L - u_x^L\right)\rho^L}. \tag{4.44}$$

Lastly, the waves s^{L*} and s^{R*}, associated with the Alfvén waves λ_A^- and λ_A^+ respectively, are approximated by:

$$s^{L*} = s^M - \frac{|B_m|}{\sqrt{\rho^L \frac{s^L - u_x^L}{s^L - s^M}}} \qquad s^{R*} = s^M + \frac{|B_m|}{\sqrt{\rho^R \frac{s^R - u_x^R}{s^R - s^M}}}, \tag{4.45}$$

where $B_m = \frac{1}{2}\left(B_x^L + B_x^R\right)$ is an average value of the component of the magnetic field which is parallel to the flux. In the HLLD solver, this component is assumed to be constant through the intermediary states of the Riemann's fan. Furthermore, the velocity component parallel to the flux, u_x in this case, and the total pressure are also considered constant inside the Riemann's fan. These values are given respectively by s^M and:

$$p_T^* = \frac{\left(s^R - u_x^R\right)\rho^R p_T^L - \left(s^L - u_x^L\right)\rho^L p_T^R + \rho^L\rho^R\left(s^R - u_x^R\right)\left(s^L - u_x^L\right)\left(u_x^R - u_x^L\right)}{\left(s^R - u_x^R\right)\rho^R - \left(s^L - u_x^L\right)\rho^L}. \tag{4.46}$$

where the total pressure p_T stands for the sum of both gas and magnetic pressure as:

$$p_T = p + \frac{1}{2}\|\mathbf{B}\|^2. \tag{4.47}$$

Applying the slopes s^L and s^R into the Rankine-Hugoniot condition given in Equation 4.34, the intermediary states $\mathbf{q}^{L*}$ and $\mathbf{q}^{R*}$ are obtained for the values of a as L

or R, respectively.

$$\mathbf{q}^{a*} = \begin{bmatrix} \rho^{a*} \\ \rho^{a*}u_x^{a*} \\ \rho^{a*}u_y^{a*} \\ \rho^{a*}u_z^{a*} \\ E^{a*} \\ B_x^{a*} \\ B_y^{a*} \\ B_z^{a*} \end{bmatrix} = \begin{bmatrix} \rho^a \frac{s^a - u_x^a}{s^a - s^M} \\ \rho^{a*} s^M \\ \rho^{a*}\left(u_y^a - B_y^a \chi_1 \chi_2\right) \\ \rho^{a*}\left(u_z^a - B_z^a \chi_1 \chi_2\right) \\ \frac{(s^a - u_x^a)E^a - p_T^a u_x^a + p_T^* s^M + B_m \chi_4}{s^a - s^M} \\ B_m \\ B_y^a \chi_1 \chi_3 \\ B_z^a \chi_1 \chi_3 \end{bmatrix} \tag{4.48}$$

where the auxiliary variables χ are given by:

$$\chi_1 = \left[\rho^a \left(s^a - u_x^a\right)\left(s^a - s^M\right) - B_m^2\right]^{-1} \tag{4.49a}$$

$$\chi_2 = B_m \left(s^M - u_x^a\right) \tag{4.49b}$$

$$\chi_3 = \rho^a \left(s^a - u_x^a\right)^2 - B_m^2 \tag{4.49c}$$

$$\chi_4 = \left(s^M B_m + u_y^a B_y^a + u_z^a B_z^a - \mathbf{u}^{a*} \cdot \mathbf{B}^{a*}\right) \tag{4.49d}$$

Analysing those intermediary states formulation, it is noted that the energy term requires the computation of the other variables beforehand. Further analysis also shows that there is a combination of values that may lead to a division by zero in χ_1. Also, when $s^M = u_x^a$, $s^a = u_x^a \pm c_f^a$, $B_y^a = B_z^a = 0$ and $B_m^2 \geq \gamma p^a$, the problem may produces complex magnetosonic velocities. In those cases, the intermediary states $\mathbf{q}^{a*}$ are equal $\mathbf{q}^a$.

To obtain the remaining intermediary states $\mathbf{q}^{a**}$, are assumed that $\rho^{a**} = \rho^{a*}$. Applying the slopes s^{L*} and s^{R*} into the Rankine-Hugoniot condition, these states

are obtained as:

$$\mathbf{q}^{a**} = \begin{bmatrix} \rho^{a**} \\ \rho^{a**}u_x^{a**} \\ \rho^{a**}u_y^{a**} \\ \rho^{a**}u_z^{a**} \\ E^{a**} \\ B_x^{a**} \\ B_y^{a**} \\ B_z^{a**} \end{bmatrix} = \begin{bmatrix} \rho^{a*} \\ \rho^{a*}s^M \\ \rho^{a*}\chi_5\left[u_y^{L*}\sqrt{\rho^{L*}} + u_y^{R*}\sqrt{\rho^{R*}} + \left(B_y^{R*} - B_y^{L*}\right)\mathrm{sign}(B_m)\right] \\ \rho^{a*}\chi_5\left[u_z^{L*}\sqrt{\rho^{L*}} + u_z^{R*}\sqrt{\rho^{R*}} + \left(B_z^{R*} - B_z^{L*}\right)\mathrm{sign}(B_m)\right] \\ E^{a*} \mp \sqrt{\rho^{a*}}\left(\mathbf{u}^{a*}\cdot\mathbf{B}^{a*} - \mathbf{u}^{a**}\cdot\mathbf{B}^{a**}\right)\mathrm{sign}(B_m) \\ B_m \\ \chi_5\left[B_y^{R*}\sqrt{\rho^{L*}} + B_y^{L*}\sqrt{\rho^{R*}} + \sqrt{\rho^{L*}\rho^{R*}}\left(u_y^{R*} - u_y^{L*}\right)\mathrm{sign}(B_m)\right] \\ \chi_5\left[B_z^{R*}\sqrt{\rho^{L*}} + B_z^{L*}\sqrt{\rho^{R*}} + \sqrt{\rho^{L*}\rho^{R*}}\left(u_z^{R*} - u_z^{L*}\right)\mathrm{sign}(B_m)\right] \end{bmatrix} \tag{4.50}$$

where $\mp$ signal corresponds to minus or plus signals depending of the values $a = L$ or R, respectively. The auxiliary variable χ_5 is given by:

$$\chi_5 = \left(\sqrt{\rho^{L*}} + \sqrt{\rho^{R*}}\right)^{-1}. \tag{4.51}$$

Considering that the density ρ is always greater than zero, χ_5 can not be a value involving a division by zero. However, the function $\mathrm{sign}(B_m)$ is not defined when $B_m = 0$. In this case, according to Equation 4.45, both the slopes s^{L*} and s^{R*} assumes the value s^M, resulting in a two-state solver with the slopes s^L, s^M and s^R. Therefore, the solution inside intermediary states $\mathbf{q}^{a**}$ are considered equal to its correspondent states $\mathbf{q}^{a*}$.

Obtained the slopes and the intermediary states, the numerical flux $\mathcal{F}_{\mathrm{HLLD}}\left(\mathbf{q}^L, \mathbf{q}^R\right)$ approximates the solution of the Riemann problem in Equation 4.28 as:

$$\mathcal{F}_{\mathrm{HLLD}}\left(\mathbf{q}^L, \mathbf{q}^R\right) = \begin{cases} \mathbf{F}\left(\mathbf{q}^L\right), & \text{if } s^L > 0 \\ \mathbf{F}\left(\mathbf{q}^L\right) + s^L\left(\mathbf{q}^{L*} - \mathbf{q}^L\right), & \text{if } s^L \leq 0 \leq s^{L*} \\ \mathbf{F}\left(\mathbf{q}^L\right) + s^{L*}\mathbf{q}^{L**} - \left(s^{L*} - s^L\right)\mathbf{q}^{L*} - s^L\mathbf{q}^L, & \text{if } s^{L*} \leq 0 \leq s^M \\ \mathbf{F}\left(\mathbf{q}^R\right) + s^{R*}\mathbf{q}^{R**} - \left(s^{R*} - s^R\right)\mathbf{q}^{R*} - s^R\mathbf{q}^R, & \text{if } s^M \leq 0 \leq s^{R*} \\ \mathbf{F}\left(\mathbf{q}^R\right) + s^R\left(\mathbf{q}^{R*} - \mathbf{q}^R\right), & \text{if } s^{R*} \leq 0 \leq s^R \\ \mathbf{F}\left(\mathbf{q}^R\right), & \text{if } s^R < 0 \end{cases} \tag{4.52}$$

4.4 Runge-Kutta methods

The Runge–Kutta (RK) methods are a class of numerical methods for solving ODE's in the form $\frac{d\mathbf{q}}{dt} = \mathbf{f}(\mathbf{q}, t)$, where $\mathbf{f}$ is a continuous function inside the integrated domain. Those methods are characterised by not requiring the computation of derivatives, neither storing solutions from previous iterations. Instead, the RK methods requires the current solution and evaluations of $\mathbf{f}(\mathbf{q}, t)$.

The number of computations of $\mathbf{f}$ required for every time evolution iteration using a RK method is called the number of stages. In general, a RK method with s stages is written in the form:

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{4.53}$$

where

$$\mathbf{k}_i = \Delta t \mathbf{f}\left(t^n + c_i \Delta t,\ \mathbf{q}^n + \sum_{j=1}^{i-1} a_{i,j}\mathbf{k}_j,\right). \tag{4.54}$$

The constants a, b and c characterises each RK method. The choice of these constants are done so that Equation 4.53 reconstructs the Taylor series expansion of $\mathbf{q}^n$ to $\mathbf{q}^{n+1}$ until the term of $O\left(\Delta t^{p+1}\right)$, where p is the convergence order obtained by the RK method, that is:

$$\Bigg\| \underbrace{\mathbf{q}^n + \sum_{i=1}^{s} b_i \mathbf{k}_i}_{\text{RK evolution}} \underbrace{-\mathbf{q}^n - \sum_{i=1}^{\infty} \frac{\Delta t^i}{i!} \frac{d^i \mathbf{q}^n}{dt^i}}_{\text{Taylor series expansion}} \Bigg\| = O(\Delta t^{p+1}). \tag{4.55}$$

The convergence order obtainable with a RK method is associated with an appropriate choice of constants and the number of stages, where higher convergence order requires more stages to be obtained. In Burden and Faires (1989) is presented the best convergence order obtainable for a s stages RK method, These results are shown in Table 4.1.

Table 4.1 - Best convergence order obtainable with a s stages RK method.

Number of stages (s)	1	2	3	4	$5 \leq s \leq 7$	$8 \leq s \leq 9$	$10 \leq s$
Convergence order	1	2	3	4	$s-1$	$s-2$	$s-3$

Source: Burden and Faires (1989).

In the MHD solver proposed in Chapter 7, the RK methods are applied to perform

the time evolution of every cell of the FV discretisation given in Equation 4.6. Thus:

$$\mathbf{f}_{i,j,k} = \frac{\mathcal{F}^{-}_{i,j,k} - \mathcal{F}^{+}_{i,j,k}}{\Delta x} + \frac{\mathcal{G}^{-}_{i,j,k} - \mathcal{G}^{+}_{i,j,k}}{\Delta y} + \frac{\mathcal{H}^{-}_{i,j,k} - \mathcal{H}^{+}_{i,j,k}}{\Delta z} + \mathbf{S}\left(\mathbf{q}_{i,j,k}\right). \tag{4.56}$$

In particular, the solver implemented in this work uses the two stages, second order, RK method defined by the constants:

$$b_1 = \frac{1}{2}, \qquad b_2 = \frac{1}{2}, \qquad c_2 = 1, \qquad a_{2,1} = 1. \tag{4.57}$$

This RK method is written in the formulation given in Equation 4.53 as:

$$\mathbf{q}^{n+1} = \mathbf{q}^{n} + \frac{1}{2}\mathbf{f}\left(\mathbf{q}^{n}, t^{n}\right) + \frac{1}{2}\mathbf{f}\left(\mathbf{q}^{n} + \mathbf{f}\left(\mathbf{q}^{n}, t^{n}\right), t^{n} + \Delta t\right), \tag{4.58}$$

which can be written in the two-step compact formulation:

$$\begin{aligned} \mathbf{q}^{\text{aux}} &= \mathbf{q}^{n} + \mathbf{f}(\mathbf{q}^{n}, t) \\ \mathbf{q}^{n+1} &= \tfrac{1}{2}\left[\mathbf{q}^{n} + \mathbf{q}^{\text{aux}} + \mathbf{f}(\mathbf{q}^{\text{aux}}, t + \Delta t)\right] \end{aligned} \tag{4.59}$$

4.5 Courant-Friedrich-Lewy Condition

The Courant-Friedrich-Lewy condition (CFL) is a constraint required in order to ensure the convergence and stability of the numerical scheme. The idea is to obtain a relation among the mesh spacing Δx, Δy and Δz, the characteristics of the problem and the time step so that the fastest characteristic do not cross an entire cell during the time step.

This relation is useful to obtain a safe time step for the next time evolution Δt^{n+1} as:

$$\Delta t^{n+1} = \sigma\left[\max\left(s_{\text{adv}}, s_{\text{dif}}\right)\right]^{-1} \tag{4.60}$$

where the dimensionless parameter $\sigma \in (0, 1)$ is a predefined value called Courant number, s_{adv} and s_{dif} are values associated with the fastest slopes due to advective and diffusive effects, respectively. Considering the primitive formulation of the MHD model, the advective effects are determined by the matrices $\mathbb{A}_x$, $\mathbb{A}_y$ and $\mathbb{A}_z$, which are predominant in Ideal MHD. The advective effects are computed as:

$$s_{\text{adv}} = \frac{\max_{\forall \mathcal{C} \in \Omega}\left(\lambda_x\right)}{\Delta x} + \frac{\max_{\forall \mathcal{C} \in \Omega}\left(\lambda_y\right)}{\Delta y} + \frac{\max_{\forall \mathcal{C} \in \Omega}\left(\lambda_z\right)}{\Delta z} \tag{4.61}$$

where λ_x is the highest slope, in absolute values, among the characteristics approximated for a Riemann problem solved in the x direction. Thus, for the MHD equations:

$$\lambda_x = \max_{\forall \mathcal{C} \in \Omega} \left[|s^L|, |s^R| \right]. \tag{4.62}$$

The values λ_y and λ_z have analogous definitions for the y and z axis, respectively.

In contrast, the diffusive effects are associated with the matrices coupled with second derivative terms, such as A_{xx}, A_{yy} or A_{zz}. These terms do not appear in the Ideal MHD model, but occurs in formulations such as the Resistive MHD model. In this case, the diffusive effects are determined by:

$$s_{\text{dif}} = \frac{\Delta x^2 \Delta y^2 \Delta z^2}{2 \max_{\forall \mathcal{C} \in \Omega} (\eta) \left(\Delta x^2 + \Delta y^2 + \Delta z^2 \right)}, \tag{4.63}$$

where $\max_{\forall \mathcal{C} \in \Omega} (\eta)$ is the highest value inside the domain for the resistivity.

5 DIVERGENCE CLEANING

In certain numerical methods, such as the FV, the simulations of the MHD equations produces numerical errors in the magnetic field that violates the Gauss' law for magnetism $\nabla \cdot \mathbf{B} = 0$. Physically, as discussed in Brackbill and Barnes (1980), these errors appear as a force parallel to the magnetic field, which creates magnetic monopoles. This effect may compromises the topology of the magnetic field lines, leading to plasma transport orthogonal to the magnetic field. Moreover, this non-physical behaviour can lead to numerical instabilities (DEDNER et al., 2002; TÓTH, 2000).

The origin of these effects can be illustrated by considering the divergence of the Induction equation (Equation 3.2d):

$$\frac{\partial \left(\nabla \cdot \mathbf{B}\right)}{\partial t} + \nabla \cdot \left[\nabla \times \left(\mathbf{B} \times \mathbf{u}\right)\right] = 0. \tag{5.1}$$

Mathematically, due to the identity $\nabla \cdot (\nabla \times \cdot) = 0$, the term $\nabla \cdot \mathbf{B}$ is expected to remain constant during the entire simulation. However, this identity is not numerically valid, leading to the generation of $\nabla \cdot \mathbf{B}$ components after every time step, causing the loss of the $\nabla \cdot \mathbf{B} = 0$ constraint.

The influence of these components over the physics of the problem are measured with several approaches. In particular, Hopkins (2016) define a parameter given by:

$$\mathcal{D}_{\mathbf{B}} = \min\left(\Delta x, \Delta y, \Delta z\right) \frac{|\nabla \cdot \mathbf{B}|}{\|\mathbf{B}\|} \tag{5.2}$$

If the parameter $\mathcal{D}_{\mathbf{B}}$ is greater than 1, the divergence over $\mathbf{B}$ is considered to compromise the solution. This methodology is used in Hopkins (2016) to measure and ensure the quality of the solution for several divergence cleaning approaches.

These approaches are numerical strategies applied alongside with the time evolution process in order to avoid these divergence errors. One of the first approaches was proposed in Brackbill and Barnes (1980), where a Poisson equation is associated with the divergence of the magnetic field in order to obtain the divergence components and then, subtract from $\mathbf{B}$. This approach is unfeasible for many applications due to the high computational costs associated with the implicit methods required to solve the Poisson equation.

Another well known approach is the parabolic-hyperbolic approach based in Gener-

alised Lagrange Multipliers proposed in Dedner et al. (2002). This approach couples a new equation and a new variable to the system. These new elements are responsible for transporting and diffusing the numerical errors associated to the formation of the magnetic monopoles. This approach did not remove the components which causes divergence in $\mathbf{B}$ completely, but it removes the enough to sustain the physics and the stability for many applications at a very low computational cost.

Other popular divergence cleaning methods are the 8-wave (POWELL, 1997) and constrained transport (EVANS; HAWLEY, 1988). In Miyoshi and Kusano (2011) is performed a comparative study of those methods.

5.1 Generalised Lagrange Multipliers approach

One of the most successful approaches for the divergence control are the Generalised Lagrangian Multiplier (GLM) based divergence cleaning corrections. These approaches are based in the strategy introduced in Assous et al. (1993) for Maxwell's Equations. In Dedner et al. (2002) this formulation is extended to MHD problems based in the formulation of the Lagrange multiplier presented in Munz et al. (2000).

Generalised Lagrange Multipliers are a class of methods for maximising or minimising function under some constraints. In the context of divergence cleaning methods, the usage of the GLM methods consists in maximising the Induction equation, while imposing the constraint $\nabla \cdot \mathbf{B} = 0$. For that, a scalar field is coupled to the Gauss's Law:

$$\mathcal{D}(\psi) + \nabla \cdot \mathbf{B} = 0 \tag{5.3}$$

where $\mathcal{D}$ is a differential operator that characterises the divergence cleaning approach. The solution ψ is applied into the Induction equation in order control the evolution of the magnetic field:

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{uB} - \mathbf{Bu}) + \nabla\psi = \mathbf{0}. \tag{5.4}$$

In order to study the properties of different choices of $\mathcal{D}(\psi)$ over the divergence cleaning process, a formulation for the evolution of $\nabla \cdot \mathbf{B}$ is obtained by taking the divergence of the Equation 5.4, obtaining:

$$\frac{\partial (\nabla \cdot \mathbf{B})}{\partial t} + \nabla^2\psi = 0. \tag{5.5}$$

This equation can be written in terms of $\mathcal{D}(\psi)$ using Equation 5.3:

$$\frac{\partial \mathcal{D}(\psi)}{\partial t} - \nabla^2 \psi = 0. \tag{5.6}$$

Dedner et al. (2002) presents different options for the operator $\mathcal{D}$, obtaining the classes of divergence cleaning methods discussed in the Subsections 5.1.2, 5.1.3, 5.1.4 and 5.1.5. In order to minimise the alterations in the MHD original implementations, in some of these corrections, the terms associated with the divergence cleaning are integrated by using splitting methods.

5.1.1 Splitting methods

Splitting methods are a class of techniques used to divide the integration of a complicate PDE into several simpler parts and solve them independently. In the context of the GLM divergence cleaning methods, this formulation is applied in order to separate the evolution of the MHD equations from the divergence cleaning terms. This allows the inclusion of the GLM approaches with few modifications in the original solver.

In the context of the MHD equations with GLM divergence cleaning, the fluxes or source terms can be decomposed so that:

$$\frac{d\mathbf{q}}{dt} = \mathbf{f}_1(\mathbf{q}) + \mathbf{f}_2(\mathbf{q}), \tag{5.7}$$

where $\mathbf{f}_1(\mathbf{q})$ represents the fluxes from the MHD equation as presented in Equation 4.6 and $\mathbf{f}_2(\mathbf{q})$ are the terms associated with the divergence cleaning. In order to obtain a mathematical description of the splitting methods, the following operators are defined as a time evolution iteration with a time step Δt over the correspondent PDE:

$$\mathcal{H}^{\Delta t}(\mathbf{q}) : \frac{\partial \mathbf{q}}{\partial t} = \mathbf{f}_1(\mathbf{q}) \tag{5.8a}$$

$$\mathcal{S}^{\Delta t}(\mathbf{u}) : \frac{\partial \mathbf{q}}{\partial t} = \mathbf{f}_2(\mathbf{q}). \tag{5.8b}$$

In this work, the splitting method used for the divergence cleaning is the 1^{st} order Godunov splitting, presented in LeVeque (1990) as:

$$\mathbf{q}(t^n + \Delta t) = \mathcal{S}^{\Delta t}\left[\mathcal{H}^{\Delta t}(\mathbf{q}(t^n))\right] \tag{5.9}$$

5.1.2 Parabolic correction

The parabolic correction is characterised by the operator $\mathcal{D}(\psi) = \frac{1}{c_p^2}\psi$, with $c_p \in [0, \infty]$. Its effects over the divergence of the magnetic field can be analysed by applying this operator into the Equation 5.6:

$$\frac{\partial \psi}{\partial t} - c_p^2 \nabla^2 \psi = 0. \tag{5.10}$$

This is a parabolic equation, analogous to the heat equation, which presents diffusive effects over ψ. Considering the relation $\frac{1}{c_p^2}\psi = \mathcal{D}(\psi) = -\nabla \cdot \mathbf{B}$ from Equation 5.3, the parabolic correction diffuses ψ, and consequently the divergence in $\mathbf{B}$, at a diffusion speed c_p^2.

The relation $\psi = -c_p^2 \nabla \cdot \mathbf{B}$, obtained for the parabolic operator, can be applied into Equation 5.4, obtaining:

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{uB} - \mathbf{Bu}) = c_p^2 \nabla (\nabla \cdot \mathbf{B}). \tag{5.11}$$

Thus, the GLM-MHD model with parabolic correction is obtained by changing the induction equation of the MHD model, given in Equation 3.2d, by the Equation 5.11. This new model is solved treating the term $c_p^2 \nabla (\nabla \cdot \mathbf{B})$ as a source term.

5.1.3 Hyperbolic correction

The hyperbolic correction is defined by the operator $\mathcal{D}(\psi) = \frac{1}{c_h^2}\frac{\partial \psi}{\partial t}$, with $c_h \in [0, \infty]$. Performing an analogous analysis as for the parabolic correction, the effects of this correction over the divergence components of the magnetic field are described by the hyperbolic equation:

$$\frac{\partial^2 \psi}{\partial t^2} - c_h^2 \nabla^2 \psi = 0. \tag{5.12}$$

This equation is analogous to the wave equation, which presents advective properties. Therefore, ψ and $\nabla \cdot \mathbf{B}$ are transported at a speed c_h^2. Unlike the parabolic operator, the hyperbolic operator does not provide a solution for ψ to be applied into Equation 5.4. Instead, substituting this operator in the Equation 5.3 produces a new equation which describes the evolution of ψ:

$$\frac{\partial \psi}{\partial t} + c_h^2 \nabla \cdot \mathbf{B} = 0. \tag{5.13}$$

Thus, the GLM-MHD model with the hyperbolic correction consists in including the Equation 5.13 into the system and changing the MHD induction equation for the formulation given in Equation 5.4, which in the conservative form became:

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{uB} - \mathbf{Bu} + \psi \mathbb{I}) = \mathbf{0}, \tag{5.14}$$

where $\mathbb{I}$ is an identity matrix. This equation is integrated so that the divergence cleaning terms are included into the fluxes. This modifies the formulation of the Jacobian matrices of the primitive formulation presented in Equation 3.5. Using the primitive variables $\bar{\mathbf{w}} = (\rho, p, v_x, v_y, v_z, B_x, B_y, B_z, \psi)$, the Jacobian matrices became:

$$\mathbb{A}_x^{GLM} = \begin{bmatrix} u_x & 0 & \rho & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & u_x & \gamma p & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\rho} & u_x & 0 & 0 & -\frac{B_x}{\rho} & \frac{B_y}{\rho} & \frac{B_z}{\rho} & 0 \\ 0 & 0 & 0 & u_x & 0 & -\frac{B_y}{\rho} & -\frac{B_x}{\rho} & 0 & 0 \\ 0 & 0 & 0 & 0 & u_x & -\frac{B_z}{\rho} & 0 & -\frac{B_x}{\rho} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & B_y & -B_x & 0 & -u_y & u_x & 0 & 0 \\ 0 & 0 & B_z & 0 & -B_x & -u_z & 0 & u_x & 0 \\ 0 & 0 & 0 & 0 & 0 & c_h^2 & 0 & 0 & 0 \end{bmatrix} \tag{5.15a}$$

$$\mathbb{A}_y^{GLM} = \begin{bmatrix} u_y & 0 & 0 & \rho & 0 & 0 & 0 & 0 & 0 \\ 0 & u_y & 0 & \gamma p & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & u_y & 0 & 0 & -\frac{B_y}{\rho} & -\frac{B_x}{\rho} & 0 & 0 \\ 0 & \frac{1}{\rho} & 0 & u_y & 0 & \frac{B_x}{\rho} & -\frac{B_y}{\rho} & \frac{B_z}{\rho} & 0 \\ 0 & 0 & 0 & 0 & u_y & 0 & -\frac{B_z}{\rho} & -\frac{B_y}{\rho} & 0 \\ 0 & 0 & -B_y & B_x & 0 & u_y & -u_x & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & B_z & -B_y & 0 & -u_z & u_y & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_h^2 & 0 & 0 \end{bmatrix} \tag{5.15b}$$

$$\mathbb{A}_z^{GLM} = \begin{bmatrix} u_z & 0 & 0 & 0 & \rho & 0 & 0 & 0 & 0 \\ 0 & u_z & 0 & 0 & \gamma p & 0 & 0 & 0 & 0 \\ 0 & 0 & u_z & 0 & 0 & -\frac{B_z}{\rho} & 0 & -\frac{B_x}{\rho} & 0 \\ 0 & 0 & 0 & u_z & 0 & 0 & -\frac{B_z}{\rho} & -\frac{B_y}{\rho} & 0 \\ 0 & \frac{1}{\rho} & 0 & 0 & u_z & \frac{B_x}{\rho} & \frac{B_y}{\rho} & -\frac{B_z}{\rho} & 0 \\ 0 & 0 & -B_z & 0 & B_x & u_z & 0 & -u_x & 0 \\ 0 & 0 & 0 & -B_z & B_y & 0 & u_z & -u_y & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_h^2 & 0 \end{bmatrix} \tag{5.15c}$$

The numerical flux associated with the matrix $\mathbb{A}_x^{GLM}$ can be written as:

$$\mathcal{F}\left(\bar{\mathbf{q}}^L, \bar{\mathbf{q}}^R\right) = \mathbb{A}_x^{GLM} \frac{\partial \bar{\mathbf{w}}}{\partial x} = \mathbb{A}_x^{MHD} \frac{\partial \bar{\mathbf{w}}}{\partial x} + \mathbb{A}_x^{\psi} \frac{\partial \bar{\mathbf{w}}}{\partial x} \tag{5.16}$$

where $\bar{\mathbf{q}}$ is the vector of conservative variables containing the variable ψ, $\mathbb{A}_x^{MHD}$ is the Jacobian matrix of the original MHD formulation with an extra column and line of zeros, corresponding to ψ and its equation, and $\mathbb{A}_x^{\psi}$ contains the terms associated with the divergence cleaning. Thus, the numerical flux for the GLM formulation with the hyperbolic correction can be obtained using the HLL-type numerical fluxes presented in Section 4.3 as:

$$\mathcal{F}^{GLM}\left(\bar{\mathbf{q}}^L, \bar{\mathbf{q}}^R\right) = \mathcal{F}_{\text{HLL/HLLD}}\left(\bar{\mathbf{q}}^L, \bar{\mathbf{q}}^R\right) + \mathcal{F}_{\psi}\left(\bar{\mathbf{q}}^L, \bar{\mathbf{q}}^R\right), \tag{5.17}$$

where the flux $\mathcal{F}_{\psi} = \mathbb{A}_x^{\psi} \frac{\partial \bar{\mathbf{w}}}{\partial x}$. For sake of simplicity, the null lines and columns of the matrix $\mathbb{A}_x^{\psi}$ are neglected. Thus, this flux became:

$$\mathcal{F}_{\psi} = \begin{pmatrix} 0 & 1 \\ c_h^2 & 0 \end{pmatrix} \begin{pmatrix} B_x^* \\ \psi^* \end{pmatrix} \tag{5.18}$$

where $\begin{pmatrix} B_x^* \\ \psi^* \end{pmatrix}$ is the solution for the Riemann problem associated with the matrix $\mathbb{A}_x^{\psi}$ whose eigenvalues are $-c_h$ and c_h. This two wave problem can be solved analogously as performed for the HLL flux in Equation 4.38b, obtaining :

$$\mathcal{F}_{\psi}\left(\bar{\mathbf{q}}^L, \bar{\mathbf{q}}^R\right) = \frac{1}{2} \begin{pmatrix} \psi^R + \psi^L \\ c_h^2 \left(B_x^R + B_x^L\right) \end{pmatrix} - \frac{c_h}{2} \begin{pmatrix} B_x^R - B_x^L \\ \psi^R - \psi^L \end{pmatrix} \tag{5.19}$$

where the parameter c_h is given in function of the Courant number as:

$$c_h = \frac{\sigma}{\Delta t} \min(\Delta x, \Delta y, \Delta z). \tag{5.20}$$

5.1.4 Parabolic-hyperbolic correction

The parabolic-hyperbolic correction is the most used approach among the proposed in Dedner et al. (2002). This correction is based in the combination of the parabolic and the hyperbolic operators, obtaining $\mathcal{D}(\psi) = \frac{1}{c_p^2}\psi + \frac{1}{c_h^2}\frac{\partial \psi}{\partial t}$, with c_p and $c_h \in [0, \infty]$. Applying this operator into Equation 5.6 leads to the telegraph equation:

$$\frac{\partial^2 \psi}{\partial t^2} + \frac{c_h^2}{c_p^2}\frac{\partial \psi}{\partial t} - c_h^2 \nabla^2 \psi = 0, \tag{5.21}$$

which presents advective and diffusive effects determined by the values c_h^2 and c_p^2, respectively.

As the GLM-MHD model with hyperbolic correction, the induction equation for the parabolic-hyperbolic correction is replaced by Equation 5.14. In this case, the equation which describes the evolution of ψ, obtained by substituting the parabolic-hyperbolic operator in Equation 5.3, is given by:

$$\frac{\partial \psi}{\partial t} + c_h^2 \nabla \cdot \mathbf{B} = -\frac{c_h^2}{c_p^2}\psi. \tag{5.22}$$

The parabolic-hyperbolic correction is executed using a splitting method so that the operators $\mathcal{H}^{\Delta t}$ and $\mathcal{S}^{\Delta t}$ corresponds to the integrating the equations:

$$\mathcal{H}^{\Delta t}(\mathbf{q}): \begin{array}{l} \dfrac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{uB} - \mathbf{Bu} + \psi \mathbb{I}) = \mathbf{0} \\ \dfrac{\partial \psi}{\partial t} + c_h^2 \nabla \cdot \mathbf{B} = \mathbf{0} \end{array} \tag{5.23a}$$

$$\mathcal{S}^{\Delta t}(\mathbf{q}): \frac{\partial \psi}{\partial t} = -\frac{c_h^2}{c_p^2}\psi. \tag{5.23b}$$

where the operator $\mathcal{H}^{\Delta t}$ is identical to the hyperbolic correction presented in Equation 5.17. Thus, the first step of the parabolic-hyperbolic correction consists in performing a time evolution iteration using the hyperbolic correction.

The operator $\mathcal{S}^{\Delta t}$ is associated with the parabolic terms. Considering the solution

$\hat{\psi}$, obtained after applying the operator $\mathcal{H}^{\Delta t}$, as initial condition, the equation associated with the operator $\mathcal{S}^{\Delta t}$ has the analytical solution:

$$\psi(t) = \hat{\psi} e^{\frac{-c_h^2}{c_p^2} t}. \tag{5.24}$$

Therefore, the variable ψ is updated to the next time instant by the equation:

$$\psi^{n+1} = \hat{\psi} e^{-c_h \alpha_p \Delta t / \min(\Delta x, \Delta y, \Delta z)}, \tag{5.25}$$

where $\alpha_p = \min(\Delta x, \Delta y, \Delta z) \frac{c_h}{c_p^2}$.

5.1.5 Elliptic correction

The elliptic correction is a approach analogous to the projection method presented in Brackbill and Barnes (1980). In the context of the GLM divergence cleaning, it is defined by the operator $\mathcal{D}(\psi) = 0$. Applying this operator into Equation 5.6 leads to the Poisson equation:

$$\nabla^2 \psi = 0. \tag{5.26}$$

This operator is expected to compute and remove the divergence components globally. As the parabolic-hyperbolic correction, this strategy solves the modified Induction equation, presented in Equation 5.4, using a splitting method. The first step consists in solving the original Induction equation, producing a magnetic field with divergence errors denoted by $\hat{\mathbf{B}}$. Then, in the second step, the following equation is solved:

$$\frac{\partial \hat{\mathbf{B}}}{\partial t} = -\nabla \psi. \tag{5.27}$$

Discretising the time derivative, this equation became:

$$\mathbf{B}^{n+1} = \hat{\mathbf{B}} - \Delta t \nabla \psi. \tag{5.28}$$

The challenge of this approach lies in the calculation of the scalar field ψ. For that, is considered the divergence of this equation:

$$\nabla \cdot \mathbf{B}^{n+1} = \nabla \cdot \hat{\mathbf{B}} - \Delta t \nabla \cdot (\nabla \psi). \tag{5.29}$$

Considering that $\nabla \cdot \mathbf{B}^{n+1} = 0$, the Equation 5.29 became the Poisson equation:

$$\nabla^2 \psi = \frac{1}{\Delta t} \nabla \cdot \hat{\mathbf{B}}. \tag{5.30}$$

This equation can be numerically solved for ψ. Then, this solution is applied into Equation 5.28 to produce a divergence free magnetic field.

The Poisson equation is an elliptic problem which requires an implicit method to be solved. This implies in the resolution of a system of linear equations. In order to illustrate this system, the solution vector $\boldsymbol{\psi}$ is defined. This vector is a column vector with $N = N_x N_y$ lines for 2D problems or $N = N_x N_y N_z$ lines for 3D problems. The elements of $\boldsymbol{\psi}$, denoted as ψ_p, consists in a reordered sequence of all values of $\psi_{i,j}$ or $\psi_{i,j,k}$. This reordered sequence is indexed by p, which is mapped to the indexes i, j and k by the relation:

- $p = i + jN_x$, for 2D problems;
- $p = i + jN_x + kN_xN_y$, for 3D problems.

This formulation allows to define the discretisation matrix $\mathbb{A}$ and the vector $\mathbf{b}$ with the values of $\frac{1}{\Delta t} \nabla \cdot \hat{\mathbf{B}}$ mapped according the vector $\boldsymbol{\psi}$. Thus, the Poisson equation is reduced to the linear system $\mathbb{A}\boldsymbol{\psi} = \mathbf{b}$. Linear systems obtained from refined discretisations are too big for approaches like Gauss' elimination to handle, requiring iterative methods, discussed in the following.

5.2 Relaxation Methods

Relaxation methods are particularly interesting to solve linear systems derived from the discretisation of a elliptic PDE (VARGA, 1999). These methods are based fixed point iterations, which the system is rewritten in order to represent the solution as a fixed point of a function $\mathbf{f}$. A fixed point is an element of the domain of a function that maps itself in the image, that is:

$$\boldsymbol{\psi} = \mathbf{f}(\boldsymbol{\psi}), \tag{5.31}$$

According the fixed point theorem, a sequence started from a guessing solution $\boldsymbol{\psi}^{(0)}$ applied recursively into the relation

$$\boldsymbol{\psi}^{(m+1)} = \mathbf{f}\left(\boldsymbol{\psi}^{(m)}\right), \qquad m = 0, 1, \ldots \tag{5.32}$$

converges to the solution $\boldsymbol{\psi}$ if some requirements are fulfilled (BURDEN; FAIRES, 1989).

Jacobi Method

The Jacobi method is one of the simplest approaches for solving linear systems numerically. It consist in decomposing the discretisation matrix $\mathbb{A}$ as the sum of the matrix $\mathbb{D}$, with the elements of the diagonal of $\mathbb{A}$, and the matrix $\mathbb{R}$ with the remaining elements of $\mathbb{A}$, *i.e.*:

$$\mathbb{D} = \begin{bmatrix} a_{1,1} & 0 & 0 & \dots & 0 \\ 0 & a_{2,2} & 0 & \dots & 0 \\ 0 & 0 & a_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{N,N} \end{bmatrix}, \qquad \mathbb{R} = \begin{bmatrix} 0 & a_{1,2} & a_{1,3} & \dots & a_{1,N} \\ a_{2,1} & 0 & a_{2,3} & \dots & a_{2,N} \\ a_{3,1} & a_{3,2} & 0 & \dots & a_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & a_{N,3} & \dots & 0 \end{bmatrix} \tag{5.33}$$

Using this decomposition, the system $\mathbb{A}\boldsymbol{\psi} = \mathbf{b}$ is rewritten as:

$$(\mathbb{D} + \mathbb{R})\,\boldsymbol{\psi} = \mathbf{b} \tag{5.34}$$

The usage of the diagonal matrix $\mathbb{D}$ is particularly interesting because it is easily inverted. Thus, the solution of the system can be written in a fixed point formulation as:

$$\boldsymbol{\psi} = \mathbb{D}^{-1}\left(\mathbf{b} - \mathbb{R}\boldsymbol{\psi}\right). \tag{5.35}$$

The Jacobi method is defined by using this relation as a fixed point method to iterate an initial guess $\boldsymbol{\psi}^{(0)}$ until convergence, *i.e.* the residual after m iterations $\mathfrak{r} = \mathbb{A}\boldsymbol{\psi}^{(m)} - \mathbf{b}$ became lower than a predefined tolerance. Thus, each value $\psi_p \in \boldsymbol{\psi}$ after $m+1$ iterations are written as:

$$\psi_p^{(m+1)} = \frac{1}{a_{p,p}}\left(\mathbf{b}_p - \sum_{q \neq p} a_{p,q}\psi_q^{(m)}\right). \tag{5.36}$$

In particular, for the Poisson equation associated with the Elliptic solver for 2D

problems, the fixed-point equation, using the ijk mapping, became:

$$\psi_{i,j}^{(m+1)} = \frac{\delta_{2\mathrm{D}}}{2}\left[\left(\psi_{i+1,j}^{(m)} + \psi_{i-1,j}^{(m)}\right)\Delta y^2 + \left(\psi_{i,j+1}^{(m)} + \psi_{i,j-1}^{(m)}\right)\Delta x^2 - \frac{\Delta x^2 \Delta y^2}{\Delta t}\nabla \cdot \hat{\mathbf{B}}_{i,j}\right], \tag{5.37}$$

where $\delta_{2\mathrm{D}} = \left(\Delta x^2 + \Delta y^2\right)^{-1}$. For the 3D formulation, the discretisation:

$$\begin{aligned}\psi_{i,j,k}^{(m+1)} =& \frac{\delta_{3\mathrm{D}}}{2}\Big[\left(\psi_{i+1,j,k}^{(m)} + \psi_{i-1,j,k}^{(m)}\right)\Delta y^2 \Delta z^2 + \left(\psi_{i,j+1,k}^{(m)} + \psi_{i,j-1,k}^{(m)}\right)\Delta x^2 \Delta z^2 \\ &+ \left(\psi_{i,j,k+1}^{(m)} + \psi_{i,j,k-1}^{(m)}\right)\Delta x^2 \Delta y^2 - \frac{\Delta x^2 \Delta y^2 \Delta z^2}{\Delta t}\nabla \cdot \hat{\mathbf{B}}_{i,j,k}\Big],\end{aligned} \tag{5.38}$$

with $\delta_{3\mathrm{D}} = \left(\Delta x^2 \Delta z^2 + \Delta y^2 \Delta z^2 + \Delta x^2 \Delta y^2\right)^{-1}$ is used.

By this formulation, the iteration from m to $m+1$ do not require any value obtained during this iteration. Thus, the computations of $\boldsymbol{\psi}^{(m+1)}$ can be performed in no particular order. This allows the usage of parallelism to divide the workload of a Jacobi method's iteration among different processors. However, frequently during the Jacobi iteration, a value $\psi_p^{(m)}$ is required to iterate its adjacent cells after $\psi_p^{(m+1)}$ was computed. Therefore, both values $\boldsymbol{\psi}^{(m)}$ and $\boldsymbol{\psi}^{(m+1)}$ need to be stored, increasing the memory requirements significantly, which is not desirable for very refined computations.

Gauss–Seidel Method

In order to avoid this extra memory storage, the Gauss–Seidel method is defined so that the values $\psi_p^{(m)}$ can be updated to $\psi_p^{(m+1)}$ right after their computation. The Gauss-Seidel method is constructed similarly as the Jacobi method. The main difference is the decomposition of the matrix $\mathbb{A}$ as the sum of the diagonal matrix $\mathbb{D}$, with the matrices $\mathbb{L}$ and $\mathbb{U}$ defined as:

$$\mathbb{L} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ a_{2,1} & 0 & 0 & \dots & 0 \\ a_{3,1} & a_{3,2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & a_{N,3} & \dots & 0 \end{bmatrix}, \qquad \mathbb{U} = \begin{bmatrix} 0 & a_{1,2} & a_{1,3} & \dots & a_{1,N} \\ 0 & 0 & a_{2,3} & \dots & a_{2,N} \\ 0 & 0 & 0 & \dots & a_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \tag{5.39}$$

Using this decomposition, the system $\mathbb{A}\boldsymbol{\psi} = \mathbf{b}$ is rewritten as:

$$\left(\mathbb{D} + \mathbb{L} + \mathbb{U}\right)\boldsymbol{\psi} = \mathbf{b}. \tag{5.40}$$

These terms can be reordered as:

$$(\mathbb{D} + \mathbb{L})\,\boldsymbol{\psi} = \mathbf{b} - \mathbb{U}\boldsymbol{\psi}. \tag{5.41}$$

This formulation can be interpreted as a fixed point equation that converges to $(\mathbb{D} + \mathbb{L})\,\boldsymbol{\psi}$. Therefore, the fixed point iteration is obtained:

$$(\mathbb{D} + \mathbb{L})\,\boldsymbol{\psi}^{(m+1)} = \mathbf{b} - \mathbb{U}\boldsymbol{\psi}^{(m)}. \tag{5.42}$$

Reordering these terms, the values $\boldsymbol{\psi}^{(m+1)}$ are computed as:

$$\boldsymbol{\psi}^{(m+1)} = \mathbb{D}^{-1}\left(\mathbf{b} - \mathbb{L}\boldsymbol{\psi}^{(m+1)} - \mathbb{U}\boldsymbol{\psi}^{(m)}\right). \tag{5.43}$$

Therefore, a single value $\psi_p^{(m)}$ is updated as:

$$\psi_p^{(m+1)} = \frac{1}{a_{p,p}}\left(\mathbf{b}_p - \sum_{q<p} a_{p,q}\psi_q^{(m+1)} - \sum_{q>p} a_{p,q}\psi_q^{(m)}\right). \tag{5.44}$$

In special, for the Poisson equation, the fixed point iteration for a value ψ_p is identical as the presented in Equations 5.37 and 5.38 except for replacing the usage of the values $\psi_q^{(m)}$, with $q < p$, for the updated values $\psi_q^{(m+1)}$. Therefore, the fixed point iteration for the Poisson equation for two and three dimensions, using the ijk mapping, became:

$$\psi_{i,j}^{(m+1)} = \frac{\delta_{\text{2D}}}{2}\left[\left(\psi_{i+1,j}^{(m)} + \psi_{i-1,j}^{(m+1)}\right)\Delta y^2 + \left(\psi_{i,j+1}^{(m)} + \psi_{i,j-1}^{(m+1)}\right)\Delta x^2 - \frac{\Delta x^2\Delta y^2}{\Delta t}\nabla\cdot\hat{\mathbf{B}}_{i,j}\right], \tag{5.45}$$

$$\begin{aligned}\psi_{i,j,k}^{(m+1)} =& \frac{\delta_{\text{3D}}}{2}\Big[\left(\psi_{i+1,j,k}^{(m)} + \psi_{i-1,j,k}^{(m+1)}\right)\Delta y^2\Delta z^2 + \left(\psi_{i,j+1,k}^{(m)} + \psi_{i,j-1,k}^{(m+1)}\right)\Delta x^2\Delta z^2 \\ &+ \left(\psi_{i,j,k+1}^{(m)} + \psi_{i,j,k-1}^{(m+1)}\right)\Delta x^2\Delta y^2 - \frac{\Delta x^2\Delta y^2\Delta z^2}{\Delta t}\nabla\cdot\hat{\mathbf{B}}_{i,j,k}\Big],\end{aligned} \tag{5.46}$$

By this formulation, the storage of the value $\psi_p^{(m)}$ is no longer required after the value $\psi_p^{(m+1)}$ is obtained, implying in a formulation with better memory management than the Jacobi method, and also converging with less iterations (BURDEN; FAIRES, 1989). However, this formulation does not allow the use of parallelism since the method requires values that are obtained during the iteration, implying that the iteration should be performed in a particular order, rendering the parallelisation ineffective.

In the context of the Poisson equation, a parallel formulation for the Gauss-Seidel method can be obtained through alternative formulations such as the Red/Black Gauss-Seidel (KARNIADAKIS; KIRBY II, 2003; BRIGGS et al., 2000). Other parallel strategies are developed in the works of Courtecuisse and Allard (2009) and Dolwithayakul et al. (2011) with applications in MPI and CUDA.

The Red/Black Gauss–Seidel approach consists in flagging the cells $\mathcal{C}$ as Red or Black. Then, the fixed point iteration is splitted into two steps denominated Red and Black steps. The Red step is performed by iterating and updating every red cells of the mesh. The Black step has an analogous definition.

The cells are flagged as Red or Black according to the rule:

- 2D case: $\text{Colour}(\mathcal{C}_{i,j}) = \begin{cases} \text{Black, if } i+j \text{ is even} \\ \text{Red, otherwise} \end{cases}$
- 3D case: $\text{Colour}(\mathcal{C}_{i,j,k}) = \begin{cases} \text{Black, if } i+j+k \text{ is even} \\ \text{Red, otherwise} \end{cases}$

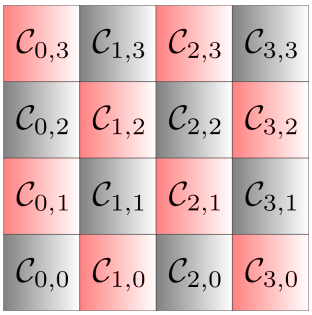
An example of a 2D mesh with the Red/Black flagging is presented in Figure 5.1. When performing the Red step, the iteration of a red cell requires only the values of the surrounding black cells, which are known *a priori*. Since the iteration of every red cell is independent of any unknown value, the red cells can be divided among the processors trivially.

After the Red step, all the red cells have their updated values at iteration $m+1$, which is the required information to perform the Black step. As well as the Red step, during the Black step, the black cells can be divided among the processors, since their required updated red cells are known beforehand. Thus, the Red/Black Gauss–Seidel iteration can be represented by the operator:

$$\boldsymbol{\psi}^{(m+1)} = \text{Black step}\left[\text{Red step}\left(\boldsymbol{\psi}^{(m)}\right)\right]. \tag{5.47}$$

The Red/Black Gauss–Seidel approach combines both the faster convergence and memory management gain of the original Gauss-Seidel method with the parallelisation obtainable with the Jacobi method. This approach is the technique applied in this work for solving the Poisson equation. Nevertheless, the Red/Black Gauss–Seidel approach still requires a huge amount of iterations in order to obtain convergence.

Figure 5.1 - Red/Black Gauss-Seidel colour distribution in a 2D mesh.



Source: Author's production.

In this work is proposed the use of multigrid methods in order to accelerate this convergence and perform fewer, and cheaper, Gauss-Seidel iterations.

5.3 Multigrid methods

Relaxing methods requires a large amount of iterations to converge to the desired solution. In special, for multi-dimensional problems, the number of equations and variables became too large when finer meshes or grids, are used, turning a single iteration of the relaxing method very costly.

In this context, a methodology capable of reducing the cost of a single iteration and accelerating the convergence, thus requiring less iterations, became essential. One of the most successful approaches for this problem are the multigrid (MG) methods (YAVNEH, 2006; WESSELING, 2004; ARRARÁS et al., 2015). These methods represents the vector of data using a series of coarser grids so that the refinement of a grid in the level $\ell = \ell_{min}, \ldots, L$ is always 2^d times coarser than the grid representing the next refinement level. The most refined level of this sequence is the level L, which uses the same refinement parameters of the simulation.

The MG methods are based in the idea whereupon techniques like the Jacobi and Gauss-Seidel methods vanishes the high frequencies of the error between the exact and the iterated solutions, denoted by $\mathbf{e} = \boldsymbol{\psi} - \boldsymbol{\psi}^{(m)}$, in few iterations. On the other hand, the lower frequencies requires a huge amount of iterations (BRIGGS et al., 2000; URBAN, 2009) to be solved. Using this error definition, the system $\mathbb{A}^L \boldsymbol{\psi}^L = \mathbf{b}^L$ can be rewritten as:

$$\mathbb{A}^L \left(\boldsymbol{\psi}^{(m)\,L} + \mathbf{e}^L \right) = \mathbf{b}^L \tag{5.48}$$

Considering that the high frequencies of $\mathbf{e}^L$ are vanished as $\boldsymbol{\psi}^{(m)\,L}$ is iterated a few times, this error is considered to be dominated by low frequency terms after a

predefined ν_{pre} number of iterations. The MG methods aims for calculating the error $\mathbf{e}^{\,L}$ instead of keep iterating $\boldsymbol{\psi}^{(m)\,L}$ until obtain convergence.

In order to calculate the error $\mathbf{e}^{\,L}$, the Equation 5.48 is reordered as:

$$\mathbb{A}^{L}\mathbf{e}^{\,L} = \mathbf{b}^{\,L} - \mathbb{A}^{L}\boldsymbol{\psi}^{(\nu_{\text{pre}})\,L}. \tag{5.49}$$

Substituting the right side for the residual $\mathfrak{r}^{\,L} = \mathbf{b}^{\,L} - \mathbb{A}^{L}\boldsymbol{\psi}^{(\nu_{\text{pre}})\,L}$:

$$\mathbb{A}^{L}\mathbf{e}^{\,L} = \mathfrak{r}^{\,L}, \tag{5.50}$$

The idea of solving this new system for $\mathbf{e}^{\,L}$ is combined with the property that projecting a solution into a lower resolution level transforms their low frequency components into higher frequency components (BRIGGS et al., 2000). This allows to solve a system in the next coarser level, obtaining a solution $\mathbf{e}_{*}^{\,L-1}$ so that the error $\mathbf{e}^{\,L}$ is reconstructed as:

$$\mathbf{e}^{\,L} = \mathcal{P}^{L-1\to L}\mathbf{e}_{*}^{\,L-1}, \tag{5.51}$$

where $\mathcal{P}^{L-1\to L}$ is a prolongation operator performed as a multi-linear interpolation as defined in Deiterding (2011).

Solving the system in the coarser grid requires fewer iterations than would be required for solving in the finer grid. Besides that, a single iteration in the coarser grid have a lower computational cost for the relaxing method iteration, due to presenting one quarter of the equations and variables for 2D cases, and an eighth of them for the 3D cases.

Therefore, instead of solving the Equation 5.50, the following system is solved in the next coarser level $L-1$:

$$\mathbb{A}^{L-1}\mathbf{e}_{*}^{\,L-1} = \mathfrak{r}_{*}^{\,L-1}, \tag{5.52}$$

where $\mathbb{A}^{L-1}$ is the discretisation matrix for the level $L-1$ and $\mathfrak{r}_{*}^{\,L-1}$ is obtained by the projection of the residual $\mathfrak{r}^{\,L}$:

$$\mathfrak{r}_{*}^{\,L-1} = \mathcal{P}^{L\to L-1}\mathfrak{r}^{\,L}, \tag{5.53}$$

where the projection operator $\mathcal{P}^{L\to L-1}$ is defined as an averaging function (DEITERDING, 2011). .

Analogously as performed in Equation 5.48 for the most refined level, the solution

$\mathbf{e}_*^{L-1}$ can be decomposed as:

$$\mathbb{A}^{L-1}\left(\mathbf{e}_*^{(m)\,L-1} + \mathbf{e}^{\,L-1}\right) = \mathfrak{r}_*^{L-1}, \tag{5.54}$$

where $\mathbf{e}^{\,L-1}$ are the components with low frequency of the error $\mathbf{e}_*^{L-1}$ and $\mathbf{e}_*^{(m)\,L-1}$ is the approximation of this error after m iterations of the relaxing method. These low frequency components $\mathbf{e}^{\,L-1}$ can be solved analogously as the error presented in Equation 5.50 as:

$$\mathbb{A}^{L-1}\mathbf{e}^{\,L-1} = \mathfrak{r}^{\,L-1}, \tag{5.55}$$

where $\mathfrak{r}^{\,L-1} = \mathfrak{r}_*^{L-1} - \mathbb{A}^{L-1}\mathbf{e}_*^{(\nu_{\text{pre}})\,L-1}$.

This process of solving the lower frequency components of the error in a coarser grid can be repeated recursively until a predefined coarsest level ℓ_{min}, where the component $\mathbf{e}^{\,\ell_{min}}$ can be neglected after some iterations.

Once the error $\mathbf{e}_*^{(\nu_{\text{pre}})\,\ell}$, obtained from a level $\ell \neq L$, is considered to converge to the exact solution of the system $\mathbb{A}^{\ell}\mathbf{e}_*^{\ell} = \mathfrak{r}_*^{\ell}$, it is prolongated to the level $\ell + 1$ in order to approximate the value $\mathbf{e}^{\,\ell+1}$ analogously as presented in Equation 5.51 for the level $L - 1$.

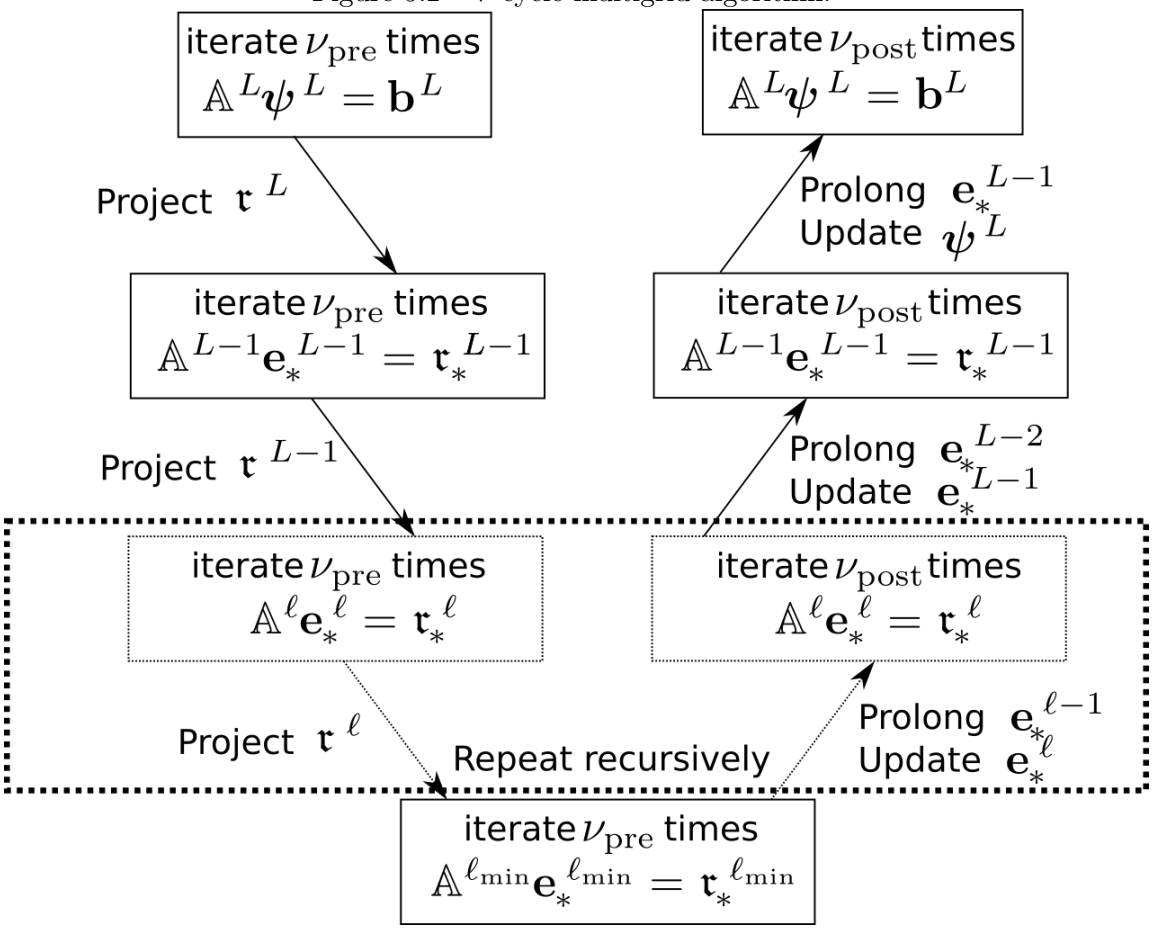
After the low components of the error $\mathbf{e}^{\,\ell}$ are obtained, they are added to the solution $\mathbf{e}_*^{(\nu_{\text{pre}})\,\ell}$, as presented in Equation 5.54 for $\ell = L - 1$. Then, the system is iterated again for a predetermined number of iterations ν_{post}, obtaining a solution $\mathbf{e}_*^{(\nu_{\text{pre}}+\nu_{\text{post}})\,\ell}$ that converges to $\mathbf{e}_*^{\ell}$. This process can be applied recursively until the solution $\boldsymbol{\psi}$ is obtained in the finest level.

In general, there is no particular order of which order the projection and prolongation operator should be applied, as long as the finest and the coarsest levels are respected. This leads to infinity variations of MG methods, where the most commons are the V-cycle, the W-cycle and the full MG (URBAN, 2009). In particular, the V-cycle, illustrated in Figure 5.2, is chosen for this work.

5.4 Combining the elliptic and parabolic-hyperbolic corrections

One of the original contributions of this work is the combination of explicit parabolic-hyperbolic and the implicit elliptic approaches. The idea is to take the advantages of both methods, the low computational cost of the explicit method and the insurance of the divergence constraint provided by the implicit method.

Figure 5.2 - V-cycle multigrid algorithm.



Source: Author's production.

The proposed method, denominated GLM triple correction, consists in performing the explicit parabolic-hyperbolic approach, producing a small divergence over $\mathbf{B}$, until a cleaning criteria is fulfilled. Then, the implicit elliptic method is applied in order to restore the divergence constraint. For this work, the cleaning criteria chosen is perform the elliptic operator after every number of time evolution iterations n_{MG}. In order to reduce the computational cost of this step, this work applies the MG formulation over a Red/Black Gauss-Seidel method to take advantage of both the efficiency of the MG method and the parallel approach of the Red/Black Gauss-Seidel method.

6 PATCH-STRUCTURED ADAPTIVE MESH REFINEMENT

The computational discretisation of the physical domain as presented in Section 4.1 divides the entire domain into cells of equal refinement. However, the choice of a proper refinement for this type of discretisation is challenging, once a coarse refinement may cause the solution to not be properly represented in the mesh, specially if it contains localised structures or steep gradients, causing loss of information. On the other hand, a very fine refinement leads to a considerable amount of unnecessary computations in these cases, wasting a lot of computational time and memory.

In this context, adaptive techniques are proposed to overcome these limitations. These techniques maximise the efficiency of the simulation by using an adaptive mesh which is more refined in the regions where the localised structures are presented, and is coarser in the smooth regions. The generation of adaptive meshes can be divided into two approaches: structured and unstructured meshes.

In general, a mesh is defined as a set of geometrical elements that covers the physical domain without gaps and overlapping elements. Thus, an unstructured mesh covers the physical domain using elements that presents an irregular pattern, such as non similar triangles or tetrahedrons. On the other hand, structured meshes are defined by elements that presents a pattern on its form, such as similar rectangles or parallelepipeds. Frequently, in these meshes, the elements presents a positioning pattern, normally being indexed alongside with the coordinate axes.

The main advantage of unstructured meshes is its superior geometrical flexibility. This type of approach, usually implemented using cell-based data structures, needs to store all the adjacent cells explicitly. The lack of a simpler methodology to go through the elements of the mesh causes the memory access during the simulations to be irregular, implying in a poor performance on vector and super-scalar computers. Among another disadvantages of this approach are the difficulty in obtaining higher order solutions and the aspect ratio of each cell.

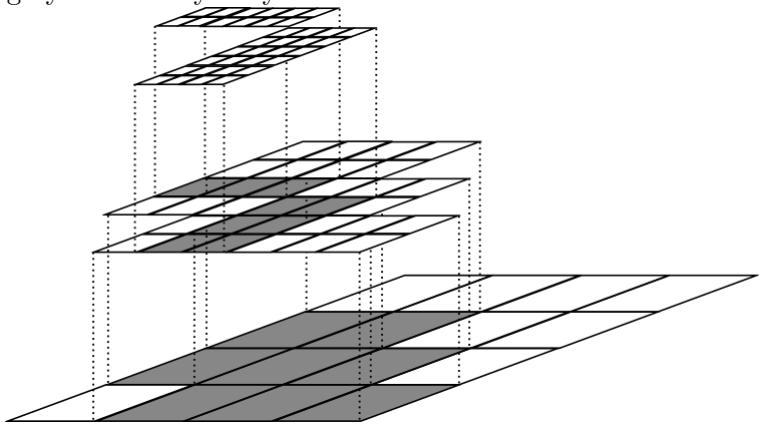
In contrast, the structured meshes are adequate when the geometric flexibility is not a priority in the application. In particular, the usage of rectangular domains allows some optimisation in comparison with the algorithms required for the unstructured meshes, such as the implementation of a single numerical scheme for every region, independently of the refinement.

The construction of the adaptive mesh is based in refinement criteria techniques,

which measures the local smoothness of the solution in every mesh element. If the result of the criteria exceeds a predetermined value ϵ, the mesh element is flagged for refinement. After that, the flagged elements are replaced, or overlayed, by a patch of r^d more refined elements, where r is the mesh refinement factor and d is the number of dimensions of the problem. In particular, this work always uses the refinement factor r as two.

In special, this work uses the patch-Structured Adaptive Mesh Refinement (SAMR) method, as presented in Deiterding (2011), to implement the proposed MHD code. This formulation differs by, instead of refining every flagged cell, they are clustered into rectangular boxes containing both flagged and non-flagged cells so that the percentage of flagged cells exceeds a predefined parameter ν, as described in Section 6.2. Then, every cluster of flagged cells are overlayed by a finer submesh. Applying this algorithm recursively in the finer submeshes, a mesh hierarchy as illustrated in Figure 6.1, is obtained.

Figure 6.1 - Example of hierarchy of retangular submeshes. The clustered regions marked in grey are overlayed by a finer submesh.



Source: Author's production.

Considering the field of FV schemes for hyperbolic partial equations, the SAMR methods were first introduced in the works of Berger and Oliger (BERGER; OLIGER, 1984; BERGER, 1982). In this earlier approach, a more refined submesh could be overlapped over a coarser mesh without any restriction of alignment between overlapping meshes, allowing the use of rotationed meshes, which requires complicated interpolation operations. Posteriorly, Berger and Corella (BERGER, 1982; BERGER; CORELLA, 1989) proposed a simpler version of the SAMR, which every mesh of the hierarchy must be aligned, allowing simpler interpolation operations. In Bell et al.

(1994), this version was demonstrated to be more efficient, specially with vector and super-scalar computers.

6.1 Mesh Hierarchy

The SAMR mesh hierarchy is composed by a sequence of meshes $\mathcal{G}^\ell$ enumerated by the refinement levels $\ell = 0, 1, \ldots, L$ with refinement Δx^ℓ, Δy^ℓ and Δz^ℓ so that they present a constant ratio r between adjacent levels, that is:

$$\frac{\Delta x^{\ell+1}}{\Delta x^\ell} = \frac{\Delta y^{\ell+1}}{\Delta y^\ell} = \frac{\Delta z^{\ell+1}}{\Delta z^\ell} = r. \tag{6.1}$$

Alternatively, the refinement of any level can be obtained from the base mesh refinement as:

$$\Delta x^\ell = \frac{1}{r^\ell}\Delta x^0 \qquad \Delta y^\ell = \frac{1}{r^\ell}\Delta y^0 \qquad \Delta z^\ell = \frac{1}{r^\ell}\Delta z^0 \tag{6.2}$$

These meshes are embedded so that the domain covered by a finer mesh is also covered by the next coarser mesh, that is:

$$\mathcal{G}^L \subset \mathcal{G}^{L-1} \subset \cdots \subset \mathcal{G}^0 = \Omega \tag{6.3}$$

where the base mesh $\mathcal{G}^0$ cover the entire physical domain Ω. These meshes are divided into a set of non overlapping rectangular submeshes $\mathcal{G}^\ell_m$ so that:

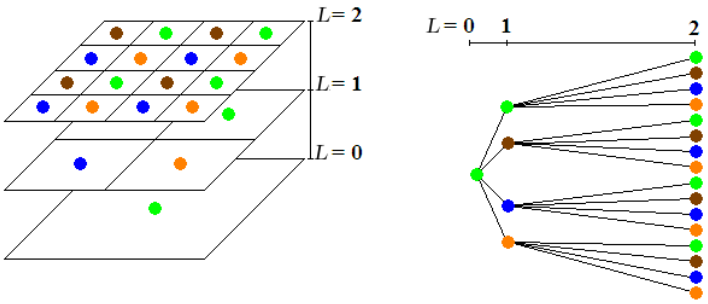
$$\mathcal{G}^\ell := \cup_{m=1}^{M_\ell} \mathcal{G}^\ell_m \qquad \text{with } \mathcal{G}^\ell_{m_1} \cap \mathcal{G}^\ell_{m_2} = \emptyset,\ m_1 \neq m_2 \tag{6.4}$$

where M_ℓ is the number of submeshes, or patches, used to represent the mesh $\mathcal{G}^\ell$. The set of meshes $\mathcal{G}^\ell$ with $\ell = 0, 1, \ldots, L$ is implemented as a tree so that each node of the level ℓ corresponds to a patch $\mathcal{G}^\ell_m$. Furthermore, these nodes presents a parent-child relation so that every node corresponding to a patch contained in $\mathcal{G}^{\ell+1}$ is a child of the node corresponding to the overlayed patch in $\mathcal{G}^\ell$. This hierarchy is exemplified in Figure 6.2. Moreover, this tree allows the definition for multiples parents for an element due to the possibility of a finer patch being placed over two coarser patches as shown in Figure 6.1.

6.1.1 Patch boundaries

The mesh hierarchy representation as multiple independent rectangular submeshes allows a single implementation of the numerical scheme for every patch, independently of the refinement level. Thereby, the time evolution process can be performed

Figure 6.2 - Adaptive mesh and corresponding tree structure to store the patches.

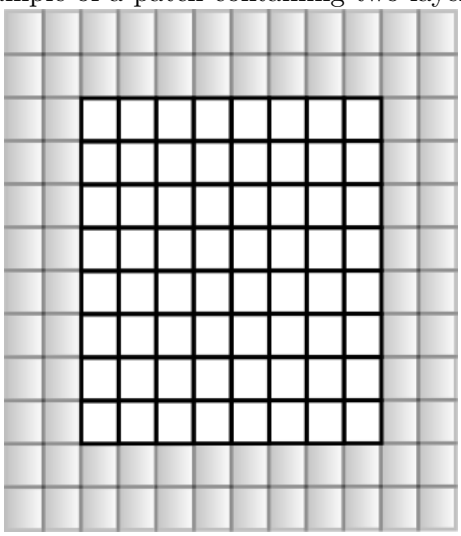


Source: Author's production.

for every patch individually. However, to compute the fluxes in the boundaries of every patch, the solution of the cells in adjacent patches are required. This restriction compromises the patch independence to perform the time evolution.

In order to overcome this limitation, the patch structure are complemented with extra auxiliary cells, called ghost cells, at their boundaries. This allows the boundary values to be stored in the same data structure as the submesh. An schematic example of a patch containing two layers of ghost cells is presented in Figure 6.3.

Figure 6.3 - Example of a patch containing two layers of ghost cells.



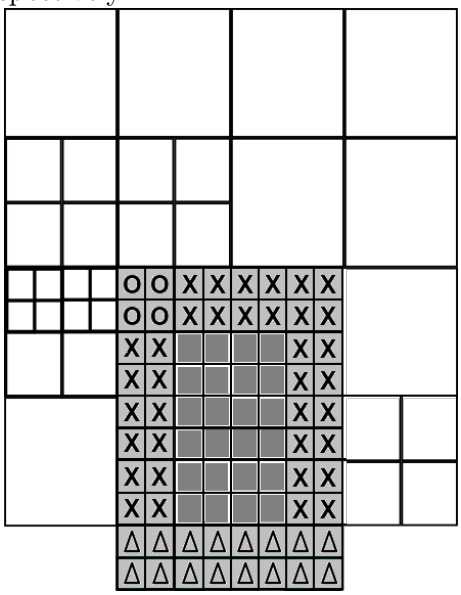
Source: Author's production.

Before and during the time evolution, the ghost cell values must be set or updated. As illustrated in Figure 6.4, the ghost cells can be divided into three cases regarding to its positioning over the mesh hierarchy. The solution for the ghost cells are obtained

according to its positioning case:

- Physical boundary: the ghost cells are updated according to the problem boundary conditions.
- Same level patch: the ghost cells are updated by coping the solution from the adjacent patch.
- Coarser patch: the ghost cells are interpolated by the solution in the coarser level using a multi-linear interpolation.

Figure 6.4 - Patch with ghost cells placed over the mesh hierarchy. The ghost cells marked with Δ represents ghost cells that overlays the physical domain of the problem, while the ghost cells marked with X and O overlays patches of lower and same refinement, respectively.



Source: Author's production.

6.2 Flagging algorithm

The construction of the adaptive meshes in the context of SAMR methods are based in overlaying finer patches over coarser patches that presents a region of interest. These regions are estimated by flagging cells of the coarser patch that fulfil a refinement criteria. Then, using the clustering algorithm presented in Bell et al. (1994), the flagged cells are grouped alongside with a some non flagged cells forming the regions that requires refinement, which are replaced by new more refined

patches. This clustering algorithm is briefly discussed in Figure 6.5. This strategy uses a parameter ν, which defines the stop criteria of the clustering process. In this work, typical values are $\nu = 0.85$ for 2D cases, and $\nu = 0.8$ for 3D cases.

During the construction of the adaptive mesh, the flagging process consists in applying an operator that requires the solution of the cell and its neighbours. If the result from this operator exceed a threshold value ϵ, the cell is flagged for refinement. In particular, the proposed MHD solver implements the following refinement criteria:

- Scaled gradient (SG): This operator flags a cell if the difference, in modulus, between its solution and the solution in a adjacent cell exceeds the threshold value for a predetermined variable, i.e., considering a cell at a position (i, j), it is flagged for refinement if at least one of the following relations is satisfied. In special, for the mass density,

$$| \rho_{i+1,\,j} - \rho_{i,\,j} | > \epsilon, \; | \rho_{i,\,j+1} - \rho_{i,\,j} | > \epsilon, \; | \rho_{i+1,\,j+1} - \rho_{i,\,j} | > \epsilon.$$

  More details in this criteria can be found in Deiterding (2011).

- Multiresolution (MR): This operator is based in the adaptive multiresolution method proposed in Harten (1995). It uses a wavelet based technique to predict the expected solution in a finer scale by a operator described in Roussel (2003) and Moreira Lopes (2014) for 1D, 2D and 3D meshes. More detail in this multiresolution approach can be found in Domingues et al. (2011).

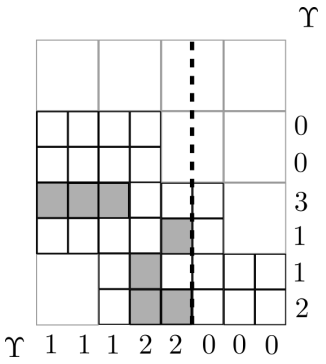
  This thesis uses a level based thresholding strategy so that the parameter ϵ has a different value depending to the refinement. Using the Harten's strategy, this level dependent ϵ^ℓ is calculated as:

$$\epsilon^\ell = \frac{\epsilon}{\Delta^3 r} 2^{d\,(\ell - L)}, \;\; 0 \leq \ell \leq L. \tag{6.5}$$

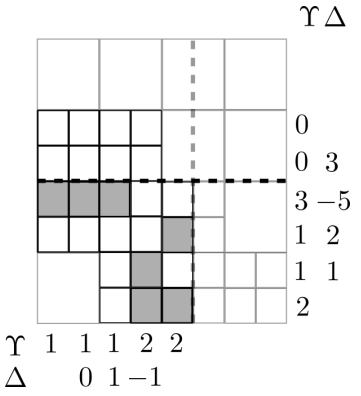
  This is used in order to control the $\mathbb{L}_1$-norm. In this work, all adaptive MHD simulations problems uses this thresholding strategy only over the variable mass density (ρ).

Recently both techniques have been compared in the hydrodynamics (DEITERDING et al., 2016; DEITERDING; DOMINGUES, 2017; DEITERDING et al., 2018) and magnetohydrodynamics (DOMINGUES et al., 2019b) contexts and the MR approach presented better results considering accuracy and CPU time.

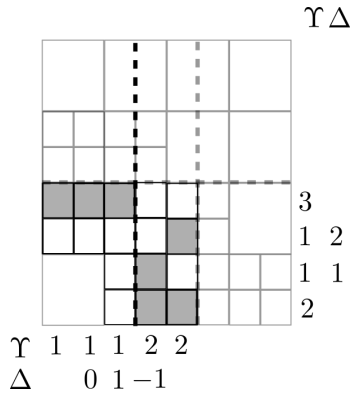
Figure 6.5 - Cell clustering algorithm. The value Υ associated with each row and column is defined as the number of flagged cells in that row or column, and $\Delta = \Upsilon_{i+1} - 2\Upsilon + \Upsilon_{i-1}$.



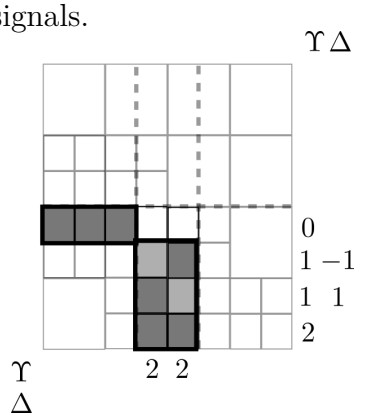
a) 1^{st} step: Mesh division where $\Upsilon = 0$.



b) 2^{nd} step: Division of the white mesh where the value Δ_n presents the highest variation between adjacent values with opposite signals.



c) Recursive execution of the 2^{nd} step over the new subdivisions.



d) Stop criteria: Percentage of flagged cells in the subdivision surpass the predefined parameter ν

Source: Author's production.

6.3 Time evolution

As the adaptive mesh is defined, the time evolution for simulations with adaptive meshes presents its own challenges regarding stability and conservation. As presented in Section 4.5, the time step parameter Δt may satisfy the CFL condition in every patch. In this context, the time evolution process, considering adaptive meshes, may be performed in two different approaches.

The simpler strategy to use a global time step that satisfy the CFL condition in every patch of every level. This value is calculated based the mesh size at the finest level, which produces the lowest values in Equation 4.60. Thus, this value satisfies

the CFL condition for the coarser cells. This formulation, used in the codes FLASH (FRYXELL et al., 2000) and CASTRO (ALMGREN et al., 2010), presents the advantages of simpler synchronisation algorithms and software frameworks, once the solution in every submesh being always in the same time instant.

The second approach, implemented in the AMROC framework (DEITERDING, 2011) discussed in Chapter 7, and consequently in the proposed MHD code, uses a refinement based time stepping strategy. This approach uses the same Courant number for every patch, independently of its refinement. Therefore, as the values Δx_ℓ varies according to the refinement level, the respective time step Δt_ℓ must variate in the same proportion, so that:

$$\frac{\Delta t_\ell}{\Delta x_\ell} = \frac{\Delta t_{\ell-1}}{\Delta x_{\ell-1}} = \frac{\Delta t_0}{\Delta x_0} \tag{6.6}$$

Alternatively, the time step for every refinement level can be obtained in function of the time step in the coarsest level as:

$$\Delta t_\ell = \frac{1}{r^\ell} \Delta t_0. \tag{6.7}$$

This refinement based time stepping causes the solution in different refinement levels to be available in different time instants, requiring synchronisation algorithms to perform the time evolution in the finest patches. In this local time stepping context, the author proposed a new strategy to perform this refinement based time stepping for high order schemes in his master thesis (MOREIRA LOPES, 2014) and improved and developed these ideas during his PhD program. The Chapter 9 contains the recent articles with that development (MOREIRA LOPES et al., 2019; MOREIRA LOPES et al., 2018b). These ideas consist in using a natural continuous extensions for Runge–Kutta methods, introduced in (ZENNARO, 1986), to perform the synchronisations required to obtain a new scheme with high order in time. In these works, the experiments were executed in the CARMEN code (ROUSSEL, 2003; DOMINGUES et al., 2008) using hidrodynamical examples.

7 COMPUTATIONAL ASPECTS

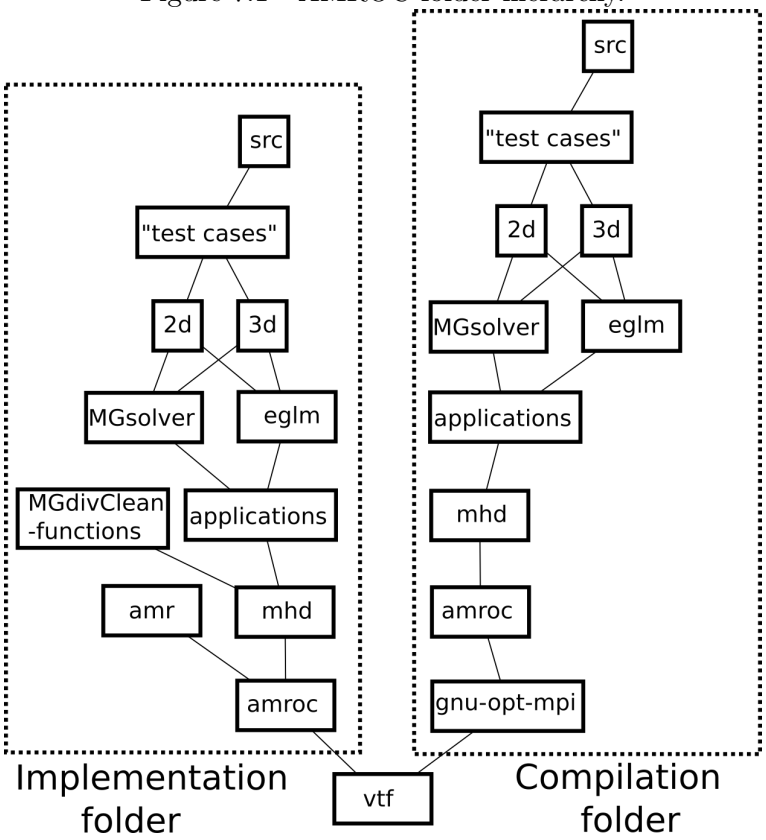
This chapter is dedicated to discuss the implementation of the methods presented in the previous chapters. These implementation are performed in the AMROC (Adaptive Mesh Refinement in Object-oriented C++) framework. This framework was conceived to support the numerical simulation of partial differential equations using adaptive methods. For that, it implements a special version of the SAMR algorithm proposed in Berger and Corella (1989) using object-oriented programming resources in the C++ language. More recently, in Domingues et al. (2019b) the framework was extended to also support the adaptive multiresolution presented in Harten (1995).

This framework, presented in Deiterding (2011), consists in about 46 thousand of lines of code in the C++ language and more 6 thousand of lines for visualisation and data conversion routines. Besides being written in the C++ language, the mathematical part of the mesh operations, such as prolongation and restriction, are performed using the FORTRAN language, which presents better performance than the C++ language for mathematical computations.

The AMROC framework is freely available in the webpage `https://bitbucket.org/deiterding/vtf/wiki/Home`, which contains the instalation, compilation and running guides. In special the AMROC MHD module running scripts already contain the commands to convert the output HDF (Hierarchical Data Format) files into binary VTK (Visualization ToolKit) files used for the data visualisation in softwares such as VisIt (CHILDS et al., 2012), available in `https://wci.llnl.gov/simulation/computer-codes/visit/downloads`, and ParaView (AHRENS et al., 2005), available in `https://www.paraview.org/download/`.

In the context of this work, the AMROC framework is divided into two main folders, the implementation and compilation folders, as presented in Figure 7.1. There are another folders that are ommited in this diagram. These are in general the folders that are not relevant for this work, such as installation folders, system folders and modules for solving another types of equations.

Figure 7.1 - AMROC folder hierarchy.



Source: Author's production.

The compilation folders are generated automatically during the installation process into two steps. The first step occur during the command:

```
autoreconf -v
```

where the script generates a *Makefile.in* file inside every subfolder of the implementation folders that are marked in the *Makefile.am* files. The second step to generate the compilation folders occur during the command:

```
./configure -C --enable-opt=yes --enable-mpi=yes &
--enable-maintainer-mode  --enable-shellnewmat HDF4_DIR=$HOME/hdf4
```

where the script generates the compilation folders based in the *Makefile.in* files generated in the first step and the *Makefile* files into the directories listed in the *configure.ac* file located inside the *vtf/amroc/mhd* folder. Once the compilation folders are built, the code can be compiled using the command:

```
make
```

inside the directory *"test case"* regarding to the problem to be simulated. After that, the compilation folder will contain the executables, some objects and libraries.

7.1 Generic SAMR solver

The folder *vtf/amroc/amr* contains the base algorithm for a numerical simulation using SAMR methods for a generic system of equations. The files contained in this folder defines the data structures and routines outside the scope of the simulated equations, such as mesh adaptation, mesh distribution per processor, boundary conditions, restriction and prolongation operators, etc.

In particular, the function *IntegrateLevel()* in the file *AMRSolver.h* calls the numerical schemes associated with the simulated equation, implemented in the base module, using the *mpass* counter. For each iteration of this counter, the scheme defined in the base is performed, then the ghost cells are updated. Considering the implemented MHD solver, this counter performs three iterations, corresponding to the first RK step, the second RK step and the divergence cleaning step, respectively.

7.2 Base MHD module

The files *EGLM2D.h* and *EGLM3D.h*, located in the *mhd* directory of the implementation folder, contains the base virtual functions to perform a generic simulation of the MHD equations. In special, these files contains the time evolution function *Step()*, called from the Generic SAMR solver, and the virtual functions called from this function. The use of virtual functions allows the definition of base functions that may be used for most of the experiments, while allowing the redefinition of these functions in the Specific MHD module, if required by the studied problem.

In general, the functions from the Base module performs numerical routines that are independent from the problem simulated, such as flux computations, limiters and divergence cleaning routines.

7.3 Specific MHD module

The file *Problem.h*, located in the directory *src*, implements functions that are particular to the studied experiment. In general, this file contains initial conditions, resistivity and gravity fields. However, if necessary for the experiment, this file may contain redefinitions for the virtual functions implemented in the base module.

7.4 Multigrid solver module

The execution of the elliptic divergence cleaning using multigrid methods, as proposed in the Sections 5.1.5 and 5.3 is implemented as a variation of the basic solver class in the file *MGdivClean.h*, located in the *mhd* directory of the implementation folder. This file is complemented with the file *MG_2Dfunctions.h* for the 2D cases, or the file *MG_3Dfunctions.h* for the 3D cases, where both files are located in the *MGdivClean-functions* directory.

The implementation of this correction requires that the file *MGdivClean.h* to be included in the end of the *Problem.h* file corresponding to the studied experiment. Then, in the correspondent *solver.in* file, the number of levels must match the value *MGLevels* defined in the end of the file *MGdivClean.h*, which contains all the parameters related to the multigrid solver.

7.5 Running test cases

After compiled, the code is able to run by executing the *run.py* script using the command:

```
./run.py N
```

where N is the number of processors to be used. The simulation parameters are configured in the *solver.in* file, which contains values to be read from different classes.

- class **SolverControl**
  - **LastTime**: final instant of the numerical simulation;
  - **Outputs**: number of instants that the code will output the solution during the simulation; alternativelly, the user may use:
  - **OutputEvery**: number of iterations between each output;
- class **AMRSolver**
  - subclass **GridHierarchy**
    - **Cells**(a): number of cells forming the base mesh in the direction of the axes x, y or z, indicated by the value $a = 1$, 2 or 3, respectively;
    - **GeomMin**(a): left boundary of the physical domain in the direction of the axes x, y or z, indicated by the value $a = 1$, 2 or

3, respectively;

  * **GeomMax**(a): right boundary of the physical domain in the direction of the axes x, y or z, indicated by the value $a = 1$, 2 or 3, respectively;
  * **PeriodicBoundary**(a): boolean value to set periodic boundary condition in the direction of the axis indicated by the value a. If not, the other options for boundary condition are set in the field "BoundaryConditions";
  * **MaxLevels**: number of refinement levels allowed to compose the adaptive mesh;

- subclass **Integrator**
  * **Scheme**: numerical scheme for the flux computations. Set 1 for the HLL flux or 2 for the HLLD flux;
  * **Limiter**: flux limiter applied during the flux computations. Set 0 for neither, 1 for Minmod, 2 for Superbee, 3 for VanLeer, 4 for MC, 5 for VanAlbada or 6 for Koren;
  * **NoClean**: boolean value which sets the usage of the parabolic-hyperbolic divergence cleaning. Set 0 to perform the correction or 1 to not;
  * **Gamma**: value of the adiabatic constant γ;
  * **CFLClean**: value of the CFL parameter during the simulation;
  * **alphap**: parameter α_p of the parabolic-hyperbolic correction;
  * **eglmClean**: numerical formulation of the divergence cleaning. Set 0 for GLM formulation or 1 for EGLM formulation;
  * **Resistivity**: choose the MHD model to be considered. Set 0 for Ideal MHD or 1 for Resistive MHD;
- subclass **BoundaryConditions**: for every boundary bellow: set {Type 0} for symmetryc , {Type 1} for slip wall, {Type 2} for inlet or {Type 3} for outlet boundaries.
  * **LeftSide**: left boundary of the x axis;
  * **RightSide**: right boundary of the x axis;
  * **BottomSide**: left boundary of the y axis;
  * **TopSide**: right boundary of the y axis;
  * **BackSide**: left boundary of the z axis;
  * **FrontSide**: right boundary of the z axis;

- subclass **Flagging**
  - ScaledGradient: sets the thresholding ϵ for the SG method;
  - MRPrediction: sets the thresholding ϵ for the MR method

7.6 Building new test cases

The implementation of new test cases in the AMROC MHD module requires several steps. The first step is to create a new folder regarding to the new test case. For sake of organisation, the new folder should be created in the respective folder according to the number of dimensions and divergence cleaning approach:

- *applications/eglm/2d*: For 2D problems using the parabolic-hyperbolic correction;
- *applications/eglm/3d*: For 3D problems using the parabolic-hyperbolic correction;
- *applications/MGsolver/2d*: For 2D problems using the Multigrid correction;
- *applications/MGsolver/3d*: For 3D problems using the Multigrid correction.

Then, the name of the added folder should be appended into the *Makefile.am* file located into the same directory the folder was added. The second step consist in adding the following files into the test case folder:

- *solver.in*: Contains the simulation parameters;
- *display_file_*.in*: Files required for the conversion of the output HDF files into the visualisation VTK files;
- *run.py*: Execution script. The variable *progname* should be set with the executable name;
- *Makefile.am*: Required for the generation of the compilation folders;
- *src/Problem.h*: Contains redefinitions of classes, such as initial conditions and resistivity field, in order to characterise each test problem;

- *src/Makefile.am*: Required for the generation of the compilation folders. Contains the executable name, required libraries and their dependences.

The third step consist in adding inside the field *AC_CONFIG_FILES* of the *configure.ac* file, a path name to a correspondent *Makefile* file in the compilation folder for every *Makefile.am* file added in this process. Then, after rebuilding the compilation folder, the new test case is able to run.

8 RESULTS: VERIFICATION

This chapter presents a verification analysis of the functionalities implemented in the AMROC's MHD solver. This analysis is performed through three rounds of tests, which aims to check the accuracy, the performance of the divergence cleaning approaches discussed in Chapter 5 with the parallel algorithms and the adaptive algorithm. Then, it is discussed a more complex configuration setup involving the Earth's magnetosphere.

The simulations performed in this work were performed in a workstation with multiple nodes with processors Intel Xeon 2.20 GHz with 12 cores each.

Some other results were published in the article (MOREIRA LOPES et al., 2018a) presented in Annex C, namely:

- MOREIRA LOPES, M.; DEITERDING, R.; GOMES, A.; MENDES, O.; DOMINGUES, M. An ideal compressible magnetohydrodynamic solver with parallel block-structured adaptive mesh refinement. **Computers and Fluids**, v. 173, p. 293-298, 2018.

Moreover, other new results are expected to be published in the article (DOMINGUES et al., 2019b) under minor revision.

8.1 Accuracy experiments

The first round of tests aims for to check the accuracy of the 2D and 3D MHD solvers. For that, three initial conditions are tested. The first one is a simpler problem with an exact solution, which the 2D and 3D solutions are compared. The second test compares the solution obtained by the AMROC's MHD solver with the results obtained by the CARMEN code (ROUSSEL, 2003), which presents a MHD module tested and validated in Gomes (2017). The third test problem is used to present a discussion about the effects concerning the choice of different limiters. This test case is also used in the performance verification in Section 8.2.

Riemann type initial condition: 1 direction

This test is described in Domingues et al. (2013) as an experiment applied in early stages of the development of a MHD code. Its purpose is to verify the accuracy of the numerical schemes, once this problem has an exact solution.

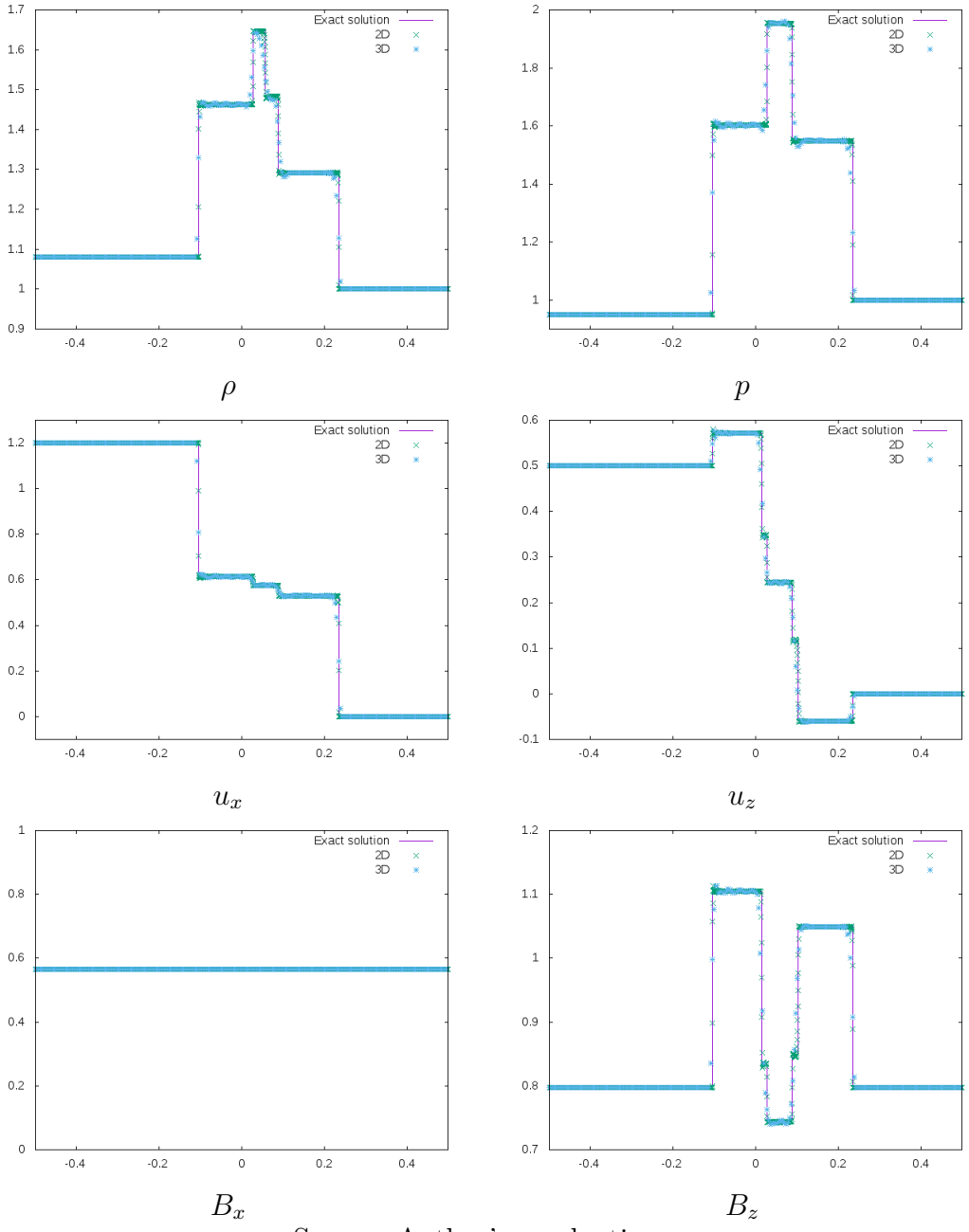
The concept of this test consist in simulating an initial configuration that resembles a Riemann problem, as defined in Section 4.2. In this case, the domain is divided into two states, left and right, at $x = 0$. The initial values for these states are given in function of its x coordinate as:

$$\mathbf{q}^0(x \leq 0) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} 1.08 \\ 0.95 \\ 1.2 \\ 0.01 \\ 0.5 \\ \frac{1}{\sqrt{\pi}} \\ \frac{1.8}{\sqrt{\pi}} \\ \frac{2}{\sqrt{2\pi}} \end{bmatrix} \qquad \mathbf{q}^0(x > 0) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\sqrt{\pi}} \\ \frac{2}{\sqrt{\pi}} \\ \frac{2}{\sqrt{2\pi}} \end{bmatrix} \tag{8.1}$$

The computational domain for this problem is $[-0.5, 0.5]^2$ for the 2D case and $[-0.5, 0.5]^3$ the 3D case. This problem is completed with Neumann type boundaries on the x-axis and periodic boundaries in the y and z-axes. For both cases, the problem is simulated using the adiabatic constant $\gamma = \frac{5}{3}$. The numerical fluxes are computed with the HLLD Riemann solver combined with a MC limiter. The divergence cleaning is performed using the parabolic-hyperbolic correction with the factor $\alpha_p = 0.5$. These simulations are performed under CFL condition $\sigma = 0.4$ until the final time $t_e = 0.1$ using the base mesh with 2048^2 and 512^3 cells for the 2D and 3D cases, respectively.

In Figure 8.1 is considered a cut parallel with the x-axis in these solutions, which is compared with the exact solution presented in Domingues et al. (2013). The numerical solution approximates the exact solution satisfactorily, indicating the correctness of the numerical solver.

Figure 8.1 - Riemann type IC: 1 direction. Cuts over the x-axis for the 2D and 3D solutions using meshes with 2048^2 and 512^3 cells, alongside with the exact solution.



Source: Author's production.

Riemann type initial condition: 2 directions

This test is presented in Dedner et al. (2002) as an extension of the previous test which is included a discontinuity at $y = 0$. This configures an initial state so that each quadrant has a different initial state, as presented in Table 8.1. Therefore, the problem consists in 4 Riemann problems that will influence themselves during the simulation.

The computational domain for this problem is $[-1, 1]^2$ for the 2D case and $[-1, 1]^3$ the 3D case. This computational domain is completed with Dirichlet boundary conditions. For both cases, the problem is simulated using the adiabatic constant $\gamma = \frac{5}{3}$. The numerical fluxes are computed with the HLLD Riemann solver combined with a MC limiter. The GLM formulation uses the parabolic-hyperbolic correction with the factor $\alpha_p = 0.5$. These simulations are performed under CFL condition $\sigma = 0.4$ until the final time $t_e = 0.25$.

The simulations were performed with a base mesh of 2048^2 and 512^3 cells for the 2D and 3D cases, respectively. Their results are compared with the solutions obtained by the CARMEN code (GOMES, 2017) in the Figures 8.2 and 8.3 for different variables. The implemented code (AMROC) presented very similar results to the validated CARMEN code for both 2D and 3D cases, indicating the correctness in the implementation of the multidimensional solver.

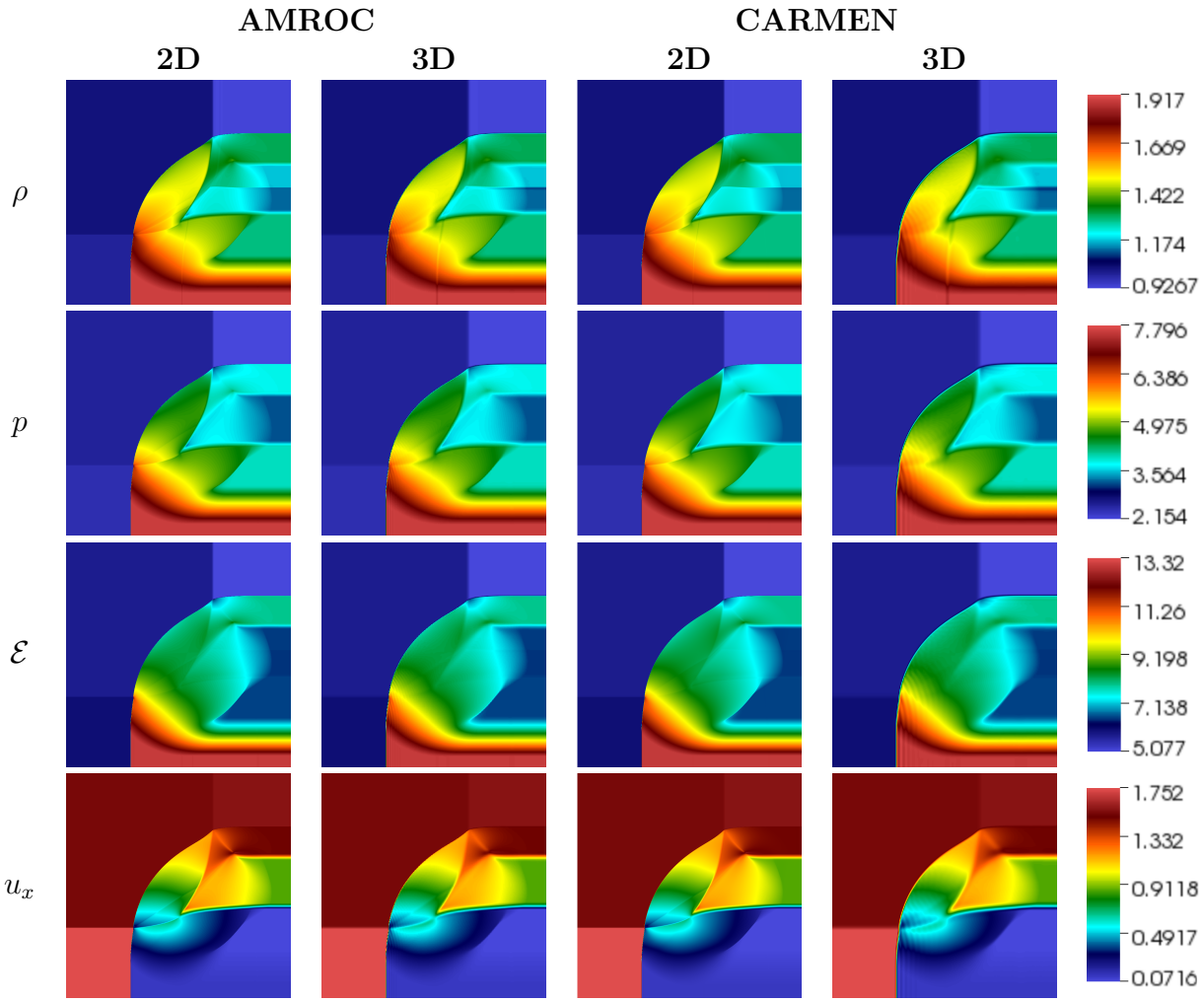
Table 8.1 - Riemann type IC: 2 directions. Initial values for the each quadrant.

Quadrants: 1^{st}) $x > 0$, $y > 0$; 2^{nd}) $x < 0$, $y > 0$;
3^{rd}) $x < 0$, $y < 0$; 4^{th}) $x > 0$, $y < 0$.

Variables	**Quadrant**			
	1^{st}	2^{nd}	3^{rd}	4^{th}
ρ	0.9308	1.0304	1.0000	3.0000
$\mathcal{E}$	5.0838	5.7813	6.0000	1.0000
ρu_x	1.4557	1.5774	1.7500	−0.7500
ρu_y	−0.4633	−1.0455	−1.0000	−0.5000
ρu_z	0.0575	−0.1016	0.0000	−0.5000
B_x	0.3501	0.3501	0.5642	−0.7500
B_y	0.9830	0.5078	0.5078	−0.5000
B_z	0.3050	0.1576	0.9830	−0.5000

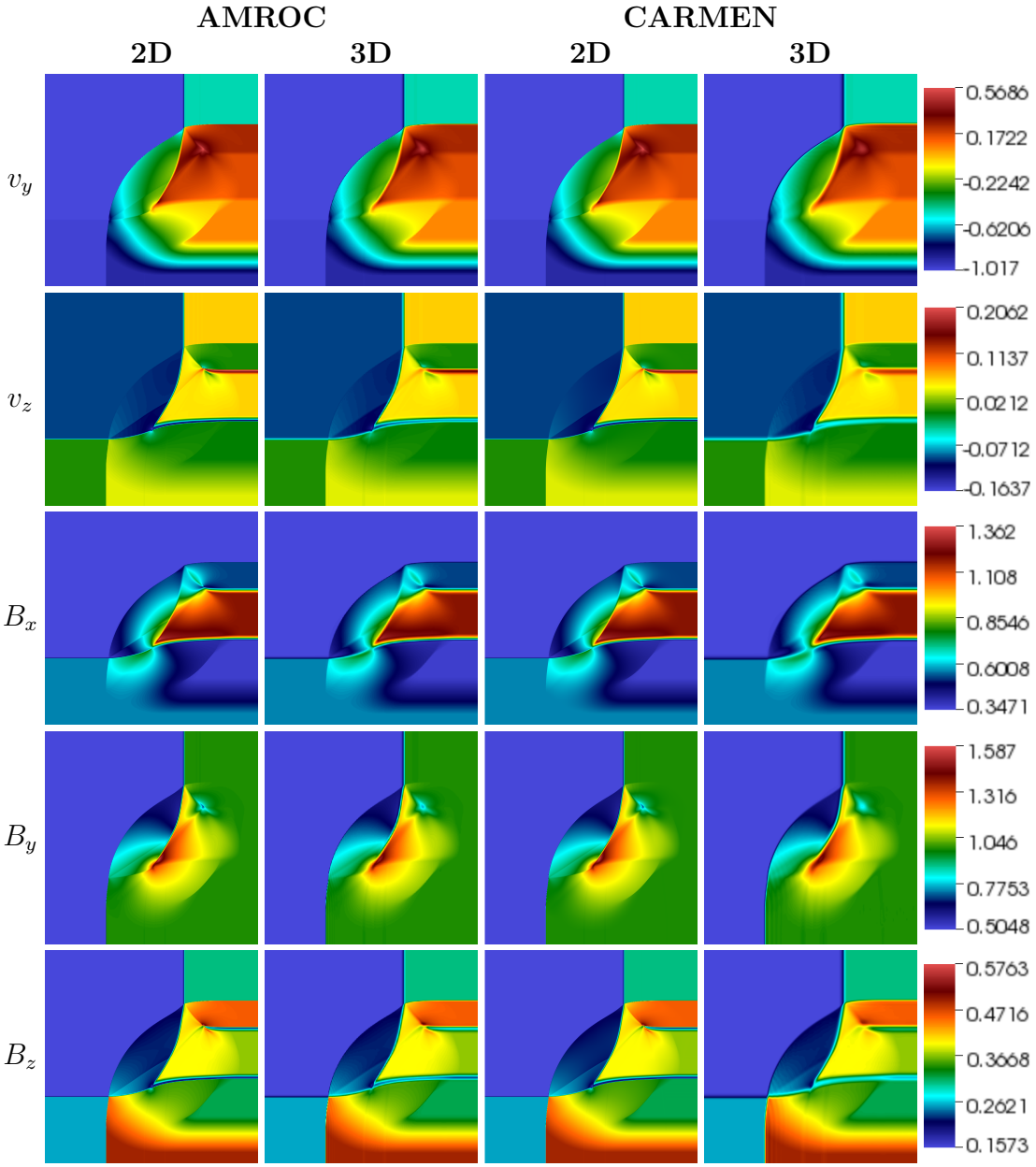
Source: Dedner et al. (2002).

Figure 8.2 - Riemann type IC: 2 directions. Results for ρ, p, $\mathcal{E}$ and u_x obtained by the 2D and 3D simulations from the AMROC and CARMEN codes.



Source: Author's production.

Figure 8.3 - Riemann type IC: 2 directions. Results for u_y, u_z, B_x, B_y and B_z obtained by the 2D and 3D simulations from the AMROC and CARMEN codes.



Source: Author's production.

Orszag-Tang vortex: 2D formulation (2D-OTV)

The Orszag-Tang vortex is a test problem introduced in Orszag and Tang (1979). It is extensively used as a verification test for ideal MHD simulations (RYU et al., 1998; DAI; WOODWARD, 1998; LONDRILLO; ZANNA, 2000). In particular, this problem deals with transitions in MHD structures and is used to test the robustness of the code regarding with the formation of MHD shocks and shock-shock interactions.

In this work, the 2D-OTV problem is used to check the effects the choice of the different limiters presented in Section 4.2.1 over the solution. In addiction, this test case is also used in the performance tests of the divergence cleaning approaches in Section 8.2.

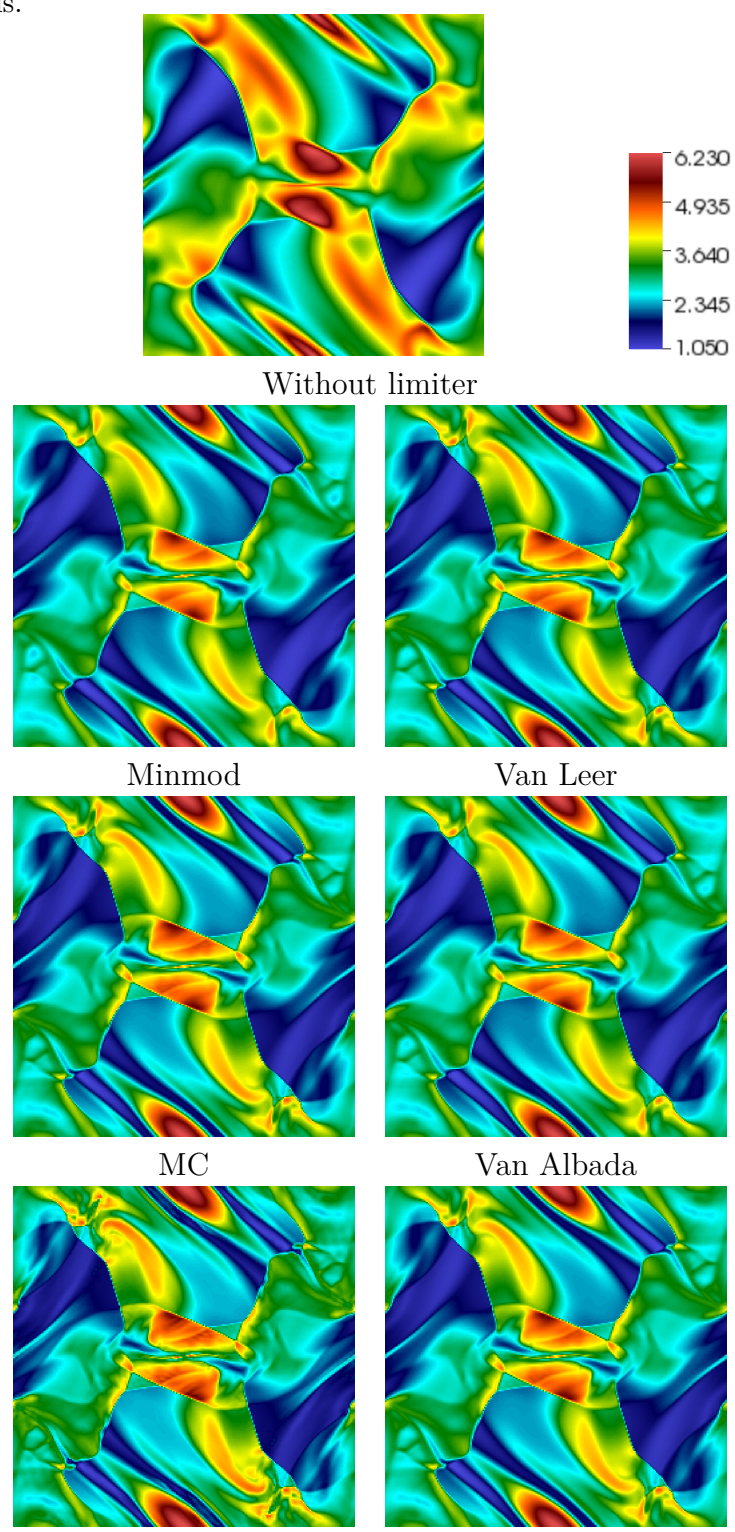
The computational domain for the 2D formulation of the OTV problem is $[0, 2\pi]^2$ with periodic boundaries. The initial condition is given in primitive variables according the adiabatic constant $\gamma = \frac{5}{3}$ as:

$$\mathbf{q}^0(x, y) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} \gamma^2 \\ \gamma \\ -\sin(y) \\ \sin(x) \\ 0 \\ -\sin(y) \\ \sin(2x) \\ 0 \end{bmatrix} \tag{8.2}$$

This problem is simulated for every flux limiter presented in Section 4.2.1 using the parabolic-hyperbolic correction with the factor $\alpha_p = 0.5$ and the HLLD Riemann solver with meshes of 256^2 and 2048^2 cells. These simulations are performed under CFL condition $\sigma = 0.4$ until the final time $t_e = \pi$.

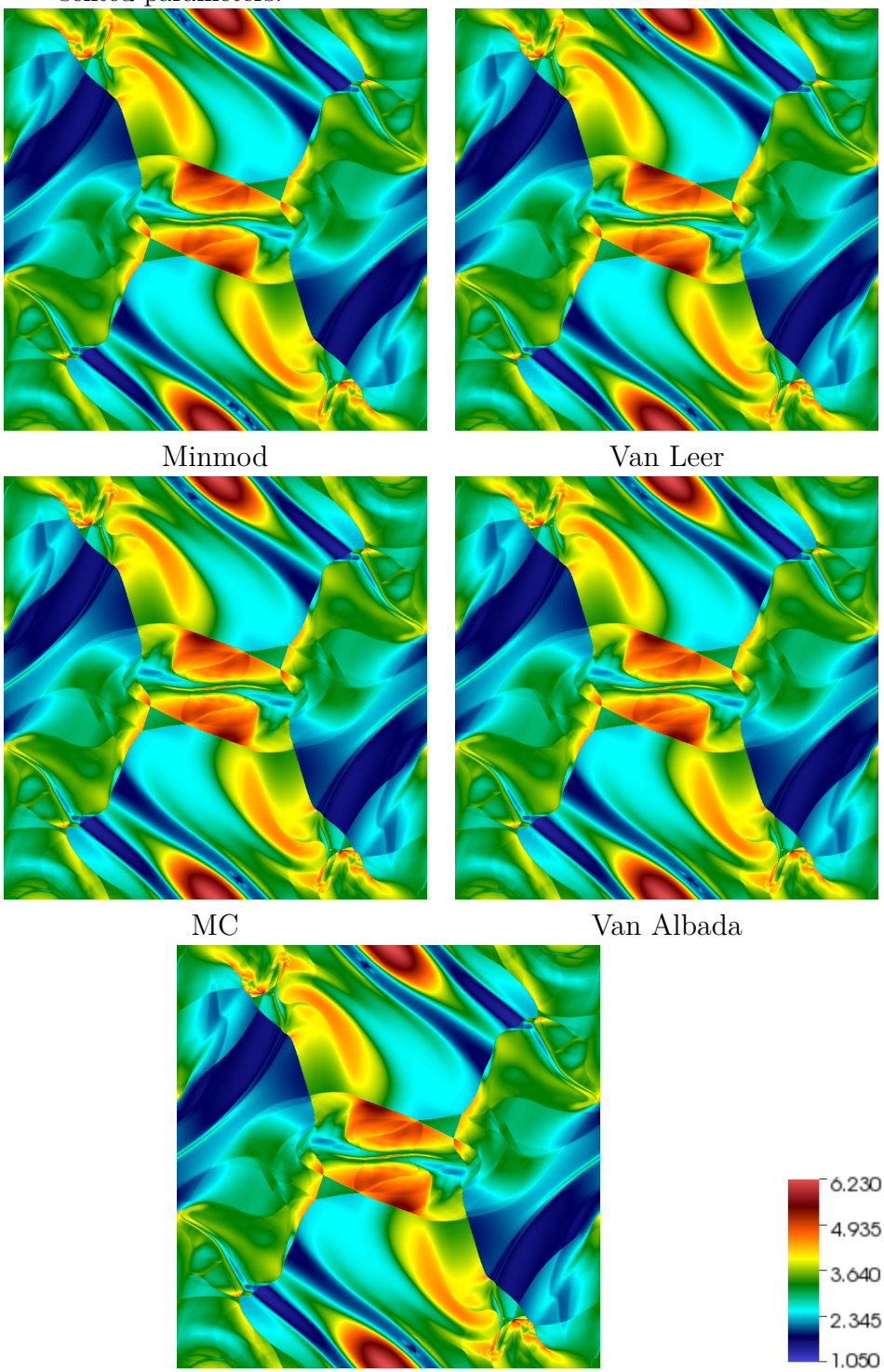
The results are presented in the Figures 8.4 and 8.5, respectively. The limiter choice is most influential in the coarser simulations. In these cases, the simulation without limiter presented the most different solution due to being a lower order method. In the other hand, the finer simulations presented similar results for every limiter. However, its choice performed a relevant role in the stability of the simulations.

Figure 8.4 - 2D-OTV: solution for ρ using different flux limiters in a base mesh with 256^2 cells.



Source: Author's production.

Figure 8.5 - 2D-OTV: solution for ρ using different flux limiters in a base mesh with 2048^2 cells. The Superbee and without limiter simulations are unstable for the presented parameters.



Minmod Van Leer

MC Van Albada

Koren

Source: Author's production.

8.2 Divergence cleaning performance experiments

The second round of tests verifies the performance of the divergence cleaning techniques presented in Chapter 5 using fully refined meshes. For that, some 2D and 3D test problems are simulated using many refinements in order to check the convergence of these solutions and the growth of the divergence in $\mathbf{B}$ when more refined meshes are applied. In addiction, these experiments are repeated using different number of processors in order to evaluate the scalability of the parallel algorithm.

These performance experiments are executed with the 2D and 3D test cases presented below and the 2D-OTV problem presented in Section 8.1. Then, the results of the discussed divergence cleaning methods are compiled, compared and discussed in Section 8.2.4.

Magnetic field loop advection (ADV)

This advection test proposed in Tóth and Odstrcil (1996) studies the effects of advection errors and numerical diffusion over $\nabla \cdot \mathbf{B}$ and is used to verify the performance of divergence cleaning techniques. The problem consists in a circular magnetic field, in equilibrium with the outside medium, that is transported through a periodic domain.

The computational domain for this problem is $[-1, 1] \times [-0.57, 0.57]$ with periodic boundaries. The initial state is given by the primitive variables according the adiabatic constant $\gamma = \frac{5}{3}$ as:

$$\mathbf{q}^0(x, y) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \frac{\sqrt{3}}{2} \\ 0.5 \\ 0 \\ \frac{\partial A_z}{\partial y} \\ -\frac{\partial A_z}{\partial x} \\ 0 \end{bmatrix}, \tag{8.3}$$

where the potential $A_z = \max\left[10^{-3}\left(0.3 - \xi\right), 0\right]$ is given in function of the distance $\xi = \sqrt{x^2 + y^2}$.

The simulations of the 2D ADV problem are performed using the HLLD Riemann solver combined with the MC limiter. The parabolic-hyperbolic correction uses the

factor $\alpha_p = 0.5$, when applied. All simulations are performed under CFL condition $\sigma = 0.4$ until the final time $t = 6.28$.

Spherical Blast Wave (BWV)

The Spherical Blast Wave problem describes the explosion of high pressure sphere contained into a uniformly magnetised medium. This experiment is applied to study the propagation of strong MHD discontinuities. According Londrillo and Zanna (2000), when poor divergence cleaning techniques are applied, the simulation of this problem may presents spurious oscillations and negative gas pressure due to the magnitude of the $\nabla \cdot \mathbf{B}$ terms increasing alongside with the background magnetic pressure.

This problem is proposed using different parameter in many works (LONDRILLO; ZANNA, 2000; BALSARA; SPICER, 1999; ZACHARY et al., 1994). In special, this work uses the formulation available in the webpage `https://www.astro.princeton.edu/~jstone/Athena/tests/blast/blast.html`, which is implemented in the Athena MHD code (STONE et al., 2018). This formulation uses the computational domain $\left[-\frac{1}{2}, \frac{1}{2}\right] \times \left[-\frac{3}{4}, \frac{3}{4}\right]$ with periodic boundaries. The solution in primitive variables for the initial state is given according the adiabatic constant $\gamma = \frac{5}{3}$ as:

$$\mathbf{q}^0(x, y) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} 1 \\ p^0 \\ 0 \\ 0 \\ 0 \\ \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix} \tag{8.4}$$

where the value p^0 is defined according if the point (x, y) is contained in the high pressure sphere with centre in the origin and radius 0.1 so that:

$$p^0 = \begin{cases} 10, \text{ if } x^2 + y^2 < 0.1^2 \text{ (High pressure sphere)} \\ 0.1, \text{ elsewhere} \end{cases} . \tag{8.5}$$

The simulations for this problem are performed using the HLLD Riemann solver combined with a MC limiter. The parabolic-hyperbolic correction uses the factor

$\alpha_p = 0.5$, when applied. All simulations are performed under CFL condition $\sigma = 0.4$ until the final time $t_e = 0.2$.

Rotor Problem (ROT)

The Rotor problem studies the propagation of strong torsional Alfvén waves. This experiment consists in a high density fluid (the rotor) rotating in high velocity inside a lighter background magnetised fluid. According to Balsara and Spicer (1999), this spinning rotor launches torsional Alfvén waves into the background fluid, causing the angular momentum of the rotor to decrease. On the other hand, the magnetic field wraps around the rotor, increasing the magnetic pressure and compressing the fluid.

The computational domain for this problem is $\left[-\frac{1}{2}, \frac{1}{2}\right]^2$ with periodic boundaries. The solution in primitive variables for the initial state is given according the adiabatic constant $\gamma = 1.4$ as:

$$\mathbf{q}^0(x,y) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} \rho^0 \\ 1 \\ u_x^0 \\ u_y^0 \\ 0 \\ \frac{5}{\sqrt{4\pi}} \\ 0 \\ 0 \end{bmatrix} \tag{8.6}$$

where the components ρ^0, u_x^0 and u_y^0 are defined according if the point (x, y) is contained inside the rotor, which is a cylinder with centre in the origin and radius 0.1, the background fluid or in a transition region. For each one of these cases, the components ρ^0, u_x^0 and u_y^0 are given in function of the distance to the origin ξ as:

$$\left(\rho^0, u_x^0, u_y^0\right) = \begin{cases} (10, -20y, -20x), \text{ if } \xi \leq 0.1 \text{ (Inside the rotor)} \\ (1, 0, 0), \text{ if } \xi > 0.115 \text{ (Background fluid)} \\ (9f + 1, -\frac{2yf}{\xi}, -\frac{2xf}{\xi}), \text{ elsewhere (Transition region)} \end{cases}, \tag{8.7}$$

with $f = \dfrac{0.115 - \xi}{0.015}$. The simulations for this problem are performed using the HLLD Riemann solver combined with a MC limiter. The parabolic-hyperbolic correction uses the factor $\alpha_p = 0.4$, when applied. All simulations are performed under CFL

condition $\sigma = 0.4$ until the final time $t_e = 0.15$.

Orszag-Tang vortex: 3D formulation (3D-OTV)

The 3D formulation of the OTV problem was initially proposed at Helzel et al. (2011), where a perturbation is added into the z axis. In this case, the computational domain became $[0, 2\pi]^3$ with periodic boundaries. The solution in primitive variables for the initial state is given according the adiabatic constant $\gamma = \frac{5}{3}$ as:

$$\mathbf{q}^0(x, y, z) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} \gamma^2 \\ \gamma \\ -\left[1 + \epsilon_p \sin(z)\right] \sin(y) \\ \left[1 + \epsilon_p \sin(z)\right] \sin(x) \\ \epsilon_p \sin(z) \\ -\sin(y) \\ \sin(2x) \\ 0 \end{bmatrix} \tag{8.8}$$

where $\epsilon_p = 0.2$ is a perturbation parameter. The simulations of this problem are performed using the HLLD Riemann solver combined with a MC limiter. The parabolic-hyperbolic correction uses the factor $\alpha_p = 0.3$, when applied. All simulations are performed under CFL condition $\sigma = 0.3$ until the final time $t_e = \pi$.

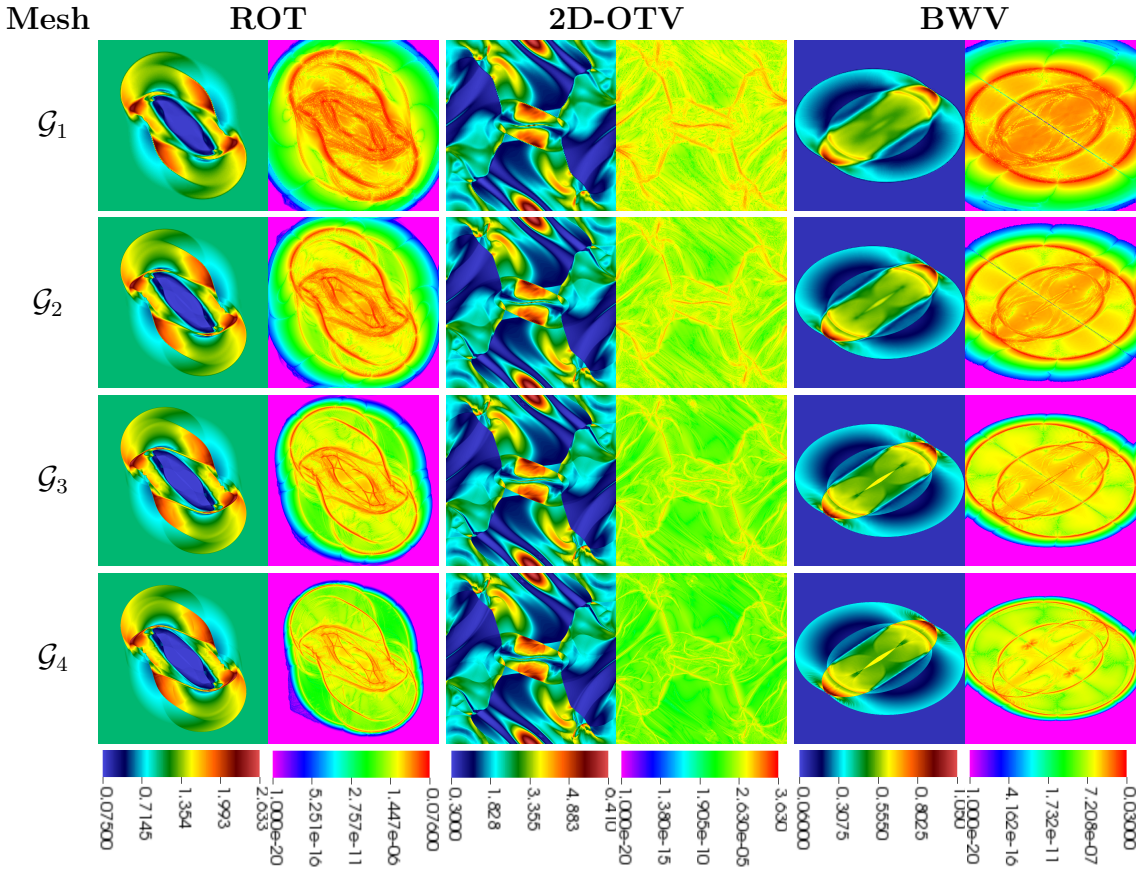
8.2.1 Results: GLM parabolic-hyperbolic correction

This section presents the results of the ROT, 2D-OTV, BWV and 3D-OTV problems obtained by the GLM-MHD model with the parabolic-hyperbolic divergence cleaning, described in Section 5.1.4. The Figure 8.6 presents the results for pressure and the parameter $\mathcal{D}_{\mathbf{B}}$, defined in Equation 5.2, for the 2D problems using several refinements, where the base meshes $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$ and $\mathcal{G}_4$ represents meshes with 256^2, 512^2, 1024^2 and 2048^2 cells for the ROT and 2D-OTV problems, respectively. For the BWV problem, these parameters represents meshes with 200×300, 400×600, 800×1200 and 1600×2400 cells.

This figure indicates a convergence of the solution when more refined meshes are applied and a reduction in the parameter $\mathcal{D}_{\mathbf{B}}$. In this ROT and BWV problems, the parameter $\mathcal{D}_{\mathbf{B}}$ presents values around machine precision zero in the pink areas. This is due to the advection of the divergence errors not reaching these regions during the simulations. The solution for the other variables are shown in Figure 8.7 for the base mesh $\mathcal{G}_4$.

Analogously, the Figure 8.8 presents the results for p and $\mathcal{D}_{\mathbf{B}}$ for the 3D-OTV problem for the base meshes $\mathcal{G}_1$ and $\mathcal{G}_2$ corresponding to meshes with 128^3 and 256^3 cells, respectively. The results for the other variables, considering the mesh $\mathcal{G}_2$, are presented in Figure 8.9.

Figure 8.6 - GLM parabolic-hyperbolic correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the 2D problems.



Source: Author's production.

In order to check the performance of the parallel algorithm, these problems alongside with the ADV problem were successively simulated using different numbers of processors, scaling by a factor of two after each simulation. The CPU time, in hours, for these simulations are presented in the Tables 8.2 and 8.4 for the 2D and 3D problems, respectively. These CPU times scaled by a factor around two until the simulations using 8 processors, but the parallel algorithm still presents a significant decrease in the CPU time for the simulations with 16 processors.

The Table 8.3 presents the CPU time obtained by the simulations with 16 processors, in function of the mesh used, for the 2D and 3D problems. These times shows the

Table 8.2 - GLM parabolic-hyperbolic correction: Elapsed time, in hours, for the parallel computations for the 2D problems, using the mesh $\mathcal{G}_4$, in function of the number of processors.

Problem	**Processors**				
	1	**2**	**4**	**8**	**16**
ADV	119.6	60.8	32.6	16.6	9.4
ROT	7.7	3.9	2.0	1.1	0.6
2D-OTV	16.5	8.4	4.2	2.2	1.5
BWV	6.6	3.3	1.8	0.9	0.6

Source: Author's production.

Table 8.3 - GLM parabolic-hyperbolic correction: Elapsed time, in hours, for the computations of the problems using 16 processors in function of the base mesh.

Problem	**Base mesh**			
	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_3$	$\mathcal{G}_4$
ADV	0.025	0.165	1.201	9.467
ROT	0.001	0.011	0.095	0.674
2D-OTV	0.002	0.022	0.159	1.513
BWV	0.001	0.011	0.085	0.649
3D-OTV	0.163	1.806	26.146	-

Source: Author's production.

Table 8.4 - GLM parabolic-hyperbolic correction: Elapsed time, in hours, for the parallel computations for the 3D-OTV problem, using the mesh $\mathcal{G}_2$, in function of the number of processors.

Problem	**Processors**		
	8	**16**	**32**
3D-OTV	3.31	1.80	1.18

Source: Author's production.

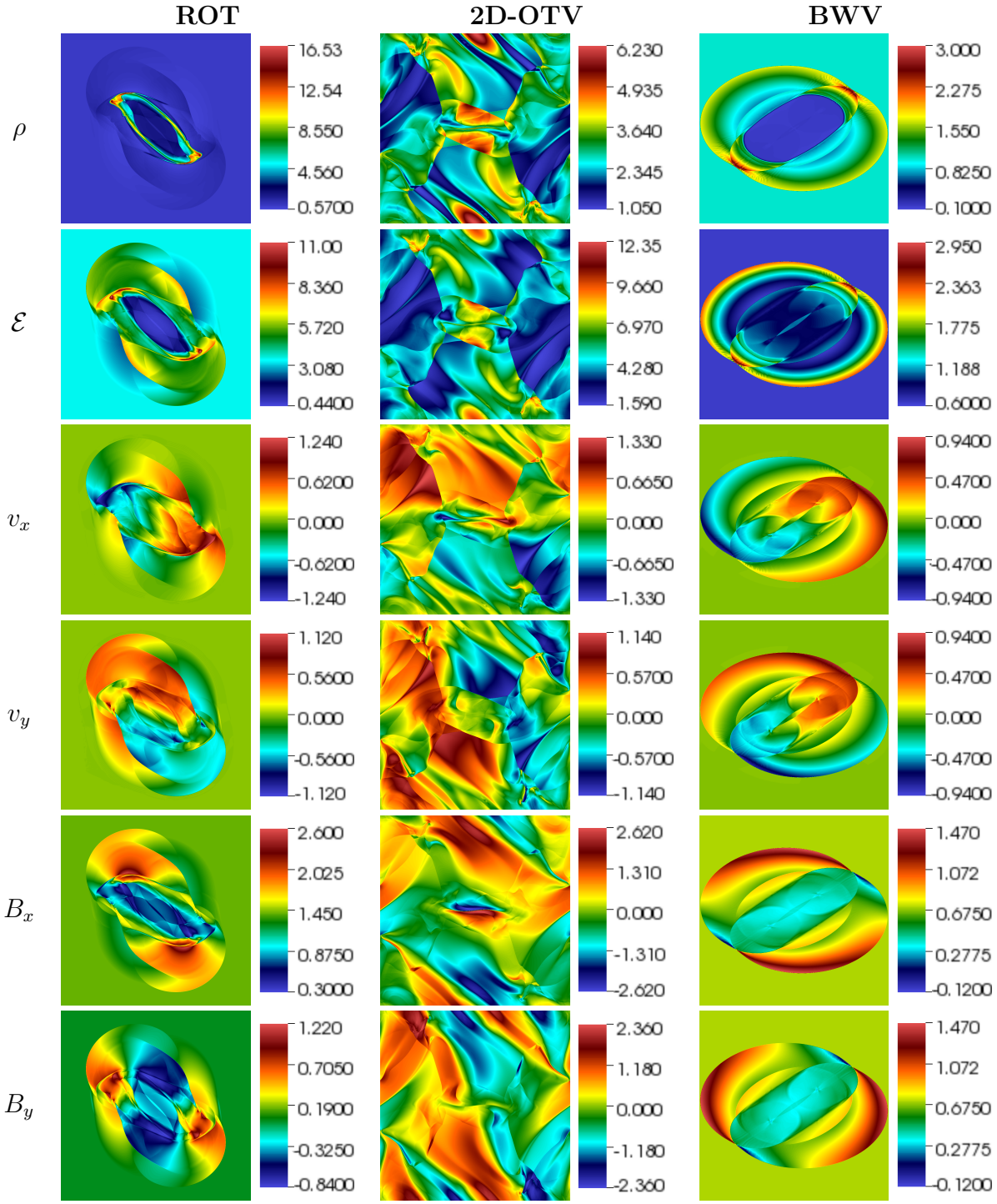
scaling of the computational cost when more refined meshes are used. In Table 8.5 is presented a breakdown of the most computationally costly tasks of the ADV simulations with different number of processors. The numerical integration, memory restart and misc tasks presented a scaling factor next to two, while the production of the output presented a slight reduction in CPU time. However this output cost is not significative in comparison with the other main tasks.

Table 8.5 - GLM parabolic-hyperbolic correction: Breakdown of the CPU time in hours spent in the main tasks computations for different numbers of processors in the ADV problem.

Task	**Processors**				
	1	**2**	**4**	**8**	**16**
Integration	108.1	54.3	27.2	14.2	7.8
Boundary	0.6	1.1	2.0	0.9	0.6
Memory restart	5.0	2.4	1.4	0.6	0.3
Misc	5.8	2.9	1.8	0.8	0.6
Output (seconds)	14.3	10.6	8.1	7.2	6.9

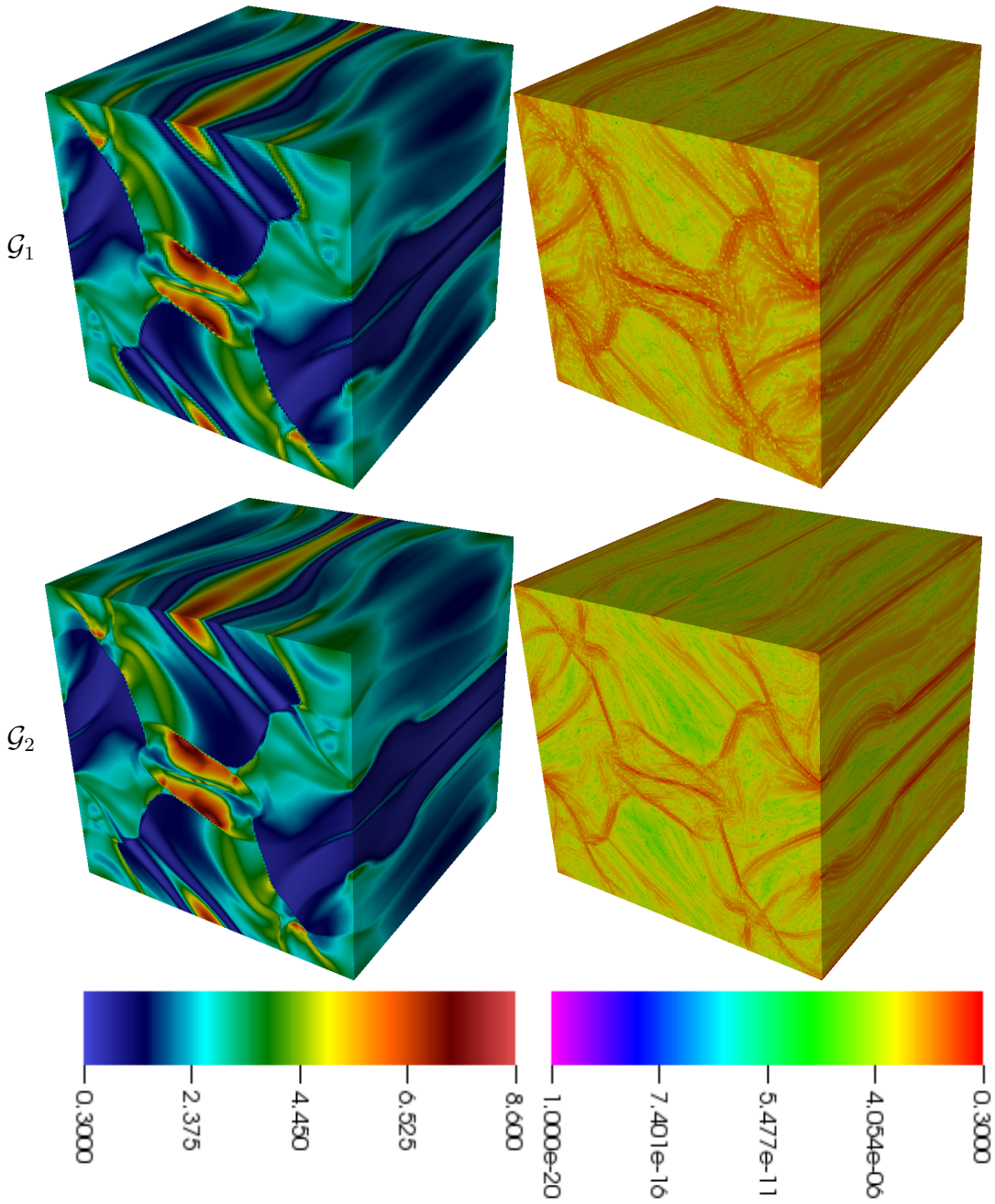
Source: Author's production.

Figure 8.7 - GLM parabolic-hyperbolic correction: Solutions obtained for the 2D problems using the mesh $\mathcal{G}_4$.



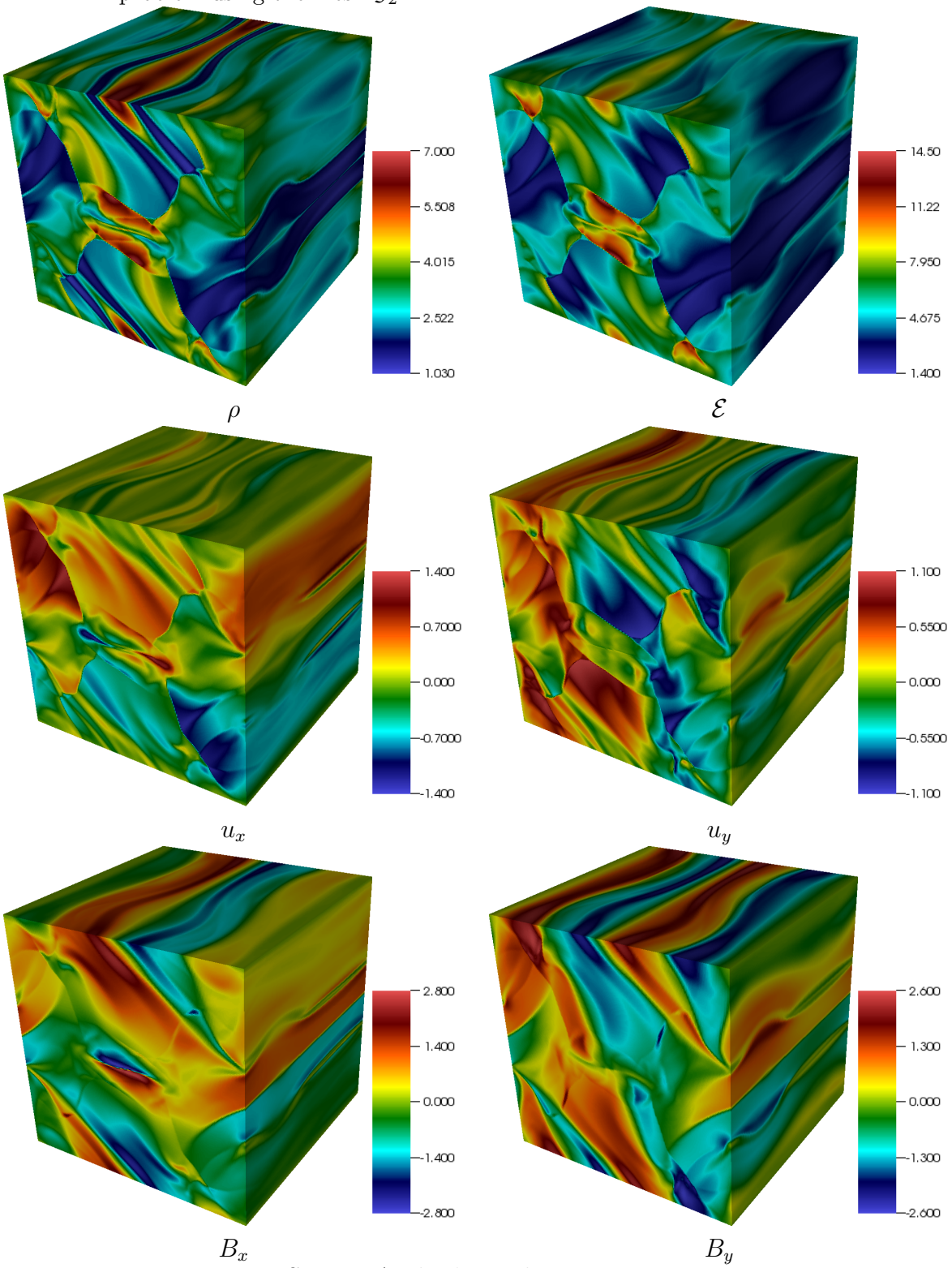
Source: Author's production.

Figure 8.8 - GLM parabolic-hyperbolic correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the 3D-OTV problem.



Source: Author's production.

Figure 8.9 - GLM parabolic-hyperbolic correction: Solutions obtained for the 3D-OTV problem using the mesh $\mathcal{G}_2$.



Source: Author's production.

Table 8.6 - GLM elliptic correction: Elapsed time, in hours, for the parallel computations for the 2D problems, using the mesh $\mathcal{G}_4$, in function of the number of processors.

Problem	**Processors**				
	1	**2**	**4**	**8**	**16**
ADV	1572.5*	548.4*	250.0*	140.8*	39.4
ROT	-	17.4	8.3	4.6	2.7
2D-OTV	82.5	37.4	16.7	9.2	5.5
BWV	37.2	17.8	8.3	4.6	2.8

*Estimated CPU time.
Source: Author's production.

8.2.2 Results: GLM elliptic correction

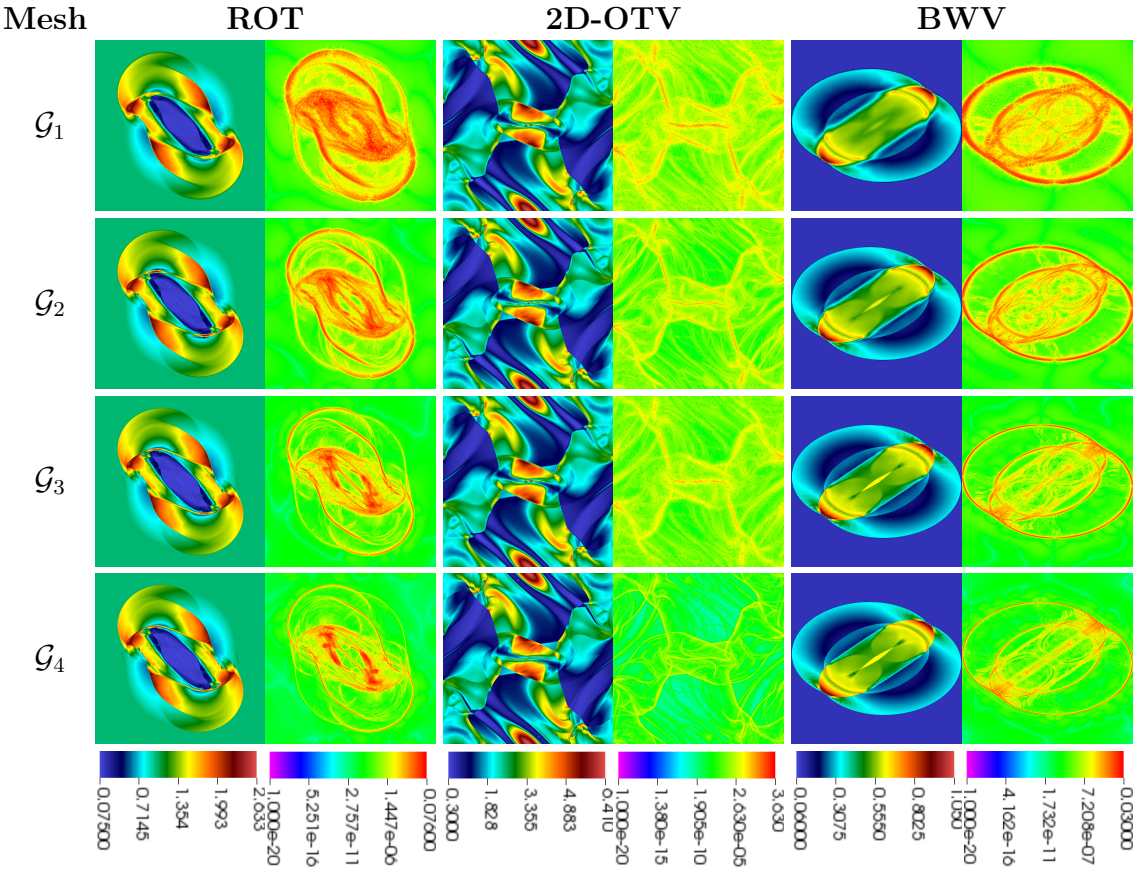
This section presents the results obtained by the elliptic correction presented in Section 5.1.5 using the multigrid solver as proposed in Section 5.3 with 3 levels and $\nu_{\text{pre}} = \nu_{\text{post}} = 1000$ Gauss-Seidel iterations. The Figures 8.10 and 8.11 presents the results for pressure and the parameter $\mathcal{D}_{\mathbf{B}}$ for the ROT, 2D-OTV, BWV and 3D-OTV problems using the same base meshes presented in Section 8.2.1.

These figures indicates the convergence of the solution p as more refined meshes are used, while the parameter $\mathcal{D}_{\mathbf{B}}$ is reduced. The solution for the other variables are presented in Figure 8.12, for the 2D problems using the mesh $\mathcal{G}_4$, and in Figure 8.13 for the 3D problem, using the base mesh $\mathcal{G}_2$.

In addition, these problems alongside with the ADV problem were successively simulated using different number of processors, scaling by a factor of two after each simulation. The CPU time, in hours, for these simulations are presented in the Tables 8.6 and 8.7 for the 2D and 3D problems, respectively. As the parabolic-hyperbolic simulations, these CPU times scaled by a factor around two until the simulations using 8 processors, but the parallel algorithm still presents a significant decrease in the CPU time for the simulations with 16 processors.

The Table 8.8 presents the CPU time obtained by the simulations with 16 processors in function of the base mesh used. These times shows the scaling of the computational cost when more refined meshes are used.

Figure 8.10 - GLM elliptic correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the 2D problems.



Source: Author's production.

Table 8.7 - GLM elliptic correction: Elapsed time, in hours, for the parallel computations for the 3D-OTV problem, using the mesh $\mathcal{G}_2$, in function of the number of processors.

Problem	**Processors**		
	8	**16**	**32**
3D-OTV	46.9	45.3	-

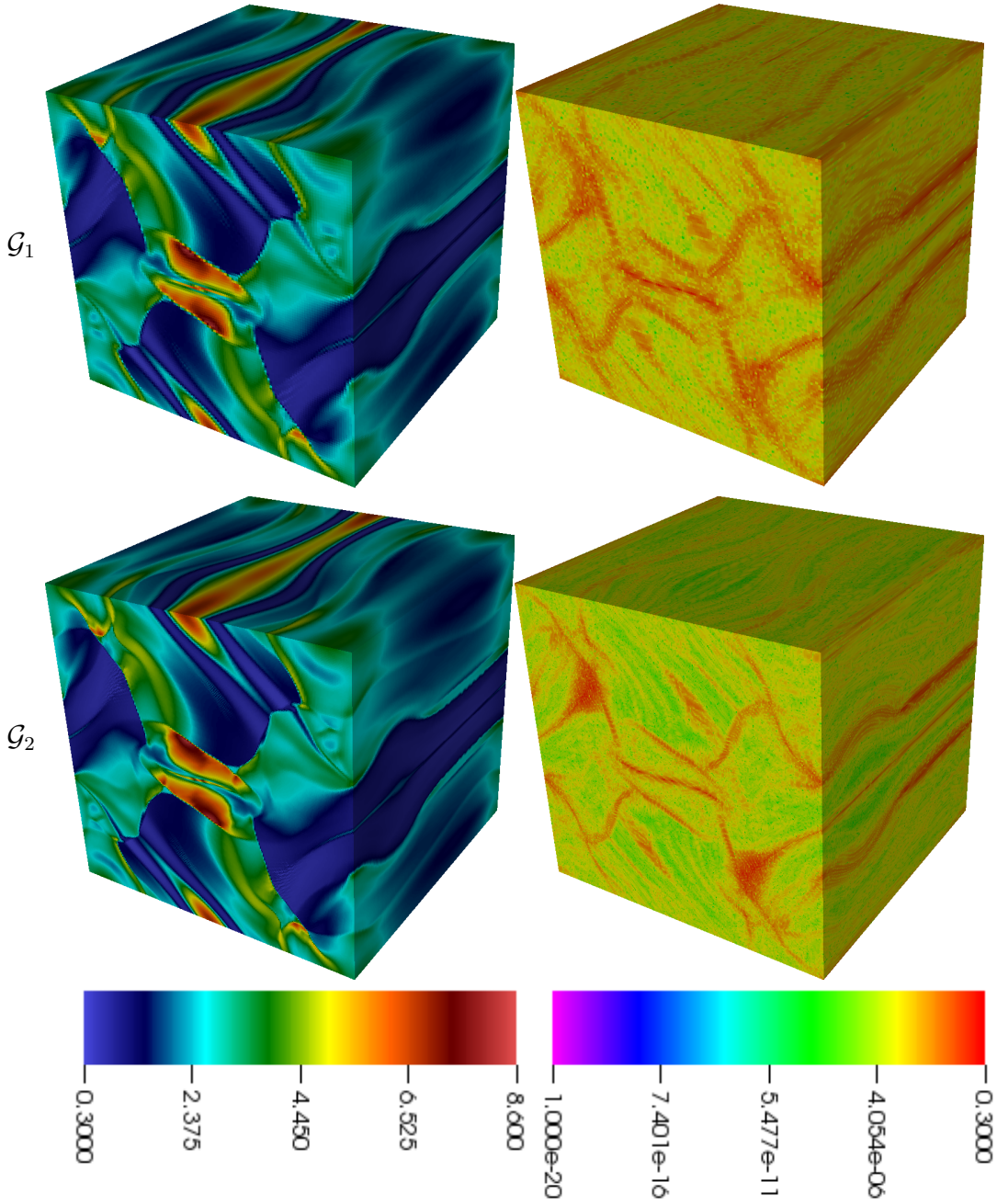
Source: Author's production.

Table 8.8 - GLM elliptic correction: Elapsed time, in hours, for the computations of the 2D and 3D problems using 16 processors in function of the base mesh used.

Problem	**Base mesh**			
	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_3$	$\mathcal{G}_4$
ADV	0.769	2.276	50.287	39.473
ROT	0.048	0.152	0.566	2.704
2D-OTV	0.104	0.320	1.121	5.546
BWV	0.061	0.160	0.572	2.8
3D-OTV	22.212	45.313	-	-

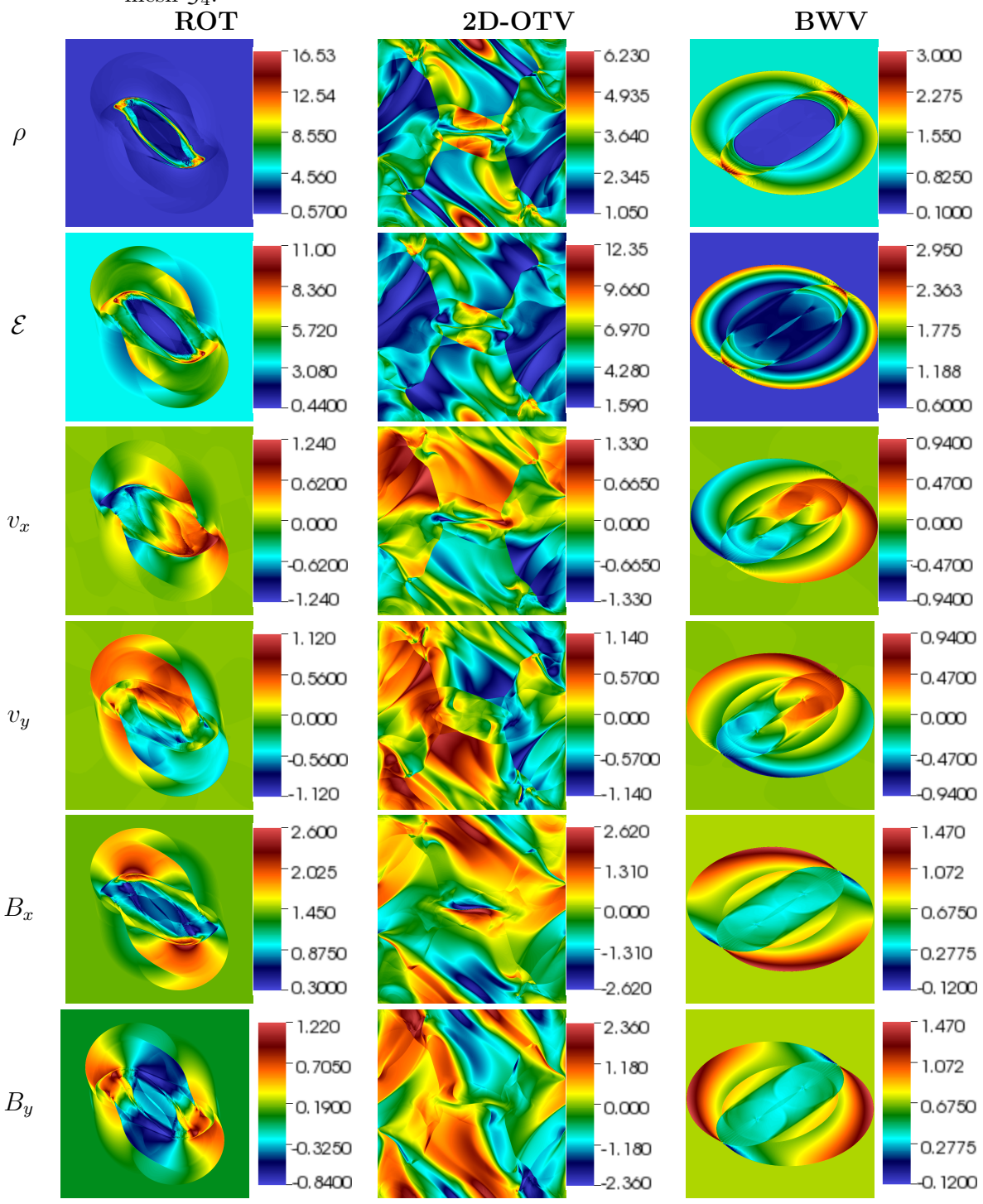
Source: Author's production.

Figure 8.11 - GLM elliptic correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the 3D-OTV problem.



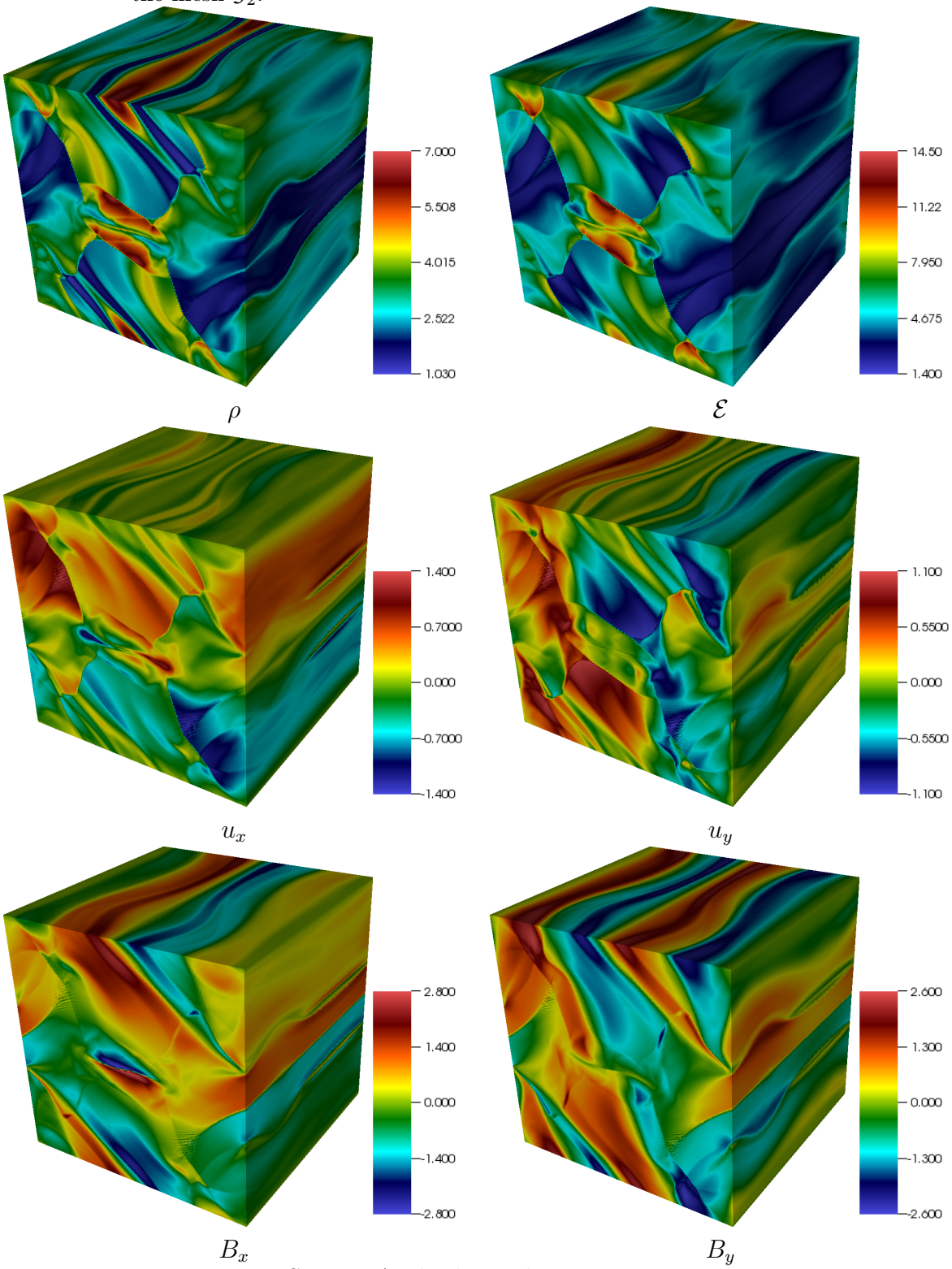
Source: Author's production.

Figure 8.12 - GLM elliptic correction: Solutions obtained for the 2D problems using the mesh $\mathcal{G}_4$.



Source: Author's production.

Figure 8.13 - GLM elliptic correction: Solutions obtained for the 3D-OTV problem using the mesh $\mathcal{G}_2$.



Source: Author's production.

8.2.3 Results: GLM triple correction

This section presents the results obtained by the proposed approach of combining the parabolic-hyperbolic correction with the elliptic correction using the MG solver as discussed in Section 5.4. These simulations are performed using 3 levels for the MG algorithm, $\nu_{\text{pre}} = \nu_{\text{post}} = 1000$ Gauss-Seidel iterations and $n_{MG} = 60$ time evolution iterations between each elliptic cleaning. The Figures 8.14 and 8.15 presents the results for pressure and the parameter $\mathcal{D}_{\mathbf{B}}$ for the ROT, 2D-OTV, BWV and 3D-OTV problems using the same base meshes presented in Section 8.2.1.

As the previous methods, these figures suggests the convergence of the solution p as more refined meshes are used, while the parameter $\mathcal{D}_{\mathbf{B}}$ is reduced. The solution for the other variables are presented in the Figure 8.16, for the 2D problems with the base mesh $\mathcal{G}_4$, and in Figure 8.17 for the 3D-OTV problem with the base mesh $\mathcal{G}_2$.

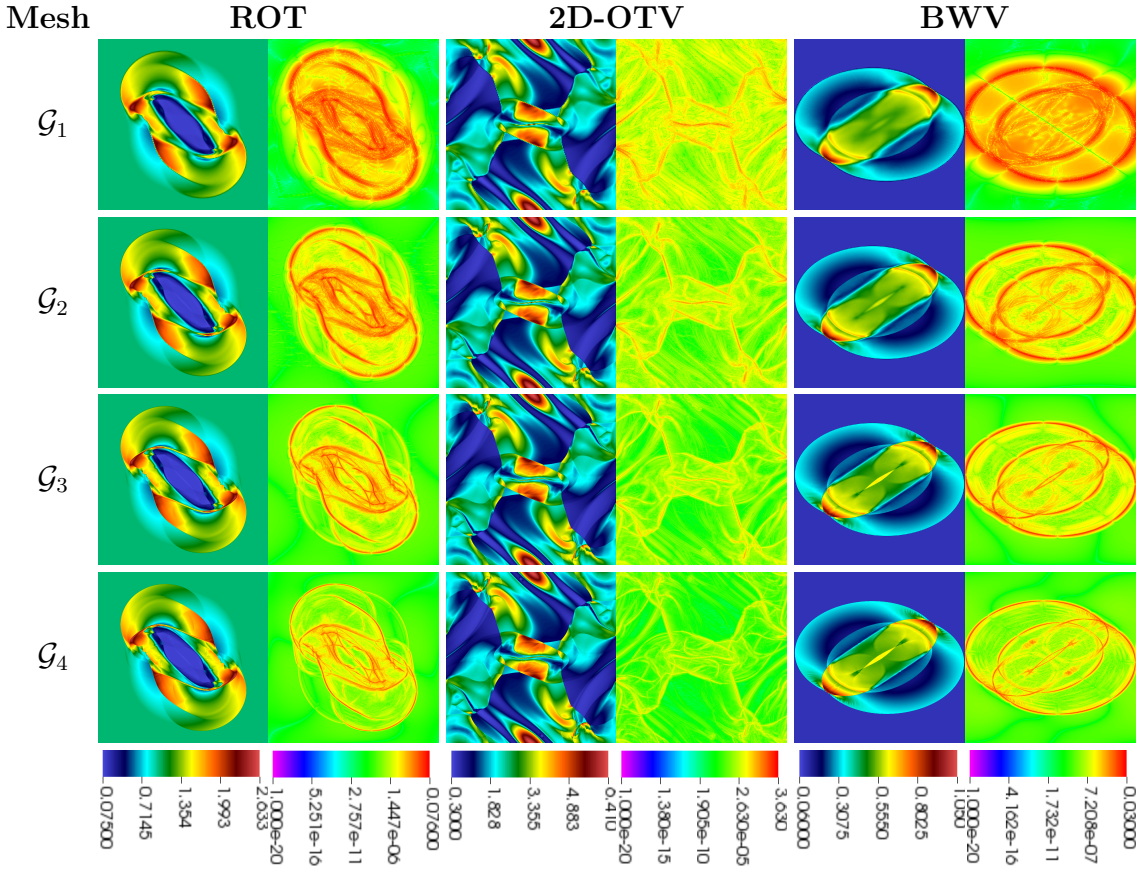
As performed for the other methods, the presented problems were simulated using different number of processors. The Tables 8.9 and 8.10 presents the CPU time for the simulations in the most refined level for the 2D and 3D problems, respectively. As the previous simulations, these CPU times scaled by a factor around two until the simulations using 8 processors, but the parallel algorithm still presents a significant decrease in the CPU time for the simulations with 16 processors. Then, in Table 8.11, the CPU times of these problems, using 16 processors, are presented in function of the base mesh $\mathcal{G}$ in order to show the scaling of the computational cost when more refined meshes are used.

Table 8.9 - GLM triple correction: Elapsed time, in hours, for the parallel computations for the 2D problems, using the mesh $\mathcal{G}_4$, in function of the number of processors.

Problem	**Processors**				
	1	**2**	**4**	**8**	**16**
ADV	157.7	84.2	40.9	21.1	12.0
ROT	8.7	4.3	2.5	1.2	0.6
OTV	31.2	15.2	7.2	3.9	2.3
BWV	9.3	4.7	-	1.2	0.7

Source: Author's production.

Figure 8.14 - GLM triple correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the ROT, 2D-OTV and BWV problems.



Source: Author's production.

Table 8.10 - GLM triple correction: Elapsed time, in hours, for the parallel computations for the 3D-OTV problem using the mesh $\mathcal{G}_2$, in function of the number of processors.

Problem	**Processors**		
	8	**16**	**32**
3D-OTV	10.3	2.9	2.2

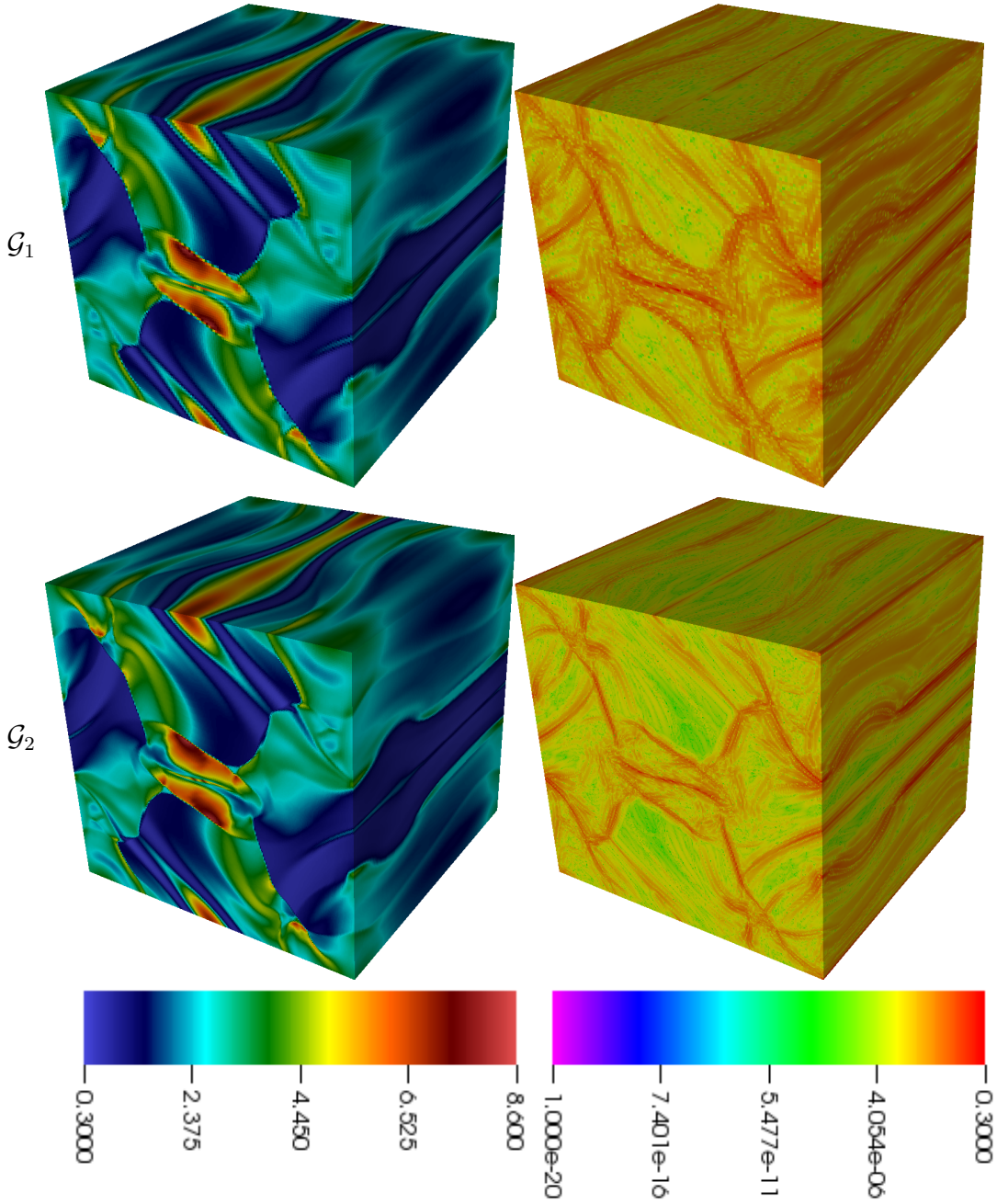
Source: Author's production.

Table 8.11 - GLM triple correction: Elapsed time, in hours, for the computations of the 2D and 3D problems using 16 processors in function of the base mesh used.

Problem	**Base mesh**			
	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_3$	$\mathcal{G}_4$
ADV	0.049	0.248	1.610	12.050
ROT	0.002	0.015	0.185	0.671
2D-OTV	0.006	0.033	0.223	2.301
BWV	0.003	0.046	0.100	0.740
3D-OTV	0.249	2.908	-	-

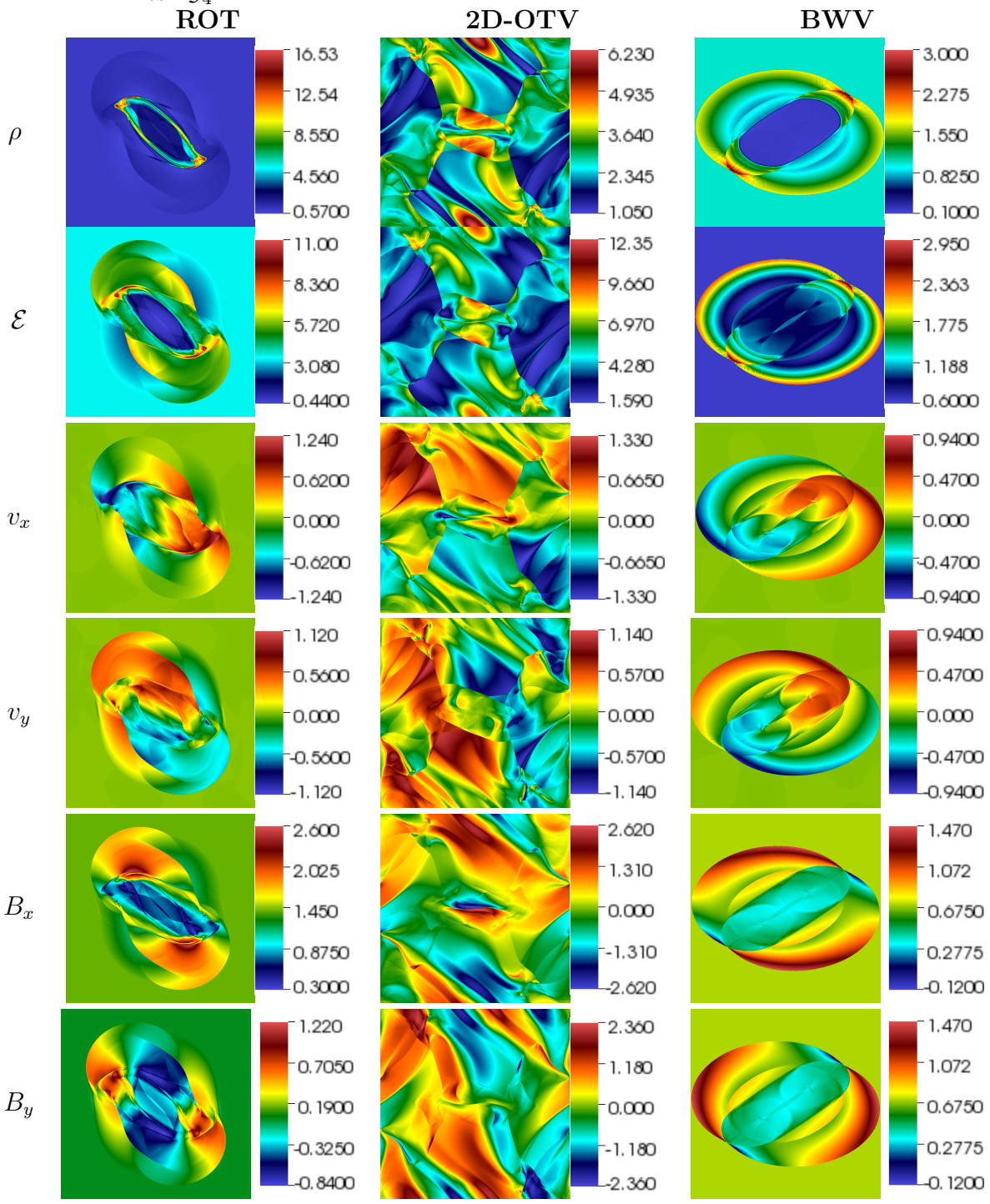
Source: Author's production.

Figure 8.15 - GLM triple correction: p and $\mathcal{D}_{\mathbf{B}}$ results for the 3D-OTV problem.



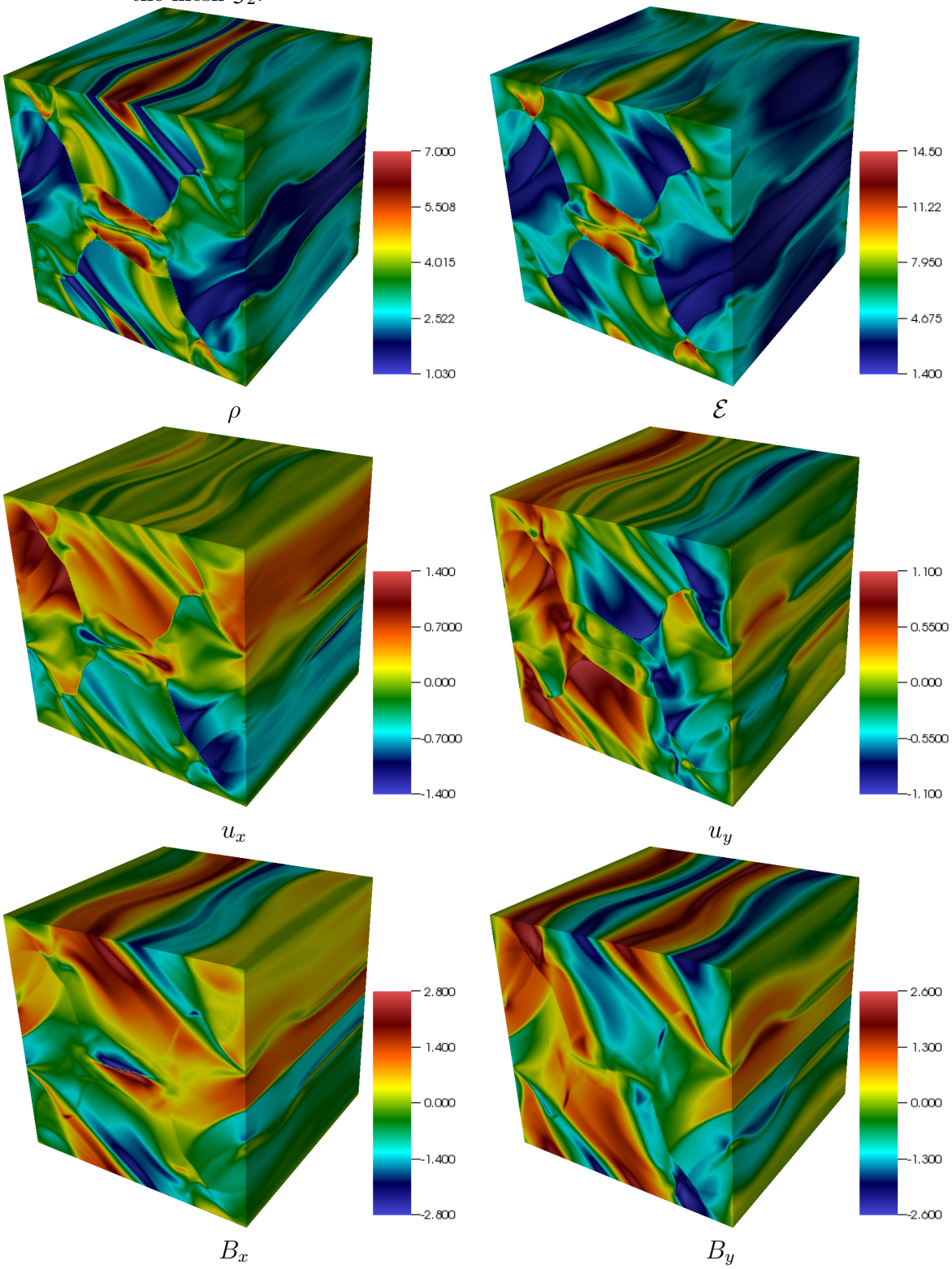
Source: Author's production.

Figure 8.16 - GLM triple correction: Solutions obtained for the 2D problems using the mesh $\mathcal{G}_4$.



Source: Author's production.

Figure 8.17 - GLM triple correction: Solutions obtained for the 3D-OTV problem using the mesh $\mathcal{G}_2$.



Source: Author's production.

8.2.4 Comparison between the GLM methods

This subsection is dedicated to compare the results obtained by the proposed triple correction with the other GLM approaches. The Figures 8.18 and 8.19 presents the results for the parameter $\mathcal{D}_{\mathbf{B}}$ obtained by each correction for the 2D and 3D problems. These simulations were performed using 16 processors and the meshes $\mathcal{G}_4$ and $\mathcal{G}_2$, respectively.

The parabolic-hyperbolic approach obtained machine zero precision in some regions of the ROT and BWV problems, that is due to the divergence errors not being transported to those regions yet. In general, the elliptic approach presented the lowest divergence errors, specially in the ADV and BWV problems. However, it presented the highest divergence peak in the ROT problem, besides presenting lower divergence globally.

The proposed triple correction presented lower divergence errors than the parabolic-hyperbolic correction, specially in the regions containing structures, such as the centre of the rotor and in the more inner parts of the blast wave. The divergence in $\mathbf{B}$ obtained by the norms $\mathbb{L}_1$ and $\mathbb{L}_\infty$, and the CPU time by each divergence cleaning approach are presented in Table 8.12 for the 2D and 3D problems. These simulations were performed using 16 processors and the meshes $\mathcal{G}_4$, for the 2D problems, and $\mathcal{G}_2$, for the 3D problems.

The results confirms the reduction in the divergence provided by the triple correction in relation to the parabolic-hyperbolic correction, while the increase in the CPU time resulted from this correction did not compromised the computational cost associated with the simulation.

In particular, the ADV problem has an exact solution to compare with the solutions obtained by the divergence cleaning approaches discussed in this work. The errors obtained for the components B_x and B_y are presented in the Table 8.13. The errors in the $\mathbb{L}_\infty$ norm were similar for every approach. However, the elliptic and triple corrections presented an $\mathbb{L}_1$ error with one order lower than the parabolic-hyperbolic approach. The reason for this higher error is presented in the Figure 8.20, which presents the results for $\|\mathbf{B}\|$ obtained for the three approaches. The result for the parabolic-hyperbolic approach presented an oscillation inside the advecting magnetic field and a background noise, which did not occur in the elliptic and triple corrections.

Table 8.12 - Divergence and CPU time, in hours, obtained by the discussed divergence cleaning methods.

	Method	$\|\nabla \cdot \mathbf{B}\|_1$	$\|\nabla \cdot \mathbf{B}\|_\infty$	t_{CPU}
ADV	Par-Hyp	2.002e-05	3.987e-03	9.467
	Elliptic	2.337e-06	4.181e-03	39.473
	Triple	5.064e-06	2.396e-03	12.050
ROT	Par-Hyp	1.947e-01	1.417e+02	0.674
	Elliptic	3.707e-01	2.396e+02	2.704
	Triple	1.842e-01	1.355e+02	0.671
2D-OTV	Par-Hyp	1.552e+00	2.243e+01	1.513
	Elliptic	5.624e-01	9.130e+00	5.546
	Triple	1.366e+00	1.856e+01	2.301
BWV	Par-Hyp	2.000e-01	6.795e+01	0.649
	Elliptic	1.051e-01	3.840e+01	2.808
	Triple	1.846e-01	6.234e+01	0.740
3D-OTV	Par-Hyp	1.336e+01	8.023e+00	1.806
	Elliptic	1.701e+01	1.520e+01	45.313
	Triple	1.383e+01	7.751e+00	2.908

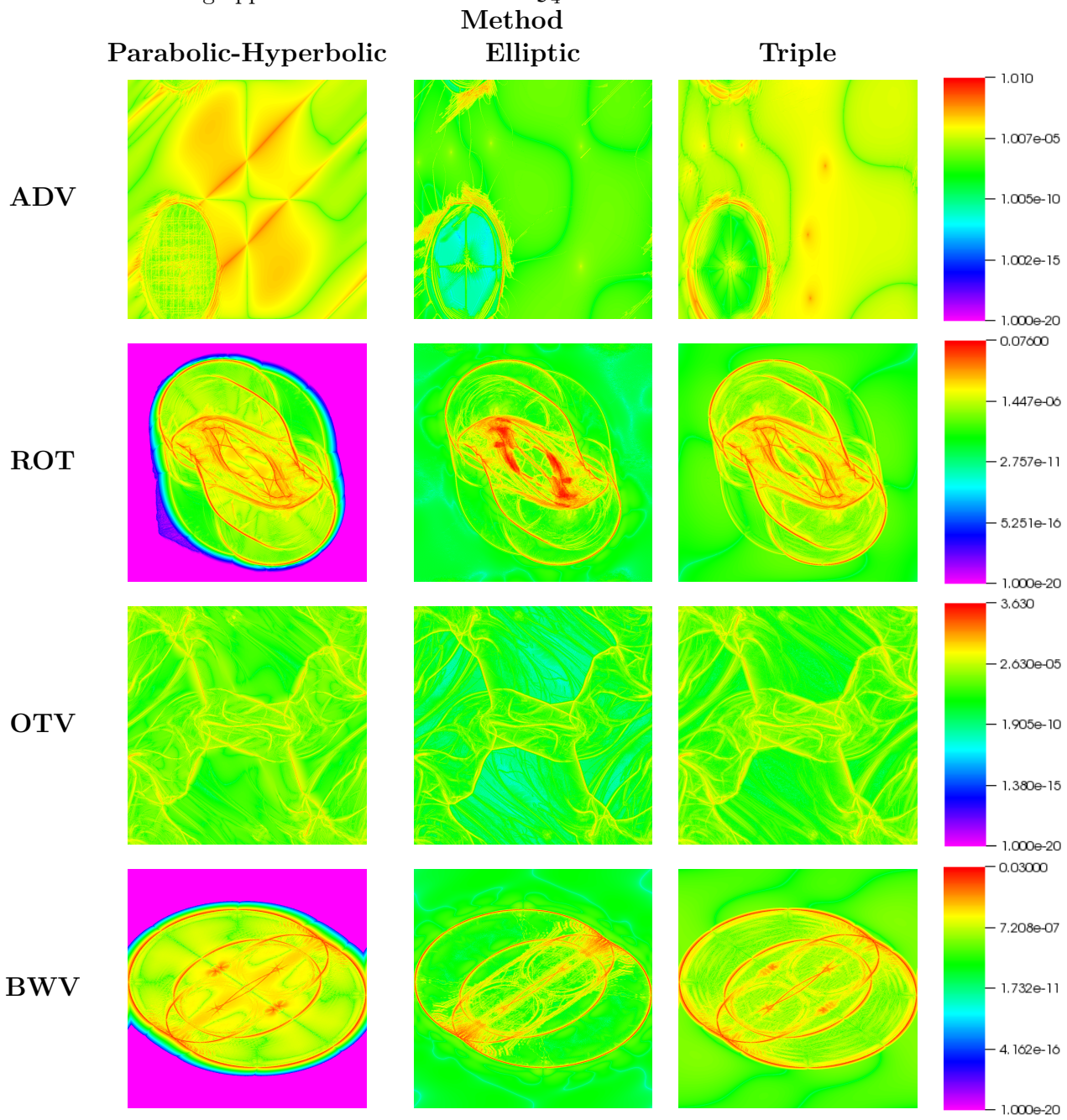
Source: Author's production.

Table 8.13 - ADV problem: Errors for the B_x and B_y components in the $\mathbb{L}_1$ and $\mathbb{L}_\infty$ norms obtained by the studied divergence cleaning approaches in the mesh $\mathcal{G}_4$ using 16 processors.

	Norm	**Method**		
		Parabolic-Hyperbolic	**Elliptic**	**Triple**
$\mathbf{B_x}$	$\mathbb{L}_1$	3.357e-05	7.463e-06	6.911e-06
	$\mathbb{L}_\infty$	1.144e-03	1.025e-03	9.940e-04
$\mathbf{B_y}$	$\mathbb{L}_1$	3.203e-05	8.332e-06	8.246e-06
	$\mathbb{L}_\infty$	9.480e-04	1.015e-03	1.076e-03

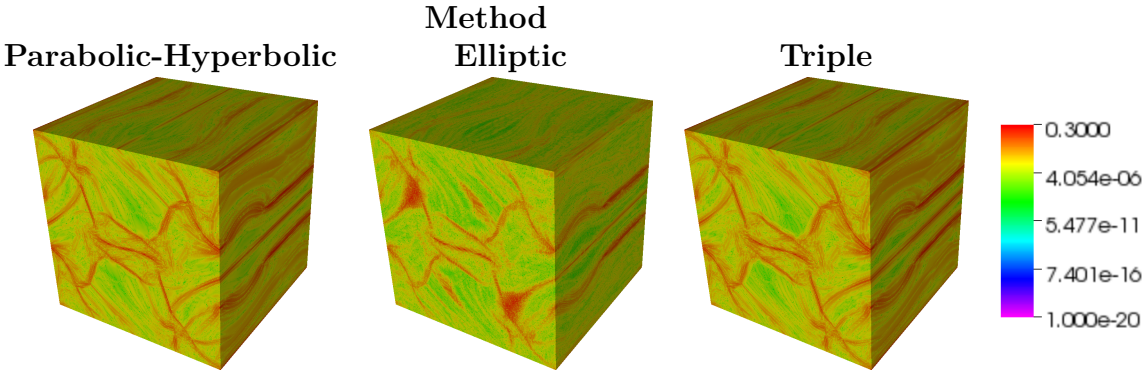
Source: Author's production.

Figure 8.18 - Parameter $\mathcal{D}_{\mathbf{B}}$, obtained for the 2D problems using the discussed divergence cleaning approaches in the base mesh $\mathcal{G}_4$.



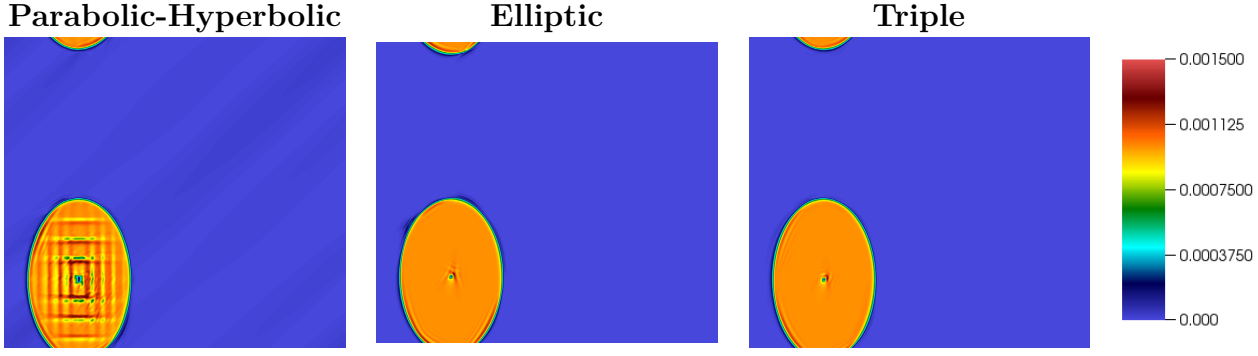
Source: Author's production.

Figure 8.19 - Parameter $\mathcal{D}_{\mathbf{B}}$, obtained for the 3D-OTV problem using the discussed divergence cleaning approaches in the base mesh $\mathcal{G}_2$.



Source: Author's production.

Figure 8.20 - $\|\mathbf{B}\|$ obtained for the base mesh $\mathcal{G}_4$ the ADV, ROT, 2D-OTV and BWV problems using the discussed divergence cleaning approaches.



Source: Author's production.

8.3 Adaptive mesh experiments

The third round of tests verifies the performance of the AMR module of the AMROC MHD solver. For that, some 2D and 3D test problems presented below, are simulated using 24 processors and adaptive meshes with several numbers of refinement levels. The results of these simulations are compared to a reference solution obtained by using a full mesh. These experiments are performed using the GLM formulation with the parabolic-hyperbolic correction.

Magnetic Reconnection (REC)

The magnetic reconnection problem is described in Jiang et al. (2012) as the reconnection of the magnetic field lines from two opposing magnetic fields, liberating a huge amount of energy. This type of phenomena is common in solar physics and is highly studied due to effects of the interaction between the Earth‘s magnetic field and the interplanetary environment. This test aims to verify the implementation of the resistive terms in MHD, once these effects are considered the responsible for the decay in the magnetic field lines leading to the reconnection.

This problem is initialised considering two different states with a small transition gap between them. Those states have magnetic fields in opposing directions and the reconnection occur inside a small region inside the transition gap, where there is a small resistivity.

The computational domain for this problem is $\left[-\frac{1}{2}, \frac{1}{2}\right] \times [-2, 2]$ with Dirichlet boundary conditions. Inside this domain, the resistivity region is defined in the subdomain $[-L_r, L_r] \times \left[-\frac{1}{5}, \frac{1}{5}\right]$, where $L_r = 0.05$. The initial condition, in primitive variables, are inspired by the one used in Jiang et al. (2012):

$$\mathbf{q}^0(x, y) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} 1 \\ 0.1 \\ 0 \\ 0 \\ 0 \\ 0 \\ B_y^0 \\ B_z^0 \end{bmatrix} \tag{8.9}$$

where the components B_y^0 and B_z^0 are given according to its corresponding state, as:

$$\left({B_y}^0, {B_z}^0\right) = \begin{cases} (1,0), & \text{if } x > L_r \text{ (Right state)} \\ (-1,0), & \text{if } x < -L_r \text{ (Left state)} \\ \left(\sin\left(\frac{\pi x}{2L_r}\right), \cos\left(\frac{\pi x}{2L_r}\right)\right), & \text{elsewhere (Transition region)} \end{cases}, \tag{8.10}$$

The value for the resistivity is defined inside the resistive subdomain in function of the parameter $\eta_0 = 6 \times 10^{-4}$ as:

$$\eta\left(x,y\right) = \begin{cases} \frac{\eta_0}{4}\left[1+\cos(10\pi x)\right]\left[1+\cos(2.5\pi y)\right], & \text{if } (x,y) \in [-L_r, L_r] \times [-0.2, 0.2] \\ 0, & \text{elsewhere} \end{cases}, \tag{8.11}$$

The simulations of this problem are performed using the adiabatic constant $\gamma = \frac{5}{3}$ and the HLLD Riemann solver combined with the MC limiter. The parabolic-hyperbolic correction uses the factor $\alpha_p = 0.4$. All simulations are performed under CFL condition $\sigma = 0.4$ until the final time $t = 2.5$.

The Table 8.14 presents the error in p using the $\mathbb{L}_1$ norm, the CPU time and the number of cells and patches used in the adaptive mesh for simulations with several refinement levels. These simulations are performed using the threshold value $\epsilon = 0.001$ for the MR refinement criteria. The number of levels used in each simulation are configured so that the most refined scale correspond to a 1024×2048 mesh. The simulation that presented the best results is obtained with $L = 2$, which presented a reduction of 49% of the CPU time while maintaining an error in the order of 5×10^{-3}. In that case the gain is roughly four times in relation to the simulation with $L = 4$. In addiction this case presented the lowest number of cells in relation with the full mesh simulation and also the lowest number of patches. Furthermore these adaptive cases required around 40% of the cells of the full mesh simulation.

The Figure 8.21 presents the solution for p, the adaptive mesh for the simulation with 4 levels and the mesh distribution among the 24 processors used. The figure containing the refinement of the adaptive mesh is interpreted so that the light blue regions of the domain are represented in the coarsest level and the yellow region is represented by the most refined scale. The mesh distribution figure is interpreted so that each coloured region of the physical domain is associated to a different processor. Considering that the more refined regions requires more computational efforts, the mesh is distributed among the processors so that it compensates the

extra workload of the finer regions by attributing smaller domains to the processors, while the processor designed for coarser meshes receives a larger domain.

The Table 8.15 is presented a breakdown of the most computationally costly tasks of the adaptive REC simulations with different number of refinement levels. In comparison with the full mesh simulations, the Boundary related tasks presented a more significative cost in the simulation scope. Also, the adaptive simulations presented the Recomposition and Regridding tasks, which presented a significant and a slightly significant cost respectively. The output production cost also presented a not significant cost in the simulation scope.

Table 8.14 - REC: Errors in p, memory consumption and CPU time obtained by using several refinement levels.

Base Mesh	**L**	**Accuracy**	**Cells**		**Patches**	**CPU Time**	
		$\mathbb{L}_1$ error ($\times 10^{-1}$)	#	%	#	(min)	%
128×256	4	0.231	928,932	44	852	27.7	29
256×512	3	0.226	895,292	42	809	48.1	51
512×1024	2	0.050	856,968	40	466	48.3	51

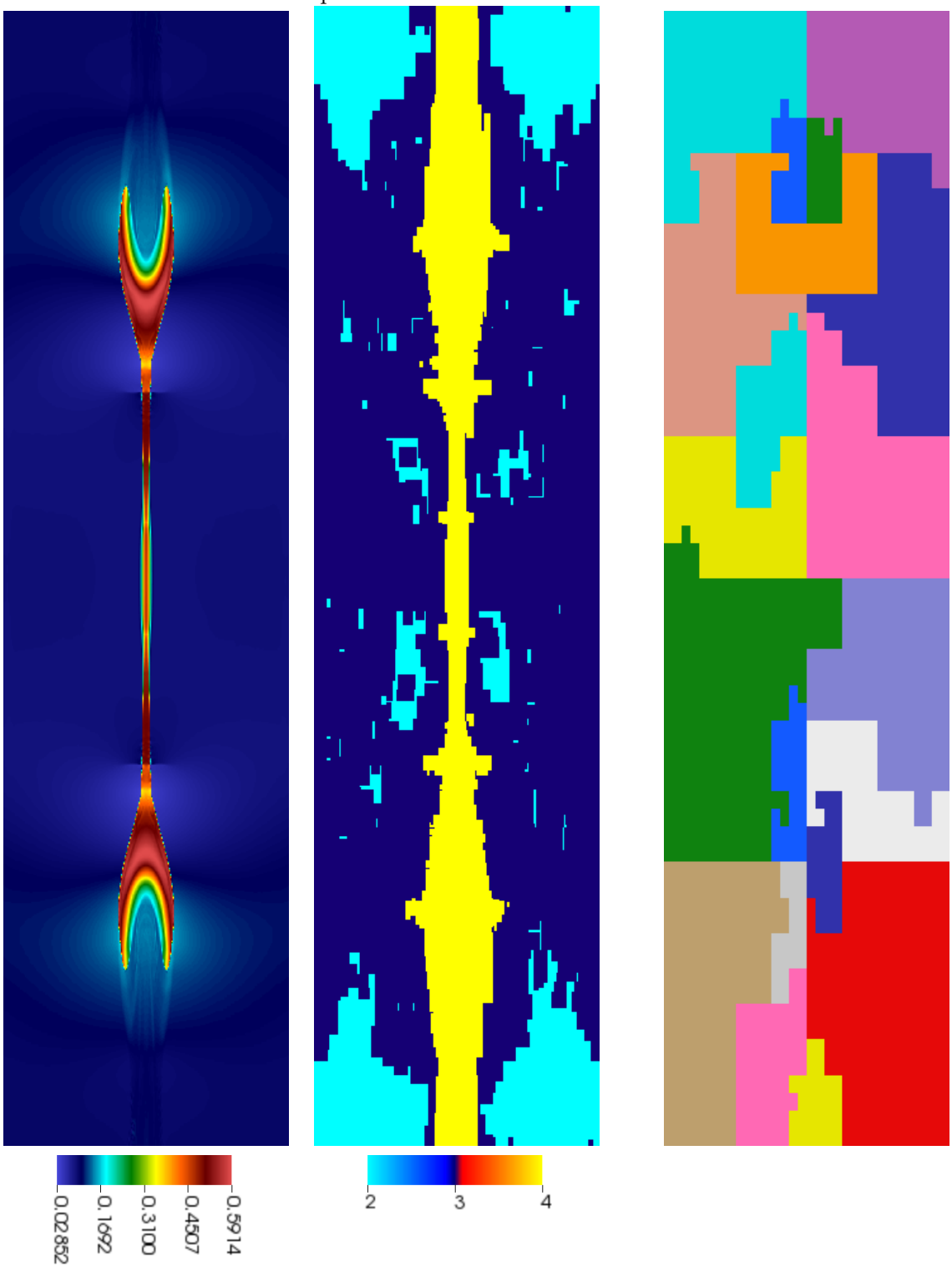
Source: Author's production.

Table 8.15 - REC: Breakdown of the CPU time in % spent in the main tasks computations for different numbers of refinement levels using $\epsilon = 0.001$.

Task	**Levels**			
	1	**2**	**3**	**4**
Integration	73.7	32.1	25.8	44.3
Boundary	14.2	29.4	20.4	30.2
Memory restart	2.5	1.2	0.3	0.3
Recomposition	-	17.2	39.0	13.7
Regridding	-	0.8	0.7	1.2
Misc	9.2	12.4	7.0	7.0
Output	0.07	0.22	0.19	0.08

Source: Author's production.

Figure 8.21 - REC: Solution for p, the adaptive mesh and mesh distribution per processors. This simulation was performed using 4 refinement levels with threshold value $\epsilon = 0.001$ and 24 processors.



p Adaptive mesh Mesh distribution per processor

Source: Author's production.

Kelvin-Helmholtz instability (KHI)

The Kelvin-Helmholtz (KHI) instability is a phenomena which occur in single continuous fluids with a velocity shear. It also occurs in the interface of two fluids with different velocities. In the context of space sciences, the KHI instability appears in the solar corona, the ionosphere and astrophysical objects.

The simulations of the KHI case, as presented in Dedner et al. (2002), are performed inside the computational domain $[0,1]\times[-1,1]$ with periodic boundaries. The initial condition, in primitive variables, is given according the adiabatic constant $\gamma = 1.4$ as:

$$\mathbf{q}^0(x,y) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} 1 \\ 50 \\ 5\left[\tanh\left[20(y+0.5)\right] - \tanh\left(20(y-0.5)\right) - 1\right] \\ 0.25\sin\left(2\pi x\right)\left(e^{-100(y+0.5)^2} - e^{-100(y-0.5)^2}\right) \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{8.12}$$

In this work, the simulations of the KHI problem are performed using the HLLD Riemann solver combined with a MC limiter. The parabolic-hyperbolic correction uses the factor $\alpha_p = 0.5$. All simulations are performed under CFL condition $\sigma = 0.4$ until the final time $t = 0.5$.

The Table 8.16 presents the error in p using the $\mathbb{L}_1$ norm, the CPU time and the number of cells and patches used in the adaptive mesh for simulations with several refinement levels. These simulations are performed using the threshold value $\epsilon = 0.001$ for the MR refinement criteria. The number of levels used in each simulation are configured so that the most refined scale correspond to a 1024×2048 mesh. The simulation with $L = 2$ and 3 obtained results with similar gain, which the first simulation has around half of the error produced by the second, however it presented roughly the double of the CPU time with less than the double of cells. In general, the precision is around ten times the chosen ϵ.

The Figure 8.22 presents the solution for p, the adaptive mesh for the simulation with 4 levels and the mesh distribution among the 24 processors used. The figure with the refinement of the adaptive mesh is interpreted so that the blue regions of the domain

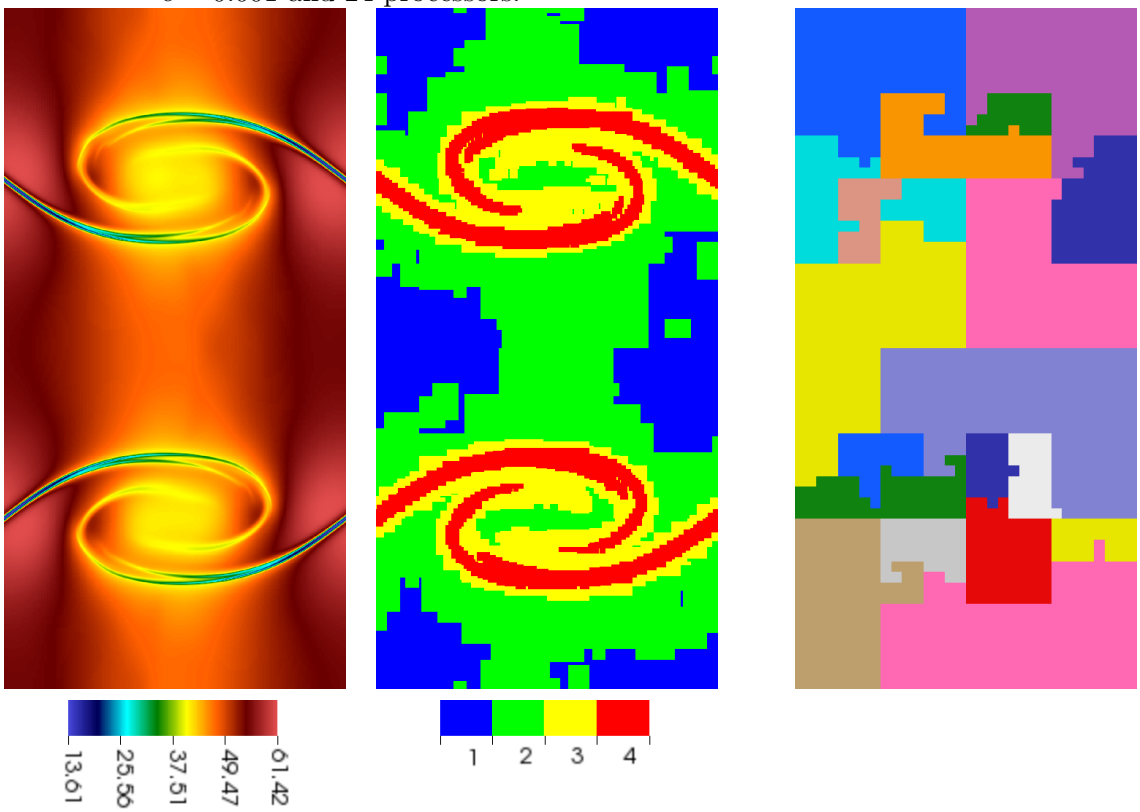
are represented in the coarsest level and the red region is represented by the most refined scale. As the previous case, the mesh distribution figure is interpreted so that each coloured region of the physical domain is associated to a different processor. The regions containing the centre of the instabilities required the most computational efforts due to the mesh adaptation, and for this reason it was attributed smaller domains to the processors. Furthermore, the expected physical behaviour of the instability was successfully achieved with a roughly desired symmetry.

Table 8.16 - KHI: Errors in p, memory consumption and CPU time obtained by using several refinement levels with $\epsilon = 0.001$.

Base Mesh	**L**	**Accuracy**	**Cells**		**Patches**	**CPU Time**	
		$\mathbb{L}^1$ error ($\times 10^{-1}$)	#	%	#	(min)	%
128×256	4	1.046	536,736	25	935	11.0	11
256×512	3	0.940	530,868	25	855	10.3	10
512×1024	2	0.514	793,776	37	695	20.2	21

Source: Author's production.

Figure 8.22 - Results for p, the adaptive mesh and mesh distribution per processors. This simulation was performed using 4 refinement levels with threshold value $\epsilon = 0.001$ and 24 processors.



p Mesh refinement Mesh distribution per processor

Source: Author's production.

Shock-Cloud Iteration (SCI)

The Shock-Cloud iteration is a test case presented in Tóth et al. (2012) used to check the performance of the numerical scheme when dealing with super-fast flows. It describes the disruption of a high-density magnetic cloud by a strong shock wave. For that, is considered an advancing plasma which causes a shock with a stationary state containing a high density cloud.

The simulations of the SCI problem are performed inside the computational domain $[0,1]^3$ with outlet boundaries. The solution, in primitive variables, for the initial state of the advancing plasma region, delimited by $x < 0.05$, as:

$$\mathbf{q}^0(x < 0.05, y, z) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} 3.86859 \\ 167.345 \\ 11.2536 \\ 0 \\ 0 \\ 0 \\ 2.1826182 \\ -2.1826182 \end{bmatrix} \tag{8.13}$$

The initial configuration for the stationary state is given by:

$$\mathbf{q}^0(x > 0.05, y, z) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} \rho^0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.56418958 \\ 0.56418958 \end{bmatrix} \tag{8.14}$$

where the value ρ^0 is defined according if the point (x, y, z) is contained in the high density cloud with centre in $(0.25, 0.5, 0.5)$ and radius 0.15:

$$\rho^0 = \begin{cases} 10, \text{ if } (x-0.25)^2 + (y-0.5)^2 + (z-0.5)^2 < 0.15^2 \text{ (High density cloud)} \\ 1, \text{ Elsewhere} \end{cases}. \tag{8.15}$$

In this work, the simulations of the SCI problem are performed using the parameter $\gamma = \frac{5}{3}$ and the HLLD Riemann solver combined with the MC limiter. The GLM formulation uses the factor $\alpha_p = 0.4$. All simulations are performed under CFL condition $\sigma = 0.4$ until the final time $t = 0.06$.

This problem was simulated using different sizes of base meshes. In each one of these cases, the AMR algorithm uses a number L of extra refinement levels so that the finest level allowed is the correspondent of a mesh of 512^3 cells. These AMR simulations are performed with a threshold value $\epsilon = 0.001$.

The $\mathbb{L}^1$ norm error obtained from these simulations in relation to a reference solution from a full 512^3 mesh, the CPU time and the number of cells and patches for each case are presented in Table 8.17. These experiments present a slight change in the number of cells in their adaptive meshes. However, the CPU time is the largest in the case $L = 2$. This may be related with the memory access required for the 3D simulations. The results with $L = 3$ presented the best gain due to besides having the best precision, it presented a better CPU time than the case with $L = 2$.

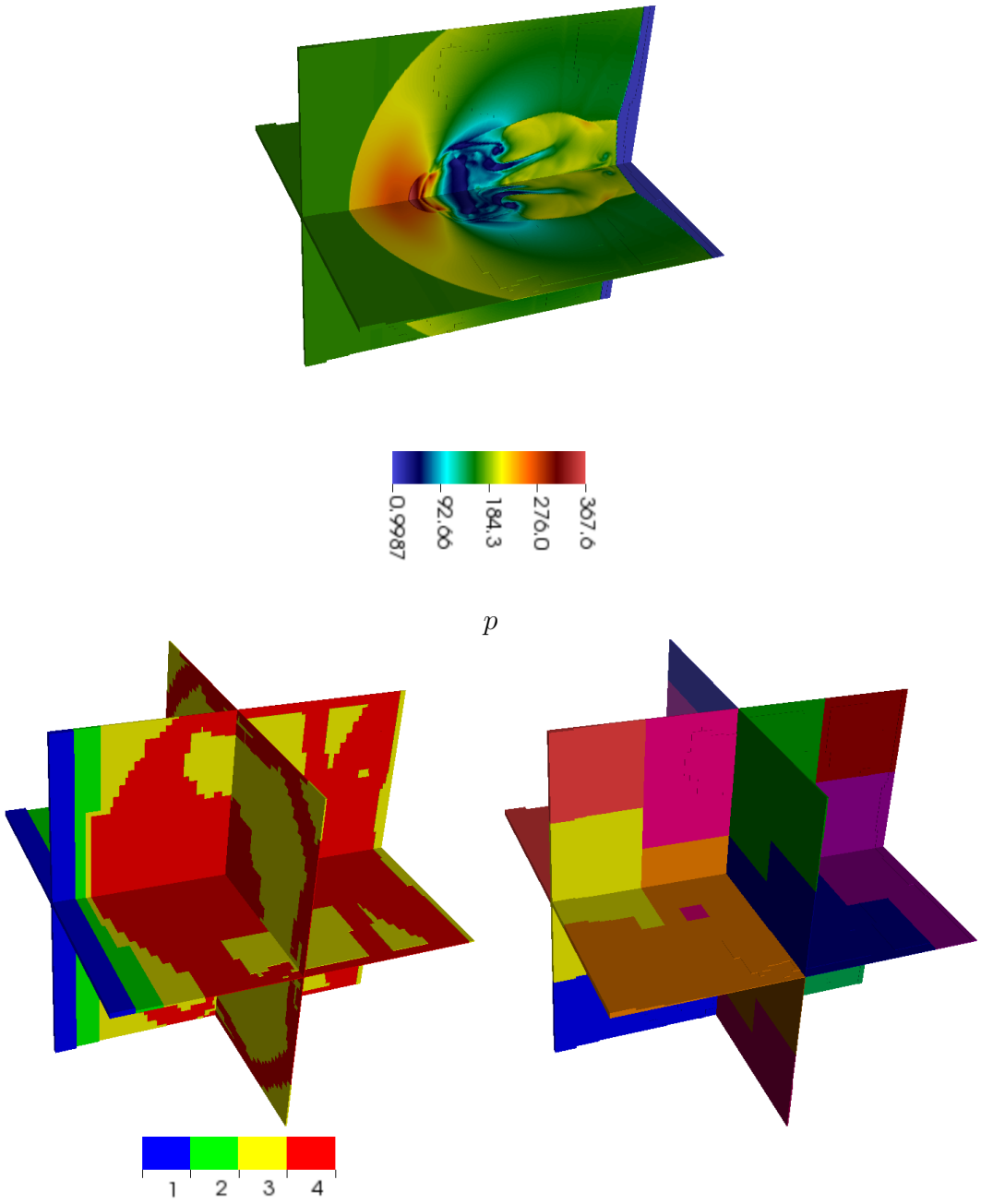
In Figure 8.23 are presented cuts of the solution for p, the adaptive mesh for this simulation with 3 levels and the mesh distribution among the 24 processors used. The figure containing the refinement of the adaptive mesh is interpreted so that the blue regions of the domain are represented in the coarsest level and the red region is represented by the most refined scale. This adaptive mesh allows the localisation of the bow shock, other structures and the centre of the explosion area and tail. In this tridimensional case, the mesh distribution per processor is too complex. Roughly to estimate the form of this distribution, it is used cuts.

Table 8.17 - SCI: Errors in p, memory consumption and CPU time obtained by using several refinement levels.

Base Mesh	**L**	**Accuracy**	**Cells**		**Patches**	**CPU Time**	
		$\mathbb{L}_1$ error ($\times 10^{-1}$)	#	%	#	(min)	%
64^3	4	1.196	83,081,640	61	15,723	11.0	36
128^3	3	1.156	82,664,744	61	15,863	11.4	37
256^3	2	1.173	82,986,448	61	16,207	14.4	48

Source: Author's production.

Figure 8.23 - SCI: Results for p, the adaptive mesh and mesh distribution per processors. This simulation was performed using 4 refinement levels with threshold value $\epsilon = 0.001$ and 24 processors.



Mesh refinement Mesh distribution per processor

Source: Author's production.

8.4 Magnethosphere simulation

This problem is proposed in Ogino et al. (1992) is a numerical experiment to simulate the near Earth environment. It considers the Earth as a sphere with constant density and pressure in time containing a magnetic dipole that will be compressed or stretched by the solar wind, producing well know physical phenomena such as Kelvin-Helmholtz instabilities and magnetic reconnections.

Physically, the modelling for this problem requires new parameters in relation to the MHD formulation deducted in Chapters 2 and 3. In special, the acceleration term $\mathbf{a}$ includes an external gravity field:

$$\mathbf{a} = \frac{q_\alpha}{m_\alpha}\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) + \mathbf{g}, \tag{8.16}$$

and the Ampère law is rewritten as:

$$\mathbf{J} = \nabla \times \left(\mathbf{B} - \mathbf{B_d}\right) \tag{8.17}$$

where $\mathbf{B_d}$ is the intrinsic dipole magnetic field of the Earth. The reason for subtract the dipole field from the Ampère law is to allow the electric current to be generated in the frontier between the two medium (interplanetary space and the outter terrestrial region).

In addiction, Ogino et. al. (OGINO, 1986; OGINO et al., 1992) introduces a viscosity term $\Phi = 10^{-5}\nabla^2\mathbf{u}$ and an artificial diffusion over ρ and p to the model. These effects are included in order to reduce the MHD fluctuations caused by the unbalanced forces in the initial condition. In these works, the MHD equations for this problem are presented in the primitive formulation. Thus, this work uses these assumptions to deduce a MHD model, analogously as performed in Chapters 2 and 3, governed by the system of equations in a semi-conservative formulation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = D\nabla^2\rho \tag{8.18a}$$

$$\begin{aligned}\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot \left[\rho \mathbf{u}\mathbf{u} + \left(p + \frac{\mathbf{B}\cdot\mathbf{B}}{2}\right)\mathbf{I} - \mathbf{B}\mathbf{B}\right] \\ = \rho\mathbf{g} + \mathbf{B} \times (\nabla \times \mathbf{B_d}) + \Phi + \mathbf{u}D\nabla^2\rho\end{aligned} \tag{8.18b}$$

$$\frac{\partial \mathcal{E}}{\partial t} + \nabla \cdot \left[\left(\mathcal{E} + p + \frac{\mathbf{B} \cdot \mathbf{B}}{2} \right) \mathbf{u} - (\mathbf{u} \cdot \mathbf{B})\, \mathbf{B} + (\eta \nabla \times \mathbf{B}) \times \mathbf{B} \right]$$
$$= \rho \mathbf{u} \cdot \mathbf{g} + \eta \|\nabla \times \mathbf{B_d}\|^2 + (\nabla \times \mathbf{B_d}) \cdot (\mathbf{u} \times \mathbf{B} - \eta \nabla \times \mathbf{B}) \tag{8.18c}$$
$$+ \frac{D_p \nabla^2 p}{\gamma - 1} + \frac{\|\mathbf{u}\|^2}{2} D \nabla^2 \rho + \mathbf{u} \cdot \Phi$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot \left[\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u} + \eta \left((\nabla \mathbf{B})^T - \nabla \mathbf{B} - (\nabla \mathbf{B}_d)^T + \nabla \mathbf{B_d} \right) \right] = \mathbf{0} \tag{8.18d}$$

where the terms in red are deducted from the gravity term included in the particle acceleration $\mathbf{a}$ and the dipole $\mathbf{B_d}$ from the Ampère law, and the terms written in blue correspond to the viscosity and artificial diffusion as introduced by Ogino (1986). These diffusion terms uses the constants $D = D_p = 0.02$.

The physical quantities included into the model, written in red, are proportional to the distance to Earth, represented by ξ. Assuming the Earth located in the xy plane origin, the distance ξ is defined as:

$$\xi = \sqrt{x^2 + y^2}. \tag{8.19}$$

Thus, the external gravity $\mathbf{g}$, defined by the vector field:

$$\mathbf{g}(x, y) = -\frac{g_0}{\xi^2} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}, \tag{8.20}$$

where $g_0 = 1.35 \times 10^{-6}$ and the line dipole magnetic field $\mathbf{B_d}$ is given by:

$$\mathbf{B_d}(x, y) = \begin{bmatrix} B_{d_x} \\ B_{d_y} \\ B_{d_z} \end{bmatrix} = \begin{bmatrix} -2xy\xi^{-4} \\ \xi^{-4}\,(x^2 - y^2) \\ 0 \end{bmatrix}. \tag{8.21}$$

The resistivity term η is defined as:

$$\eta(x, y) = \eta_0 \left(\frac{T}{T_0} \right)^{-\frac{3}{2}} \tag{8.22}$$

with $\eta_0 = 10^{-2}$, $T = \dfrac{p}{\rho}$ and $T_0 = 5.4 \times 10^{-7}$.

The physical quantities in this problem are normalised so that unit of distance correspond to the radius of the Earth (6.37×10^6 m), the unit of magnetic induction

correspond to the Earth magnetic field at the equator (3.12×10^{-5} T) and the time unit correspond to the Alfvén transit time (0.937 s), defined as the time required by the Alfvén wave to go through the equivalent of the Earth radius.

Based on these quantities, the pressure unit correspond to 7.75×10^{-4} N/m^2, the velocity unit correspond to 6.80×10^6 m/s, the acceleration unit correspond to 7.26×10^6 m/s^2 and the current density unit correspond to 3.90×10^{-6} A/m^2.

Initial Conditions

The initial configuration for this problem consists in a steady state ionosphere which describes the plasma in the Earth's neighbourhood. This ionosphere is constructed so that its pressure and density are proportional to ξ, while the initial magnetic field is the dipole presented in Equation 8.21 (OGINO, 1986). Thus, the ionosphere initial condition is given by:

$$\mathbf{q}^0(x,y) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} \max\left(\xi^{-2},\ 10^{-4}\right) \\ \max\left(p_{00}\xi^{-1},\ 3.56 \times 10^{-8}\right) \\ 0 \\ 0 \\ 0 \\ B_{d_x} \\ B_{d_y} \\ B_{d_z} \end{bmatrix} \tag{8.23}$$

where $p_{00} = \frac{(\gamma-1)g_0}{\gamma} = 5.4 \times 10^{-7}$ for $\gamma = 2$.

Boundary Conditions

This problem is simulated inside the physical domain $[-150, 450] \times [-150, 150]$, which is complemented with Neumann boundary conditions so that the derivative of the physical quantities are zero at the boundaries $x_e = 450$, $y_s = -150$ and $y_e = 150$. The boundary at $x_s = -150$ inputs the solar wind parameters, discussed in Section 8.4.

Furthermore, the physical domain also presents an internal boundary corresponding to the near Earth region. Considering the Earth positioned at the origin, this internal boundary removes the points which $\xi < 16$ from the computational domain. In order to damp out all perturbations near the ionosphere, the near Earth neighbourhood is

smoothed in relation with the initial condition after every time step by the operation:

$$\mathbf{q}^{n+1} = f\mathbf{q}_*^{n+1} + (1-f)\mathbf{q}^0 \tag{8.24}$$

where $\mathbf{q}_*^{n+1}$ is the solution obtained after the time evolution and $\mathbf{q}^0$ is the ionospheric initial condition. The weight value f is computed as:

$$\begin{aligned} \bar{f} &= 100\left(\max\left[\left(\tfrac{\xi}{16}\right)^2 - 1,\ 0\right]\right)^2 \\ f &= \tfrac{\bar{f}^2}{\bar{f}^2+1} \end{aligned} \tag{8.25}$$

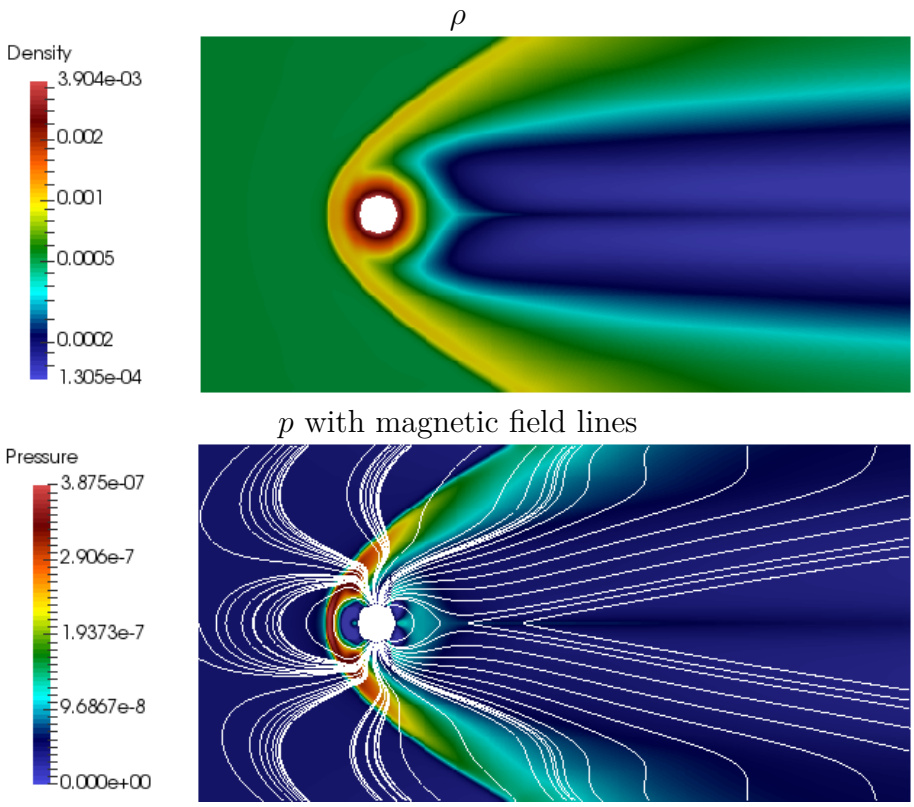
Solar wind setup

Defined the initial state of the ionosphere and the near Earth boundary condition, the magnetosphere is configured by introducing the solar wind as the boundary condition in $x_s = -150$. In this work, the configuration of the magnetosphere is performed in two steps. The first step introduces the hydrodynamic components of the solar wind as:

$$\mathbf{q}(x_s, y, t < t_{\text{hydro}}) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} 5 \times 10^{-4} \\ 3.56 \times 10^{-8} \\ 4.41 \times 10^{-2} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8.26}$$

These solar wind parameters are applied until the instant $t_{\text{hydro}} = 92,209.0$, which corresponds to approximately one day in the normalised time scale. This choice expected to be enough to balance the hydrodynamic components of the magnetosphere. In Figure 8.24 is presented the result after this hydrodynamic step.

Figure 8.24 - Magnetosphere setup after the hydrodynamic step.



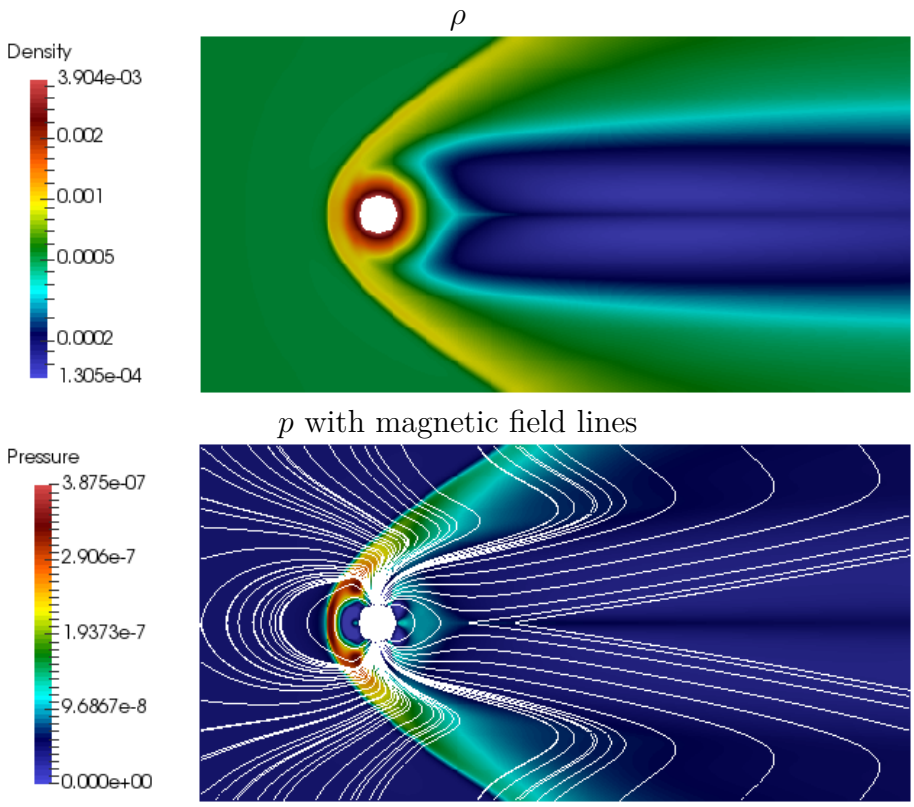
Source: Author's production.

The next step introduces the magnetic effects into the solar wind, that became:

$$\mathbf{q}(x_s, y, t_{\text{hydro}} < t < t_{\text{mag}}) = \begin{bmatrix} \rho \\ p \\ u_x \\ u_y \\ u_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} 5 \times 10^{-4} \\ 3.56 \times 10^{-8} \\ 4.41 \times 10^{-2} \\ 0 \\ 0 \\ 0 \\ 1.5 \times 10^{-4} \\ 0 \end{bmatrix}, \tag{8.27}$$

where $t_{\text{mag}} = 184,418.0$ corresponds to the end of the second day in the normalised time scale. This step is expected to balance the magnetic components of the magnetosphere. The Figure 8.25 presents a transitory configuration of the magnetosphere and plasma density distribution. This solution is converging to a steady state so that the magnetic dipole in the solar wind side is compressed and is elongated in the downstream.

Figure 8.25 - Magnetosphere setup before the inclusion of the real solar wind data.



Source: Author's production.

This magnetosphere configurations are expected in the near future to be used as initial conditions for simulations of the near Earth environment using real satellite data as the solar wind input.

9 LOCAL TIME STEPPING

In a different branch and complementary to the previous discussions in Section 6.3, this chapter presents studies involving local time stepping in the wavelet context. These methods improves the local time stepping approach presented in Domingues et al. (2008) in the context of the adaptive multiresolution method, as proposed in Harten (1995) in the CARMEN code developed in Roussel (2003) and Roussel et al. (2003) using the otimisation presented in Deiterding et al. (2016).

The presented local time stepping method is based on the natural extensions for Runge–Kutta methods, originated in Zennaro (1986). This formulation is used to obtain high-order interpolations and extrapolations necessary to the execution, synchronisation and update of the mesh hierarchy.

The proposed approach is an extension of the studies performed during the author's MSc dissertation (MOREIRA LOPES, 2014), that produced two articles. The first article, published in 2018 in the Journal of Applied Nonlinear Dynamics, presents a local time stepping formulation for a second order Runge-Kutta method. It is referred as (MOREIRA LOPES et al., 2018b), namely:

- MOREIRA LOPES, M.; DOMINGUES, M.; MENDES, O.; SCHNEIDER, K. An adaptive multiresolution scheme with second order local time-stepping for reaction-diffusion equations. **Journal of Applied Nonlinear Dynamics**, v. 7, n. 3, p. 287-295, 2018.

Annex B contains the second article that extends the results of the other article to higher order formulations, it presents more detailed algorithms concerning synchronisations in time and a discussion concerning the numerical stability of these models. It is referred as (MOREIRA LOPES et al., 2019), namely:

- MOREIRA LOPES, M.; DOMINGUES, M.O.; SCHNEIDER, K.; MENDES, O. Local time-stepping for adaptive multiresolution using natural extensions for Runge–Kutta methods. **Journal of Computational Physics**, v. 382, p. 291-318, 2019.

10 CONCLUSIONS

Motivated by the increasing needs of space environment science, scientific computation deals with mathematical-numerical methodologies to implement innovative computational codes. The National Institute for Space Research, INPE, has taken part in this kind of contribution to develop an efficient magnetohydrodynamic (MHD) simulation code aiming at future space weather forecasting.

This thesis develops a new high performance multi-dimensional adaptive MHD solver, as partially published in Moreira Lopes et al. (2018a). This solver uses the Adaptive Mesh Refinement in Object oriented C^{++} (AMROC) framework that implements a Structures Adaptive Mesh Refinement (SAMR) technique using a patch based data structure that decomposes the computational mesh into independent blocks with different refinement levels. The parallel MPI implementation applies a workload balance algorithm that estimates the computational cost associated with each submesh and distributes them into different processors. Moreover, the adaptive methodology uses the wavelet based refinement idea (DOMINGUES et al., 2019b).

In order to contribute with the accuracy and quality of the solutions provided by the MHD solver, this thesis presents a new divergence cleaning approach that combines the parabolic-hyperbolic correction introduced in Dedner et al. (2002) and the elliptic correction introduced in Munz et al. (2000), creating a triple parabolic-hyperbolic-elliptic correction that applies a multigrid strategy to overcome the performance limitations of the elliptic operator. This correction presented a reduction in the global divergence of the magnetic field, without compromising the solver performance. In the 2D problems the numerical simulations presented a large reduction in the divergence in relation with the obtained results for the 3D cases. This result is mainly due to the refinement of the 3D mesh, which is coarser than the ones used in the 2D problems. Furthermore, we expect to improve soon the 3D results even more by using finer meshes and optimising the multigrid parameters.

In the context of the adaptive modelling, the space and time adaptations are complementary approaches to perform accurate and fast simulations. However, combining these approaches has many challenges and open questions. The NERK local time methodology proposed in this thesis presented errors of about the same order of magnitude as the MRLT computations using classical RK schemes with the advantage of a significant gain in CPU time, with results published in Moreira Lopes et al. (2019).

The original contribution of this thesis are:

- Adaptive MHD solvers developed and implemented in a high performance environment (MOREIRA LOPES et al., 2018a).
- A mesh refinement criteria based on the wavelet formulation (DOMINGUES et al., 2019b).
- A computationally accurate local time stepping strategy in the wavelet context (MOREIRA LOPES et al., 2018b; MOREIRA LOPES et al., 2019).
- Approach to overcome the magnetic field numerical divergence problem.

Perspectives of work to be developed as consequence of this thesis:

- Extend the multigrid solver to support wavelet based adaptive meshes.
- Develop a multigrid solver for non-periodic boundaries in the AMROC framework context.
- Perform an analysis focusing on the effects of different choices of the multigrid parameters in 3D problems.
- Study implementation strategies and develop the high order local time stepping as presented (MOREIRA LOPES et al., 2019) in the AMROC framework.
- Validate the MHD solver implemented in the AMROC framework using satellite data as input to the Earth magnetosphere configuration in different solar wind conditions and others space applications.
- Obtain a space weather numerical forecasting modelling on the AMROC framework.
- Consider the eigenvalue system related to the resistive MHD quasi-linear form in its respective flux computation.
- Compare and extend the developed GLM-MHD model to recent iGLM formulation proposed by Derigs et al. (2018) conceived to conserve the system entropy.
- Develop a control time step like Domingues et al. (2009) for the local time stepping approach developed.

REFERENCES

AHRENS, J.; GEVECI, B.; LAW, C. **ParaView**: an end-user tool for large data visualization. Los Alamos: Los Alamos National Laboratory, 2005. 17 p. 69

ALMGREN, A. S.; BECKNER, V. E.; BELL, J. B.; DAY, M. S.; HOWELL, L. H.; JOGGERST, C. C.; LIJEWSKI, M. J.; NONAKA, A.; SINGER, M.; ZINGALE, M. CASTRO: A new compressible astrophysical solver. I. hydrodynamics and self-gravity. **The Astrophysical Journal**, v. 715, p. 1221–1238, 2010. 68

ARRARÁS, A.; GASPAR, F.; PORTERO, L.; RODRIGO, C. Domain decomposition multigrid methods for nonlinear reaction-diffusion problems. **Communications in Nonlinear Science and Numerical Simulation**, v. 20, n. 3, p. 699–710, 2015. 56

ASSOUS, F.; DEGOND, P.; HEINTZE, E.; RAVIART, P. A.; SEGRE, J. On a finite-element method for solving the three-dimensional Maxwell equations. **Journal of Computational Physics**, v. 109, n. 2, p. 222–237, 1993. 44

BALSARA, D.; KIM, J. An intercomparison between divergence-cleaning and staggered mesh formulations for numerical magnetohydrodynamics. **The Astrophysical Journal**, v. 602, 10 2003. 2

BALSARA, D. S.; DUMBSER, M. Divergence-free MHD on unstructured meshes using high order finite volume schemes based on multidimensional Riemann solvers. **Journal of Computational Physics**, v. 299, p. 687–715, 2015. 21

BALSARA, D. S.; SPICER, D. S. A staggered mesh algorithm using high order godunov fluxes to ensure solenoidal magnetic fields in magnetohydrodynamic simulations. **Journal of Computational Physics**, v. 149, n. 2, p. 270–292, 1999. 87, 88

BASTING, M.; KUZMIN, D. An FCT finite element scheme for ideal MHD equations in 1D and 2D. **Journal of Computational Physics**, v. 338, p. 585–605, 2017. 21

BELL, J.; BERGER, M.; SALTZMAN, J.; WELCOME., M. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. **SIAM Journal on Scientific Computing**, v. 15, n. 1, p. 127–138, 1994. 63, 65

BERGER, M. J. **Adaptive mesh refinement for hyperbolic partial differential equations**. Thesis (PhD) — Stanford University, Stanford, 1982. 1, 2, 62

BERGER, M. J.; CORELLA, P. Local adaptive mesh refinement for shock hydrodynamics. **Journal of Computational Physics**, v. 82, n. 1, p. 64–84, 1989. 1, 2, 62, 69

BERGER, M. J.; OLIGER, J. Adaptive mesh refinement for hyperbolic partial differential equations. **Journal of Computational Physics**, v. 53, p. 484–512, 1984. 1, 2, 62

BITTENCOURT, J. A. **Fundamentals of Plasma Physics**. 3. ed. New York: Springer–Verlag, 2004. 679 p. 5, 8, 146

BRACKBILL, J. U.; BARNES, D. C. The effect of nonzero $\nabla \cdot B$ on the numerical solution of the magnetohydrodynamic equations. **Journal of Computational Physics**, v. 35, n. 3, p. 426–430, 1980. 2, 43, 50

BRIGGS, W. L.; HENSON, V. E.; MCCORMICK, S. F. **A multigrid tutorial**. 2. ed. Philadelphia: Society for Industrial and Applied Mathematics, 2000. 193 p. 55, 56, 57

BRIO, M.; WU, C. C. An upwind differencing scheme for the equations of ideal magnetohydrodynamics. **Journal of Computational Physics**, v. 75, p. 400–422, 1988. 31

BURDEN, R. L.; FAIRES, J. D. **Numerical analysis**. Boston: PWS-Kent Publishing, 1989. 841 p. 22, 39, 52, 54

CHEN, F. F. **Introduction to plasma physics and controlled fusion**. Switzerland: Springer International, 2016. 490 p. 5

CHILDS, H.; BRUGGER, E.; WHITLOCK, B.; MEREDITH, J.; AHERN, S.; PUGMIRE, D.; BIAGAS, K.; MILLER, M.; HARRISON, C.; WEBER, G. H.; KRISHNAN, H.; FOGAL, T.; SANDERSON, A.; GARTH, C.; BETHEL, E. W.; CAMP, D.; RÜBEL, O.; DURANT, M.; FAVRE, J. M.; NAVRÁTIL, P. **VisIt**: an end-user tool for visualizing and analyzing very large data. In: BETHEL, E. W.; CHILDS, H.; HANSEN, C. (Ed.). **High performance visualization–enabling extreme-scale scientific insight**. 1. ed. New York: Chapman and Hall/CRC, 2012. chapter 16, p. 357–372. 69

CHRISTLIEB, A. J.; ROSSMANITH, J. A.; TANG, Q. Finite difference weighted essentially non-oscillatory schemes with constrained transport for ideal magnetohydrodynamics. **Journal of Computational Physics**, v. 268, p. 302–325, 2014. 21

COURTECUISSE, H.; ALLARD, J. Parallel dense Gauss-Seidel algorithm on many-core processors. In: INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING AND COMMUNICATIONS, 11, 2009, Seoul, South Korea. **Proceedings... IEEE**. Seoul, 2009. p. 139–147. 55

CUCINOTTA, F. A.; ALP, M.; ROWEDDER, B.; KIM, M.-H. Y. Safe days in space with acceptable uncertainty from space radiation exposure. **Life Sciences in Space Research**, v. 5, p. 31–38, 2015. 1

DAI, W.; WOODWARD, P. R. A simple finite difference scheme for multidimensional magnetohydrodynamical equations. **Journal of Computational Physics**, v. 142, n. 2, p. 331–369, 1998. 83

DAVIS, S. F. Simplified second-order Godunov-type methods. **SIAM Journal on Scientific and Statistical Computing**, v. 9, p. 445–473, 1988. 33

DEDNER, A.; KEMM, F.; KRÖNER, D.; MUNZ, C.-D.; SCHNITZER, T.; WESENBERG, M. Hyperbolic divergence cleaning for the MHD equations. **Journal of Computational Physics**, v. 175, n. 2, p. 645–673, 2002. 2, 3, 43, 44, 45, 49, 80, 114, 129

DEITERDING, R. Block-structured adaptive mesh refinement - theory, implementation and application. **ESAIM: Proceedings**, v. 34, p. 97–150, 2011. 2, 57, 62, 66, 68, 69

DEITERDING, R.; DOMINGUES, M. Evaluation of multiresolution mesh adaptation criteria in the AMROC framework. **Civil-Comp Press**, v. 111, 2017. 2, 66

DEITERDING, R.; DOMINGUES, M.; SCHNEIDER, K. Comparison of adaptive multiresolution and adaptive mesh refinement applied to simulations of the compressible euler equations. **SIAM Journal of Scientific Computing**, v. 38, p. S173–S193, 2016. 2, 3, 66, 127

_____. Multiresolution analysis as a criterion for effective dynamic mesh adaptation - a case study for euler equations in the samr framework AMROC. **Computers and Fluids**, 2018. (Submitted). 2, 66

DERIGS, D.; WINTERS, A. R.; GASSNER, G. J.; WALCH, S.; BOHM, M. Ideal GLM-MHD: about the entropy consistent nine-wave magnetic field divergence diminishing ideal magnetohydrodynamics equations. **Journal of Computational Physics**, v. 364, p. 420–467, 2018. 2, 130

DHATT, G.; TOUZOT, G.; LEFRANCOIS, E. **Finite element method**. London: John Wiley & Sons, 2012. 600 p. 21

DO, S.; LI, H.; KANG, M. Wavelet-based adaptation methodology combined with finite difference weno to solve ideal magnetohydrodynamics. **Journal of Computational Physics**, v. 339, p. 482–499, 2017. 21

DOLWITHAYAKUL, B.; CHANTRAPORNCHAI, C.; CHUMCHOB, N. GPU-based total variation image restoration using sliding window Gauss-Seidel algorithm. In: INTERNATIONAL SYMPOSIUM ON INTELLIGENT SIGNAL PROCESSING AND COMMUNICATIONS SYSTEMS, 2011, Chiang Mai, Thailand. **Proceedings...** [S.l.], 2011. p. 1–6. 55

DOMINGUES, M.; DEITERDING, R.; MOREIRA LOPES, M.; GOMES, A.; MENDES, O.; SCHNEIDER, K. Wavelet-based parallel dynamic mesh adaptation for magnetohydrodynamics in the AMROC framework. (accepted, 10.1016/j.compfluid.2019.06.025). 2019b. 3, 66, 69, 77, 129, 130

DOMINGUES, M.; ROUSSEL, O.; K., S. An adaptive multiresolution method for parabolic PDEs with time-step control. **International Journal for Numerical Methods in Engineering**, v. 78, p. 652–670, 2009. 130

DOMINGUES, M. O.; GOMES, A. K. F.; GOMES, S. M.; MENDES, O.; PIERRO, B. D.; SCHNEIDER, K. Extended generalised Lagrangian multipliers for magnetohydrodynamics using adaptive multiresolution methods. **ESAIM: Proceedings**, v. 43, p. 95–107, 2013. 2, 77, 78

DOMINGUES, M. O.; GOMES, S. M.; ROUSSEL, O.; SCHNEIDER, K. An adaptative multiresolution scheme with local time-stepping for evolutionary PDEs. **Journal of Computational Physics**, v. 227, n. 8, p. 3758–3780, 2008. 3, 68, 127

_____. Adaptive multiresolution methods. **ESAIM: Proceedings**, v. 34, p. 1–96, 2011. 2, 66

DOMINGUES, M. O.; MOREIRA LOPES, M.; DEITERDING, R.; MENDES, O. Discussions on the MHD adaptive solvers in the AMROC framework for space plasmas applications. In: INTERNATIONAL CONGRESS ON INDUSTRIAL

AND APPLIED MATHEMATICS, 2019, Valencia, Spain. **Proceedings...** [S.l.], 2019a. 3

EVANS, C. R.; HAWLEY, J. F. Simulation of magnetohydrodynamic flows - a constrained transport method. **Astrophysical Journal**, v. 332, p. 659–677, 1988. 44

FELKER, K. G.; STONE, J. A fourth-order accurate finite volume method for ideal MHD via upwind constrained transport. **Journal of Computational Physics**, v. 375, p. 1365 – 1400, 2018. 21

FRYXELL, B.; OLSON, K.; RICKER, P.; TIMMES, F. X.; ZINGALE, M.; LAMB, D. Q.; MACNEICE, P.; ROSNER, R.; TRURAN, J. W.; TUFO, H. FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. **The Astrophysical Journal Supplement Series**, v. 131, n. 1, p. 273–334, 2000. 68

GIACOMAZZO, B.; REZZOLLA, L. The exact solution of the Riemann problem in relativistic magnetohydrodynamics. **Journal of Fluid Mechanics**, v. 562, p. 223–259, 2006. 31

GODUNOV, S. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. **Matematicheskii Sbornik**, v. 47, n. 3, p. 271–306, 1959. 25

GOEDBLOED, J. P. H.; POEDTS, S. **Principles of Magnetohydrodynamics: With Applications to Laboratory and Astrophysical Plasmas**. New York: Cambridge University Press, 2004. 613 p. 8, 156

GOMES, A.; DOMINGUES, M. O.; SCHNEIDER, K.; MENDES, O.; DEITERDING, R. An adaptive multiresolution method for ideal magnetohydrodynamics using divergence cleaning with parabolic-hyperbolic correction. **Applied Numerical Mathematics**, v. 95, p. 199–213, 2015. 2

GOMES, A. K. F. **Análise multirresolução adaptativa no contexto da resolução numérica de um modelo de magnetohidrodinâmica ideal**. Thesis (Master in Applied Computing) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2012. 17

GOMES, A. K. F. **Simulação numérica de um modelo magneto-hidrodinâmico multidimensional no contexto da multirresolução adaptativa por médias celulares**. 171 p. Thesis (Doctor in

Applied Computing) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2017. 2, 77, 80

HAPGOOD, M. Towards a scientific understanding of the risk from extreme space weather. **Advances in Space Research**, v. 47, n. 12, p. 2059–2072, 2011. 1

HARTEN, A. Multiresolution algorithms for the numerical solution of hyperbolic conservation laws. **Communications on Pure and Applied Mathematics**, v. 48, n. 12, p. 1305–1342, 1995. 2, 3, 66, 69, 127

HARTEN, A.; LAX, P. D.; van LEER, B. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. **SIAM Review**, v. 25, n. 1, p. 35–61, 1983. 31, 33

HELZEL, C.; ROSSMANITH, J. A.; TAEL, B. An unstaggered constrained transport method for the 3D ideal magnetohydrodynamic equations. **Journal of Computational Physics**, v. 230, n. 10, p. 3803–3829, 2011. 89

HOPKINS, P. F. A constrained-gradient method to control divergence errors in numerical MHD. **Monthly Notices of the Royal Astronomical Society**, v. 462, n. 1, p. 576–587, 2016. 2, 43

JIANG, R.-L.; FANG, C.; CHEN, P.-F. A new MHD code with adaptive mesh refinement and parallelization for astrophysics. **Computer Physics Communications**, v. 183, n. 8, p. 1617–1633, 2012. 110

KARNIADAKIS, G. E.; KIRBY II, R. M. **Parallel scientific computing in C++ and MPI: a seamless approach to parallel algorithms and their implementation**. Cambridge: Cambridge University Press, 2003. 628 p. 55

KAWAI, S. Divergence-free-preserving high-order schemes for magnetohydrodynamics: an artificial magnetic resistivity method. **Journal of Computational Physics**, v. 251, p. 292–318, 2013. 2

KIVELSON, M. G.; RUSSELL, C. T. **Introduction to space physics**. Cambridge: Cambridge University Press, 1995. 588 p. (Cambridge atmospheric and space science series). 1, 18

KOREN, B. A robust upwind discretization method for advection, diffusion and source terms. In: VREUGDENHIL, C.; KOREN, B. (Ed.). **Numerical methods for advection-diffusion problems**. Germany: Vieweg, 1993, (Notes on Numerical Fluid Mechanics). p. 117–138. 27

LEVEQUE, R. **Finite difference methods for ordinary and partial differential equations**: steady-state and time-dependent problems. Philadelphia: Society for Industrial and Applied Mathematics, 2007. 357 p. 21

LEVEQUE, R. J. **Numerical methods for conservation laws**. Basel: Birkhäuser Verlag, 1990. 220 p. 21, 22, 45

LI, L.; ZHENG, W. A robust solver for the finite element approximation of stationary incompressible MHD equations in 3D. **Journal of Computational Physics**, v. 351, p. 254–270, 2017. 21

LIU, X.-D.; OSHER, S.; CHAN, T. Weighted essentially non-oscillatory schemes. **Journal of Computational Physics**, v. 115, n. 1, p. 200–212, 1994. 25

LONDRILLO, P.; ZANNA, L. D. High-order upwind schemes for multidimensional magnetohydrodynamics. **The Astrophysical Journal**, v. 530, n. 1, p. 508–524, 2000. 83, 87

MENDES, O.; MENDES DA COSTA, A.; DOMINGUES, M. O. Introduction to planetary electrodynamics: a view of electric fields and currents. **Advances in Space Research**, v. 35, n. 5, p. 812–828, 2005. 1

MIGNONE, A.; TZEFERACOS, P.; BODO, G. High-order conservative finite difference GLM-MHD schemes for cell-centered MHD. **Journal of Computational Physics**, v. 229, n. 17, p. 5896–5920, 2010. 2, 21

MIYOSHI, T.; KUSANO, K. A multi-state HLL approximate Riemann solver for ideal magnetohydrodynamics. **Journal of Computational Physics**, v. 208, p. 315–344, 2005. 35

_____. A comparative study of divergence-cleaning techniques for multi-dimensional MHD schemes. **Plasma and Fusion Research: Regular Articles**, v. 6, p. 2401124/1–2401124/5, 2011. 2, 44

MOREIRA LOPES, M. **Método de alta ordem para ajuste de passo de tempo local para a resolução numérica de equações diferenciais evolutivas com uso de análise multiresolução adaptativa.** Thesis (Master in Applied Computing) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2014. 3, 66, 68, 127

MOREIRA LOPES, M.; DEITERDING, R.; GOMES, A.; MENDES, O.; DOMINGUES, M. An ideal compressible magnetohydrodynamic solver with

parallel block-structured adaptive mesh refinement. **Computers and Fluids**, v. 173, p. 293–298, 2018. 3, 25, 77, 129, 130, 197

MOREIRA LOPES, M.; DOMINGUES, M.; MENDES, O.; SCHNEIDER, K. An adaptive multiresolution scheme with second order local time-stepping for reaction-diffusion equations. **Journal of Applied Nonlinear Dynamics**, v. 7, n. 3, p. 287–295, 2018. 3, 68, 127, 130

MOREIRA LOPES, M.; DOMINGUES, M.; SCHNEIDER, K.; MENDES, O. Local time-stepping for adaptive multiresolution using natural extension of Runge–Kutta methods. **Journal of Computational Physics**, v. 382, p. 291–318, 2019. 3, 68, 127, 129, 130, 167

MUNZ, C.-D.; OMNES, P.; SCHNEIDER, R.; SONNENDRÜCKER, E.; VOSS, U. Divergence correction techniques for Maxwell solvers based on a hyperbolic model. **Journal of Computational Physics**, v. 161, n. 2, p. 484–511, 2000. 2, 3, 44, 129

OGINO, T. A Three-Dimensional MHD simulation of the interaction of the Solar Wind with the Earth's Magnetosphere: the generation of field-aligned currents. **Journal of Geophysical Research**, v. 91, n. A6, p. 6791–6806, 1986. 120, 121, 122

OGINO, T.; WALKER, R. J.; ASHOUR-ABDALLA, M. A global magnetohydrodynamic simulation of the magnetosheath and magnetosphere when the interplanetary magnetic field is northward. **IEEE Transactions on Plasma Science**, v. 20, n. 6, p. 817–828, 1992. 120

ORSZAG, S. A.; TANG, C.-M. Small-scale structure of two-dimensional magnetohydrodynamic turbulence. **Journal of Fluid Mechanics**, v. 90, n. 1, p. 129–143, 1979. 83

POWELL, K.; ROE, P.; LINDE, T.; GOMBOSI, T.; ZEEUW, D. L. D. A solution-adaptive upwind scheme for ideal magnetohydrodynamics. **Journal of Computational Physics**, v. 154, p. 284–309, 1999. 1

POWELL, K. G. An approximate riemann solver for magnetohydrodynamics. In: HUSSAINI, M. Y.; van LEER, B.; van ROSENDALE, J. (Ed.). **Upwind and high-resolution schemes**. Berlin, Heidelberg: Springer, 1997. p. 570–583. 1, 44

ROE, P. L. Approximate Riemann solvers, parameter vectors, and difference schemes. **Journal of Computational Physics**, v. 43, n. 2, p. 357–372, 1981. 31

_____. Characteristic-based schemes for the Euler equations. **Annual Review of Fluid Mechanics**, v. 18, n. 1, p. 337–365, 1986. 26

ROUSSEL, O. **Devellopement d'un algorithme multiresolution adaptatif tridimensionnel pour la resolution des equations aux derivees patielles paraboliques: application aux instabilites thermodiffusives de flamme.** Thesis (Doctorat in Mécanique des Fluides) — Universite de la mediterranee, Marseille, 2003. 66, 68, 77, 127

ROUSSEL, O.; SCHNEIDER, K.; TSIGULIN, A.; BOCKHORN, H. A conservative fully adaptive multiresolution algorithm for parabolic PDEs. **Journal of Computational Physics**, v. 188, n. 2, p. 493–523, 2003. 3, 127

RYU, D.; MINIATI, F.; JONES, T.; FRANK, A. A divergence-free upwind code for multidimensional magnetohydrodynamic flows. **The Astrophysical Journal**, v. 509, n. 1, p. 244–255, 1998. 83

SCHRIJVER, C. J.; KAURISTIE, K.; AYLWARD, A. D.; DENARDINI, C. M.; GIBSON, S. E.; GLOVER, A.; GOPALSWAMY, N.; GRANDE, M.; HAPGOOD, M.; HEYNDERICKX, D.; JAKOWSKI, N.; KALEGAEV, V. V.; LAPENTA, G.; LINKER, J. A.; LIU, S.; MANDRINI, C. H.; MANN, I. R.; NAGATSUMA, T.; NANDY, D.; OBARA, T.; OâBRIEN, T. P.; ONSAGER, T.; OPGENOORTH, H. J.; TERKILDSEN, M.; VALLADARES, C. E.; VILMER, N. Understanding space weather to shield society: a global road map for 2015–2025 commissioned by COSPAR and ILWS. **Advances in Space Research**, v. 55, n. 12, p. 2745–2807, 2015. 1

SHAKERI, F.; DEHGHAN, M. A finite volume spectral element method for solving magnetohydrodynamic (MHD) equations. **Applied Numerical Mathematics**, v. 61, n. 1, p. 1–23, 2011. 21

STONE, J. M.; GARDINER, T. A.; TEUBEN, P.; HAWLEY, J. F.; SIMON, J. B. **Athena**: a new code for astrophysical MHD. Available in `https://arxiv.org/pdf/0804.0402.pdf`. 10 2018. 87

TAKAHASHI, K.; YAMADA, S. Exact Riemann solver for ideal magnetohydrodynamics that can handle all types of intermediate shocks and switch-on/off waves. **Journal of Plasma Physics**, v. 80, n. 2, p. 255–287, 2014. 31

TITAREV, V.; TORO, E. Finite-volume WENO schemes for three-dimensional conservation laws. **Journal of Computational Physics**, v. 201, n. 1, p. 238–260, 2004. 25

TORO, E. F. **Riemann solvers and numerical methods for fluid dynamics: a practical introduction**. Berlin: Springer, 1999. 724 p. 24, 26, 35

TÓTH, G. The $\nabla \cdot B$ constraint in shock-capturing magnetohydrodynamics codes. **Journal of Computational Physics**, v. 161, p. 605–652, 2000. 2, 43

TÓTH, G.; ODSTRCIL, D. Comparison of some flux corrected transport and total variation diminishing numerical schemes for hydrodynamic and magnetohydrodynamic problems. **Journal of Computational Physics**, v. 128, n. 1, p. 82–100, 1996. 86

TÓTH, G.; van der HOLST, B.; SOKOLOV, I. V.; ZEEUW, D. L. D.; GOMBOSI, T. I.; FANG, F. Adaptive numerical algorithms in space weather modeling. **Journal of Computational Physics**, v. 231, n. 3, p. 870–903, 2012. 1, 117

URBAN, K. **Wavelet methods for elliptic partial differential equations**. New York: Oxford University Press, 2009. 482 p. 56, 58

van ALBADA, G.; van LEER, B.; ROBERTS, W. A comparative study of computational methods in cosmic gas dynamics. In: HUSSAINI, M.; van LEER, B.; van ROSENDALE, J. (Ed.). **Upwind and high-resolution schemes**. Berlin, Heidelberg: Springer, 1997. p. 95–103. 26

van LEER, B. Towards the ultimate conservative difference scheme. II. monotonicity and conservation combined in a second-order scheme. **Journal of Computational Physics**, v. 14, n. 4, p. 361–370, 1974. 27

_____. Towards the ultimate conservative difference scheme III. upstream-centered finite-difference schemes for ideal compressible flow. **Journal of Computational Physics**, v. 23, n. 3, p. 263–275, 1977. 26

_____. Towards the ultimate conservative difference scheme. V. a second-order sequel to Godunov's method. **Journal of Computational Physics**, v. 32, n. 1, p. 101–136, 1979. 25

VARGA, R. S. **Matrix iterative analysis**. 2. ed. Berlin Heidelberg: Springer, 1999. 358 p. (Springer Series in Computational Mathematics). 51

WESSELING, P. **Introduction to multigrid methods**. Portland: R.T. Edwards, 2004. 296 p. 56

XU, Z.; BALSARA, D. S.; DU, H. Divergence-free WENO reconstruction-based finite volume scheme for solving ideal MHD equations on triangular meshes. **Communications in Computational Physics**, v. 19, n. 4, p. 841–880, 2016. 25

YANG, J.; HE, Y.; ZHANG, G. On an efficient second order backward difference Newton scheme for MHD system. **Journal of Mathematical Analysis and Applications**, v. 458, n. 1, p. 676–714, 2018. 21

YAVNEH, I. Why multigrid methods are so efficient. **Computing Science and Engineering**, v. 8, p. 12–22, 2006. 56

ZACHARY, A. L.; MALAGOLI, A.; CORELLA, P. A higher-order Gudonov method for multidimensional ideal magnetohydrodynamics. **SIAM Journal on Scientific Computing**, v. 15, n. 2, p. 263–284, 1994. 87

ZENNARO, M. Natural continuous extensions of Runge–Kutta methods. **Mathematics of Computation**, v. 46, n. 173, p. 119–133, 1986. 68, 127

ANNEX A - PROOFS OF THE THEOREMS

Theorem 1, page 10: The moments of the Boltzmann Equation produces the following set of equations:

$$\frac{\partial n_\alpha m_\alpha}{\partial t} + \nabla \cdot (n_\alpha m_\alpha \mathbf{u}_\alpha) = S_\alpha \tag{A.1a}$$

$$\frac{\partial \left(n_\alpha m_\alpha \mathbf{u}_\alpha\right)}{\partial t} + \nabla \cdot \left[n_\alpha m_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha + \mathbb{P}_\alpha\right] - n_\alpha q_\alpha \left(\mathbf{E} + \mathbf{u}_\alpha \times \mathbf{B}\right) = \mathbf{A}_\alpha. \tag{A.1b}$$

$$\frac{1}{2}\frac{\partial n_\alpha m_\alpha \|\mathbf{u}_\alpha\|^2}{\partial t} + \frac{\partial N_d k n_\alpha T_\alpha}{\partial t} + \nabla_r \cdot \left(\frac{1}{2} n_\alpha m_\alpha \|\mathbf{u}_\alpha\|^2 \mathbf{u}_\alpha + \frac{1}{2} N_d k n_\alpha T_\alpha \mathbf{u}_\alpha + \mathbf{u}_\alpha \cdot \mathbb{P}_\alpha + \mathbf{h}_\alpha\right) - n_\alpha q_\alpha \mathbf{E} \cdot \mathbf{u}_\alpha = \mathbf{u}_\alpha \cdot \mathbf{A}_\alpha + Q_\alpha \tag{A.1c}$$

where q_α is the electric charge of a particle α, S_α is rate per unit of volume which the particles α are produced or lost due to collision effects, $\mathbf{A}$ is the rate of change of the momentum of the particles α due to collisions with particles of another types, Q_α is generated heat per unit of volume of the particles α due to collisions with another particles, $\mathbf{B}$ is the magnetic field and $\mathbf{E}$ is the electric field.

Proof. This proof is divided into three parts, which each one consist in taking a different moment of the Boltzmann equation.

Proof part 1: Taking the 0^{th} moment

The 0^{th} moment of the Boltzmann Equation produces the following relation:

$$\underbrace{m_\alpha \int_{\mathbf{v}} \frac{\partial f_\alpha}{\partial t} d^3v}_{I} + \underbrace{m_\alpha \int_{\mathbf{v}} \mathbf{v} \cdot \nabla_r f_\alpha d^3v}_{II} + \underbrace{m_\alpha \int_{\mathbf{v}} \mathbf{a} \cdot \nabla_v f_\alpha d^3v}_{III} = \underbrace{m_\alpha \int_{\mathbf{v}} \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v}_{IV}, \tag{A.2}$$

In order to facilitate this proof, the integrals I, II, III and IV are treated separately. Then, they are assembled in Equation A.19.

Integral I

Considering the relation presented in Equation 2.10, the integral I can be rewritten as:

$$m_\alpha \int_{\mathbf{v}} \frac{\partial f_\alpha}{\partial t} d^3v = m_\alpha \frac{\partial}{\partial t} \int_{\mathbf{v}} f_\alpha d^3v = \frac{\partial n_\alpha m_\alpha}{\partial t}. \tag{A.3}$$

Integral II

Using the derivative product rule, the integral II can be splitted into:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \cdot \nabla_r f_\alpha d^3v = m_\alpha \int_{\mathbf{v}} \nabla_r \cdot (f_\alpha \mathbf{v})\, d^3v - m_\alpha \int_{\mathbf{v}} f_\alpha \left(\nabla_r \cdot \mathbf{v}\right) d^3v. \tag{A.4}$$

As the variables $\mathbf{v}$ and $\mathbf{r}$ of the phase space are independent, the linear operator ∇_r treats the variable $\mathbf{v}$ as a constant. Therefore the second integral of the right side is null. Applying the Leibniz's rule into the first integral, the integral II is rewritten as:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \cdot \nabla_r f_\alpha d^3v = m_\alpha \nabla_r \cdot \int_{\mathbf{v}} (f_\alpha \mathbf{v})\, d^3v. \tag{A.5}$$

Lastly, using the definition given in Equation 2.12, the integral II is calculated as:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \cdot \nabla_r f_\alpha d^3v = m_\alpha \nabla_r \cdot (n_\alpha \mathbf{u}_\alpha) = \nabla \cdot (n_\alpha m_\alpha \mathbf{u}_\alpha)\,. \tag{A.6}$$

Integral III

In order to calculate the integral III, the acceleration term $\mathbf{a}$ is modelled by the Newton's second law $\mathbf{F} = m_\alpha \mathbf{a}$, where the force $\mathbf{F}$ is the Lorentz force. Thus, considering SI units, this term is given by:

$$\mathbf{a} = \frac{q_\alpha}{m_\alpha} \left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right), \tag{A.7}$$

where $\mathbf{E}$ is the external electric field and $\mathbf{B}$ is the magnetic field. Substituting this acceleration into the integral III:

$$m_\alpha \int_{\mathbf{v}} \mathbf{a} \cdot \nabla_v f_\alpha d^3v = \int_{\mathbf{v}} q_\alpha \left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) \cdot \nabla_v f_\alpha d^3v. \tag{A.8}$$

The right side integral can be splitted into the integrals denoted by $III.a$ and $III.b$, obtaining:

$$m_\alpha \int_{\mathbf{v}} \mathbf{a} \cdot \nabla_v f_\alpha d^3v = q_\alpha \left(\underbrace{\int_{\mathbf{v}} \mathbf{E} \cdot \nabla_v f_\alpha d^3v}_{III.a} + \underbrace{\int_{\mathbf{v}} (\mathbf{v} \times \mathbf{B}) \cdot \nabla_v f_\alpha d^3v}_{III.b} \right). \tag{A.9}$$

Using the derivative product rule, the integral $III.a$ can be written as:

$$\int_{\mathbf{v}} \mathbf{E} \cdot \nabla_v f_\alpha d^3v = \int_{\mathbf{v}} \nabla_v \cdot (f_\alpha \mathbf{E})\, d^3v - \int_{\mathbf{v}} f_\alpha \left(\nabla_v \cdot \mathbf{E}\right) d^3v. \tag{A.10}$$

The second integral of the right side is null due to the electric field $\mathbf{E}$ being independent of the velocity field $\mathbf{v}$. Applying the Gauss' divergence theorem at the first integral in the right side, the integral $III.a$ is rewritten as:

$$\int_{\mathbf{v}} \mathbf{E} \cdot \nabla_v f_\alpha d^3v = \oint_{\mathcal{S}} (f_\alpha \mathbf{E} \cdot \mathbf{n})\, d\mathcal{S}, \tag{A.11}$$

where $\mathcal{S}$ is a surface that bounds the entire space $\mathbf{v}$, and $\mathbf{n}$ is a outward pointing unit vector normal to the surface $\mathcal{S}$. Considering the Maxwellian distribution, the term f_α vanishes quickly as it goes to the boundaries of the surface $\mathcal{S}$. Therefore, the integral $III.a$ is calculated as:

$$\int_{\mathbf{v}} \mathbf{E} \cdot \nabla_v f_\alpha d^3v = 0 \tag{A.12}$$

The integral $III.b$ also can be decomposed by the derivative product rule, obtaining:

$$\int_{\mathbf{v}} (\mathbf{v} \times \mathbf{B}) \cdot \nabla_v f_\alpha d^3v = \int_{\mathbf{v}} \nabla_v \cdot (f_\alpha \mathbf{v} \times \mathbf{B})\, d^3v - \int_{\mathbf{v}} f_\alpha \nabla_v \cdot (\mathbf{v} \times \mathbf{B})\, d^3v \tag{A.13}$$

The second integral at the right side is zero due to the operator ∇_v being perpendicular to $\mathbf{v} \times \mathbf{B}$, while the first integral can be studied by applying the Gauss' divergence theorem, obtaining:

$$\int_{\mathbf{v}} (\mathbf{v} \times \mathbf{B}) \cdot \nabla_v f_\alpha d^3v = \oint_{\mathcal{S}} (f_\alpha \mathbf{v} \times \mathbf{B}) \cdot \mathbf{n} d\mathcal{S}. \tag{A.14}$$

Performing an analogous analysis as the surface integral obtained in Equation A.11, the integral $III.b$ became:

$$\int_{\mathbf{v}} (\mathbf{v} \times \mathbf{B}) \cdot \nabla_v f_\alpha d^3v = 0. \tag{A.15}$$

Once the integrals $III.a$ and $III.b$ are null, the integral III is also null, *i.e.*:

$$m_\alpha \int_{\mathbf{v}} \mathbf{a} \cdot \nabla_v f_\alpha d^3v = 0. \tag{A.16}$$

Integral IV

Using the derivative product rule, the integral IV can be splitted into:

$$m_\alpha \int_{\mathbf{v}} \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = m_\alpha \left(\frac{\delta}{\delta t}\int_{\mathbf{v}} f_\alpha d^3v\right)_{coll} \tag{A.17}$$

Using the definition of n_α presented in Equation 2.10, the result of the integral IV, denoted by S_α is calculated as:

$$S_\alpha = m_\alpha \int_{\mathbf{v}} \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = \left(\frac{\delta n_\alpha m_\alpha}{\delta t}\right)_{coll} \tag{A.18}$$

This value represents the rate per unit of volume which the particles α are produced or lost due to collision effects such as ionization, recombination, attachment, charge transfer or other effects (BITTENCOURT, 2004).

Assembling the* 0^{th} *moment integrals

Substituting the Equations A.3, A.6, A.16 and A.18 into Equation A.2 produces the Equation 2.20a:

$$\frac{\partial n_\alpha m_\alpha}{\partial t} + \nabla \cdot (n_\alpha m_\alpha \mathbf{u}_\alpha) = S_\alpha. \tag{A.19}$$

This equation, called continuity equation, describes the transport of mass through the medium.

Proof part 2: Taking the 1^{st} moment

The 1^{st} moment of the Boltzmann Equation produces the following relation:

$$\begin{aligned}\underbrace{m_\alpha \int_{\mathbf{v}} \mathbf{v}\frac{\partial f_\alpha}{\partial t} d^3v}_{V} + \underbrace{m_\alpha \int_{\mathbf{v}} \mathbf{v}\,(\mathbf{v}\cdot\nabla_r)\, f_\alpha d^3v}_{VI} + \underbrace{m_\alpha \int_{\mathbf{v}} \mathbf{v}\,(\mathbf{a}\cdot\nabla_v f_\alpha)\, d^3v}_{VII} = \\ \underbrace{m_\alpha \int_{\mathbf{v}} \mathbf{v}\left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v}_{VIII},\end{aligned} \tag{A.20}$$

Analogously as performed in the Part 1, the integrals V, VI, VII and $VIII$ are treated separately. Then, they are assembled in Equation A.43.

***Integral* V**

Using the derivative product rule, the integral V is decomposed as:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \frac{\partial f_\alpha}{\partial t} d^3v = m_\alpha \int_{\mathbf{v}} \frac{\partial}{\partial t} \left(\mathbf{v} f_\alpha\right) d^3v - m_\alpha \int_{\mathbf{v}} f_\alpha \frac{\partial \mathbf{v}}{\partial t} d^3v. \tag{A.21}$$

Considering that the phase space velocity $\mathbf{v}$ and t are independent variables, the second integral of the right side is null. Using the Leibniz integral rule, the integral V is written as:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \frac{\partial f_\alpha}{\partial t} d^3v = m_\alpha \frac{\partial}{\partial t} \int_{\mathbf{v}} \mathbf{v} f_\alpha d^3v. \tag{A.22}$$

Applying the definition of n_α presented in Equation 2.10, the integral V is calculated:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \frac{\partial f_\alpha}{\partial t} d^3v = \frac{\partial \left(n_\alpha m_\alpha \mathbf{u}_\alpha\right)}{\partial t}. \tag{A.23}$$

Integral VI

Using some vector calculus identities, the integral VI became:

$$\begin{aligned} m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{v} \cdot \nabla_r\right) f_\alpha d^3v = m_\alpha \int_{\mathbf{v}} \nabla_r \cdot \left(f_\alpha \mathbf{v}\mathbf{v}\right) d^3v - m_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{v} \cdot \nabla_r \mathbf{v} d^3v \\ - m_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{v} \nabla_r \cdot \mathbf{v} d^3v. \end{aligned} \tag{A.24}$$

Due to the independence of the variables $\mathbf{v}$ and $\mathbf{r}$ in the phase space, the operator ∇_r treats the variable $\mathbf{v}$ as a constant. Therefore the second and third integrals of the right side are null, obtaining:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{v} \cdot \nabla_r\right) f_\alpha d^3v = m_\alpha \int_{\mathbf{v}} \nabla_r \cdot \left(f_\alpha \mathbf{v}\mathbf{v}\right) d^3v. \tag{A.25}$$

Using the Leibniz integral rule, the right side term can be rewritten as:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{v} \cdot \nabla_r\right) f_\alpha d^3v = m_\alpha \nabla_r \cdot \int_{\mathbf{v}} \left(f_\alpha \mathbf{v}\mathbf{v}\right) d^3v. \tag{A.26}$$

As presented in Equation 2.11, the velocity term $\mathbf{v}$ can be decomposed into the sum of the average fluid velocity $\mathbf{u}_\alpha$ and the thermal velocity $\mathbf{w}_\alpha$, obtaining:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{v} \cdot \nabla_r\right) f_\alpha d^3v = m_\alpha \nabla_r \cdot \int_{\mathbf{v}} f_\alpha \left(\mathbf{u}_\alpha + \mathbf{w}_\alpha\right) \left(\mathbf{u}_\alpha + \mathbf{w}_\alpha\right) d^3v. \tag{A.27}$$

Distributing the terms, the right side integral is splitted into the sum:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{v} \cdot \nabla_r\right) f_\alpha d^3v = m_\alpha \nabla_r \cdot \left[\int_{\mathbf{v}} f_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha d^3v + \int_{\mathbf{v}} f_\alpha \mathbf{w}_\alpha \mathbf{w}_\alpha d^3v + 2\int_{\mathbf{v}} f_\alpha \mathbf{u}_\alpha \mathbf{w}_\alpha d^3v\right] \tag{A.28}$$

Considering that $\mathbf{u}_\alpha$ is an average value in the physical space and is independent of the velocity space, it is treated as a constant by the integral operator over the velocity space. Therefore:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{v} \cdot \nabla_r\right) f_\alpha d^3v = m_\alpha \nabla_r \cdot \left[\mathbf{u}_\alpha \mathbf{u}_\alpha \int_{\mathbf{v}} f_\alpha d^3v + \int_{\mathbf{v}} f_\alpha \mathbf{w}_\alpha \mathbf{w}_\alpha d^3v + 2\mathbf{u}_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{w}_\alpha d^3v\right] \tag{A.29}$$

The third integral of the right side is equivalent to the average $< \mathbf{w} >_\alpha$, which is equal zero according Equation 2.13. Using the definitions for n_α and pressure tensor $\mathbb{P}_\alpha$ presented in the Equations 2.10 and 2.14c into the first and second integrals of the right side, respectively, the integral VI is calculated as:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{v} \cdot \nabla_r\right) f_\alpha d^3v = \nabla \cdot \left[n_\alpha m_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha + \mathbb{P}_\alpha\right] \tag{A.30}$$

Integral VII

Using some vector calculus identities, the integral VII is expanded into the terms:

$$\begin{aligned} m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = m_\alpha \int_{\mathbf{v}} \nabla_v \cdot \left(f_\alpha \mathbf{v}\mathbf{a}\right) d^3v - m_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{v} \nabla_v \cdot \mathbf{a} d^3v \\ - m_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{a} \cdot \nabla_v \mathbf{v} d^3v \end{aligned} \tag{A.31}$$

Applying the Gauss' divergence theorem at the first integral in the right side, the integral VII is rewritten as:

$$\begin{aligned} m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = m_\alpha \oint_{\mathcal{S}} \left(f_\alpha \mathbf{v}\mathbf{a}\right) \cdot \mathbf{n} d\mathcal{S} - m_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{v} \nabla_v \cdot \mathbf{a} d^3v \\ - m_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{a} \cdot \nabla_v \mathbf{v} d^3v \end{aligned} \tag{A.32}$$

Analogously as performed for the Equation A.11, the first integral of the right side vanishes due to the Maxwellian distribution. The second integral also vanishes due to the term:

$$\nabla_v \cdot \mathbf{a} = \frac{q_\alpha}{m_\alpha} \nabla_v \cdot \left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) = \frac{q_\alpha}{m_\alpha} \nabla_v \cdot \mathbf{E} + \frac{q_\alpha}{m_\alpha} \nabla_v \cdot \left(\mathbf{v} \times \mathbf{B}\right) = 0, \tag{A.33}$$

since $\mathbf{E}$ is independent of $\mathbf{v}$ and the operator ∇_v is perpendicular to $\mathbf{v}\times\mathbf{B}$. Therefore, the integral VII became:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -m_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{a} \cdot \nabla_v \mathbf{v} d^3v \tag{A.34}$$

Applying the acceleration term $\mathbf{a}$ presented in Equation A.7 and assuming $\nabla_v \mathbf{v} = \mathbb{I}$:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -q_\alpha \int_{\mathbf{v}} f_\alpha \left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) d^3v \tag{A.35}$$

The right side integral can be splitted as:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -q_\alpha \left(\int_{\mathbf{v}} f_\alpha \mathbf{E} d^3v + \int_{\mathbf{v}} f_\alpha \mathbf{v} \times \mathbf{B} d^3v\right) \tag{A.36}$$

Considering that $\mathbf{E}$ and $\mathbf{B}$ are independent of the velocity space, the integrals are rewritten as:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -q_\alpha \left[\mathbf{E} \int_{\mathbf{v}} f_\alpha d^3v + \left(\int_{\mathbf{v}} f_\alpha \mathbf{v} d^3v\right) \times \mathbf{B}\right] \tag{A.37}$$

Using the definitions of n_α and $\mathbf{u}_\alpha$ presented in Equations 2.10 and 2.12, the integral VII is calculated as:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -n_\alpha q_\alpha \left(\mathbf{E} + \mathbf{u}_\alpha \times \mathbf{B}\right) \tag{A.38}$$

Integral $VIII$

Using the derivative product rule, the integral $VIII$ can be splitted into:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = m_\alpha \int_{\mathbf{v}} \left(\frac{\delta (f_\alpha \mathbf{v})}{\delta t}\right)_{coll} d^3v - m_\alpha \int_{\mathbf{v}} f_\alpha \left(\frac{\delta \mathbf{v}}{\delta t}\right)_{coll} d^3v. \tag{A.39}$$

The second integral in the right side integral is null due to the phase space velocity $\mathbf{v}$ and t being independent variables. Therefore, the integral $VIII$ became:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = m_\alpha \int_{\mathbf{v}} \left(\frac{\delta (f_\alpha \mathbf{v})}{\delta t}\right)_{coll} d^3v. \tag{A.40}$$

Applying the Leibniz's rule, the integral of the right side is rewritten as:

$$m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = m_\alpha \left(\frac{\delta}{\delta t} \int_{\mathbf{v}} (f_\alpha \mathbf{v}) d^3v\right)_{coll}. \tag{A.41}$$

Using the definition $\mathbf{u}_\alpha$ presented in Equation 2.12, the result of the integral $VIII$, denoted by $\mathbf{A}_\alpha$ is calculated as:

$$\mathbf{A}_\alpha = m_\alpha \int_{\mathbf{v}} \mathbf{v} \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = \left(\frac{\delta n_\alpha m_\alpha \mathbf{u}_\alpha}{\delta t}\right)_{coll}. \tag{A.42}$$

This term represents the rate of change of the momentum of the particles α due to collisions with particles of another types. This term does not consider collisions between particles of the same type due to the momentum conservation, which causes the collision between particles of the same type α not change the total quantity of momentum among these particles.

Assembling the* 1^{st} *moment integrals

Substituting the Equations A.23, A.30, A.38 and A.42 into the Equation A.20 produces the Equation 2.20b:

$$\frac{\partial \left(n_\alpha m_\alpha \mathbf{u}_\alpha\right)}{\partial t} + \nabla \cdot \left[n_\alpha m_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha + \mathbb{P}_\alpha\right] - n_\alpha q_\alpha \left(\mathbf{E} + \mathbf{u}_\alpha \times \mathbf{B}\right) = \mathbf{A}_\alpha. \tag{A.43}$$

This equation, called momentum equation, describes the rate of change of the average momentum for each fluid unit.

Proof part 3: Taking the 2^{nd} moment

The 2^{nd} moment of the Boltzmann Equation produces the following relation:

$$\begin{aligned} \underbrace{m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \frac{\partial f_\alpha}{\partial t} d^3v}_{IX} + \underbrace{m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\mathbf{v} \cdot \nabla_r\right) f_\alpha d^3v}_{X} + \underbrace{m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v}_{XI} \\ = \underbrace{m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v}_{XII}, \end{aligned} \tag{A.44}$$

Analogously as performed in the previous parts, the integrals IX, X, XI and XII

are treated separately. Then, they are assembled in Equation A.81.

Integral IX

Using the derivative product rule, the integral IX can be splitted into:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \frac{\partial f_\alpha}{\partial t} d^3v = m_\alpha \int_{\mathbf{v}} \frac{\partial \left(\|\mathbf{v}\|^2 f_\alpha\right)}{\partial t} d^3v - m_\alpha \int_{\mathbf{v}} f_\alpha \frac{\partial \|\mathbf{v}\|^2}{\partial t} d^3v. \tag{A.45}$$

Considering that the phase space velocity $\mathbf{v}$ and t are independent variables, the derivative in the second integral in the right side is null. Thus, the integral is also null. Applying the Leibniz's rule into the first integral in the right side, the integral IX is rewritten as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \frac{\partial f_\alpha}{\partial t} d^3v = m_\alpha \frac{\partial}{\partial t} \int_{\mathbf{v}} \|\mathbf{v}\|^2 f_\alpha d^3v. \tag{A.46}$$

Using the notation of average value presented in Equation 2.9, the integral in the right side is rewritten as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \frac{\partial f_\alpha}{\partial t} d^3v = \frac{\partial n_\alpha m_\alpha < \|\mathbf{v}\|^2 >_\alpha}{\partial t}. \tag{A.47}$$

Decomposing the velocity term $\mathbf{v}$ as discussed in Equation 2.11, the average value became:

$$< \|\mathbf{v}\|^2 >_\alpha = < \|(\mathbf{u}_\alpha + \mathbf{w}_\alpha)\|^2 >_\alpha . \tag{A.48}$$

The norm in the right side can be distributed into the sum:

$$< \|\mathbf{v}\|^2 >_\alpha = < \|\mathbf{u}_\alpha\|^2 >_\alpha + 2 < \mathbf{u}_\alpha \cdot \mathbf{w}_\alpha >_\alpha + < \|\mathbf{w}_\alpha\|^2 >_\alpha . \tag{A.49}$$

Once $\mathbf{u}_\alpha$ is already an average value, the second term of the right side is null since $< \mathbf{u}_\alpha \cdot \mathbf{w}_\alpha >_\alpha = \mathbf{u}_\alpha \cdot < \mathbf{w}_\alpha >_\alpha = 0$. Thus, using the definition for T_α presented in Equation 2.14a, the average value is calculated as:

$$< \|\mathbf{v}\|^2 >_\alpha = \|\mathbf{u}_\alpha\|^2 + \frac{N_d k}{m_\alpha} T_\alpha. \tag{A.50}$$

Substituting this value into the Equation A.47, the integral IX is calculated as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \frac{\partial f_\alpha}{\partial t} d^3v = \frac{\partial n_\alpha m_\alpha \|\mathbf{u}_\alpha\|^2}{\partial t} + \frac{\partial N_d k n_\alpha T_\alpha}{\partial t} \tag{A.51}$$

Integral X

Using the derivative product rule, the integral X can be splitted into:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 (\mathbf{v} \cdot \nabla_r) f_\alpha d^3v = m_\alpha \int_{\mathbf{v}} \nabla_r \cdot \left(f_\alpha \|\mathbf{v}\|^2 \mathbf{v} \right) d^3v - m_\alpha \int_{\mathbf{v}} f_\alpha \nabla_r \cdot \left(\|\mathbf{v}\|^2 \mathbf{v} \right) d^3v. \tag{A.52}$$

Considering that the phase space variables $\mathbf{v}$ and $\mathbf{r}$ are independent, the operator ∇_r treats the variable $\mathbf{v}$ as a constant. Therefore the second integral of the right side is null. Applying the Leibniz's rule into the first integral, the integral X is rewritten as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 (\mathbf{v} \cdot \nabla_r) f_\alpha d^3v = m_\alpha \nabla_r \cdot \int_{\mathbf{v}} f_\alpha \|\mathbf{v}\|^2 \mathbf{v} d^3v. \tag{A.53}$$

Using the average notation presented in Equation 2.9, the integral in the right side is rewritten as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 (\mathbf{v} \cdot \nabla_r) f_\alpha d^3v = n_\alpha m_\alpha \nabla_r \cdot < \|\mathbf{v}\|^2 \mathbf{v} >_\alpha . \tag{A.54}$$

Decomposing the velocity term $\mathbf{v}$ as discussed in Equation 2.11, the average value became:

$$< \|\mathbf{v}\|^2 \mathbf{v} >_\alpha = < \|\mathbf{u}_\alpha + \mathbf{w}_\alpha\|^2 (\mathbf{u}_\alpha + \mathbf{w}_\alpha) >_\alpha, \tag{A.55}$$

which is equivalent to

$$< \|\mathbf{v}\|^2 \mathbf{v} >_\alpha = < (\mathbf{u}_\alpha + \mathbf{w}_\alpha) \cdot (\mathbf{u}_\alpha + \mathbf{w}_\alpha) (\mathbf{u}_\alpha + \mathbf{w}_\alpha) >_\alpha . \tag{A.56}$$

This average value can be distributed as:

$$\begin{aligned} < \|\mathbf{v}\|^2 \mathbf{v} >_\alpha = & < \|\mathbf{u}_\alpha\|^2 \mathbf{u}_\alpha >_\alpha + < 2 (\mathbf{u}_\alpha \cdot \mathbf{w}_\alpha) \mathbf{u}_\alpha >_\alpha + < \|\mathbf{w}_\alpha\|^2 \mathbf{u}_\alpha >_\alpha \\ & + < \|\mathbf{u}_\alpha\|^2 \mathbf{w}_\alpha >_\alpha + < 2 (\mathbf{u}_\alpha \cdot \mathbf{w}_\alpha) \mathbf{w}_\alpha >_\alpha + < \|\mathbf{w}_\alpha\|^2 \mathbf{w}_\alpha >_\alpha . \end{aligned} \tag{A.57}$$

Considering that $\mathbf{u}_\alpha$ is already an average value, these terms are rewritten as:

$$< \|\mathbf{v}\|^2\mathbf{v} >_\alpha = \|\mathbf{u}_\alpha\|^2\mathbf{u}_\alpha + 2\mathbf{u}_\alpha\cdot < \mathbf{w}_\alpha >_\alpha \mathbf{u}_\alpha + < \|\mathbf{w}_\alpha\|^2 >_\alpha \mathbf{u}_\alpha + \|\mathbf{u}_\alpha\|^2 < \mathbf{w}_\alpha >_\alpha + 2\mathbf{u}_\alpha\cdot < \mathbf{w}_\alpha\mathbf{w}_\alpha >_\alpha + < \|\mathbf{w}_\alpha\|^2\mathbf{w}_\alpha >_\alpha . \tag{A.58}$$

Using the definitions presented in the Equations 2.13, 2.14a, 2.14b and 2.14c, the average value $< \|\mathbf{v}\|^2\mathbf{v} >_\alpha$ became:

$$< \|\mathbf{v}\|^2\mathbf{v} >_\alpha = \|\mathbf{u}_\alpha\|^2\mathbf{u}_\alpha + \frac{N_d k}{m_\alpha}T_\alpha\mathbf{u}_\alpha + \frac{2\mathbf{u}_\alpha\cdot\mathbb{P}_\alpha}{n_\alpha m_\alpha} + \frac{2\mathbf{h}_\alpha}{n_\alpha m_\alpha}. \tag{A.59}$$

Substituting this value into Equation A.54, the integral X is calculated as:

$$m_\alpha\int_{\mathbf{v}}\|\mathbf{v}\|^2(\mathbf{v}\cdot\nabla_r)f_\alpha d^3v = \nabla_r\cdot\left(n_\alpha m_\alpha\|\mathbf{u}_\alpha\|^2\mathbf{u}_\alpha + N_d k n_\alpha T_\alpha\mathbf{u}_\alpha + 2\mathbf{u}_\alpha\cdot\mathbb{P}_\alpha + 2\mathbf{h}_\alpha\right). \tag{A.60}$$

Integral XI

Using the derivative product rule, the integral XI can be splitted into:

$$m_\alpha\int_{\mathbf{v}}\|\mathbf{v}\|^2(\mathbf{a}\cdot\nabla_v f_\alpha)\,d^3v = m_\alpha\int_{\mathbf{v}}\nabla_v\cdot\left(f_\alpha\|\mathbf{v}\|^2\mathbf{a}\right)d^3v - m_\alpha\int_{\mathbf{v}}f_\alpha\nabla_v\cdot\left(\|\mathbf{v}\|^2\mathbf{a}\right)d^3v. \tag{A.61}$$

Applying the Gauss' divergence theorem at the first integral in the right side, the integral XI is rewritten as:

$$m_\alpha\int_{\mathbf{v}}\|\mathbf{v}\|^2(\mathbf{a}\cdot\nabla_v f_\alpha)\,d^3v = m_\alpha\oint_{\mathcal{S}}\left(f_\alpha\|\mathbf{v}\|^2\mathbf{a}\right)\cdot\mathbf{n}d\mathcal{S} - m_\alpha\int_{\mathbf{v}}f_\alpha\nabla_v\cdot\left(\|\mathbf{v}\|^2\mathbf{a}\right)d^3v. \tag{A.62}$$

The terms of this surface integral vanishes as $\mathbf{v}\to\infty$ due to the Maxwellian distribution. Hence, the integral XI became:

$$m_\alpha\int_{\mathbf{v}}\|\mathbf{v}\|^2(\mathbf{a}\cdot\nabla_v f_\alpha)\,d^3v = -m_\alpha\int_{\mathbf{v}}f_\alpha\nabla_v\cdot\left(\|\mathbf{v}\|^2\mathbf{a}\right)d^3v. \tag{A.63}$$

Using the acceleration term $\mathbf{a}$ presented in Equation A.7, the Integral XI can be splitted into:

$$m_\alpha\int_{\mathbf{v}}\|\mathbf{v}\|^2(\mathbf{a}\cdot\nabla_v f_\alpha)\,d^3v = -q_\alpha\int_{\mathbf{v}}f_\alpha\nabla_v\cdot\left(\|\mathbf{v}\|^2\mathbf{E}\right)d^3v - q_\alpha\int_{\mathbf{v}}f_\alpha\nabla_v\cdot\left(\|\mathbf{v}\|^2\mathbf{v}\times\mathbf{B}\right)d^3v \tag{A.64}$$

The second integral at the right side is zero due to the operator ∇_v being perpendicular to $\mathbf{v} \times \mathbf{B}$. Thus:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -q_\alpha \int_{\mathbf{v}} f_\alpha \nabla_v \cdot \left(\|\mathbf{v}\|^2 \mathbf{E}\right) d^3v. \tag{A.65}$$

Using the product rule, the right side term is rewritten as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -q_\alpha \int_{\mathbf{v}} f_\alpha \|\mathbf{v}\|^2 \left(\nabla_v \cdot \mathbf{E}\right) d^3v - q_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{E} \cdot \nabla_v \|\mathbf{v}\|^2 d^3v. \tag{A.66}$$

The variable $\mathbf{E}$ is independent of $\mathbf{v}$, hence the first integral in the right side is zero. Therefore, the Integral XI became:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -q_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{E} \cdot \nabla_v \|\mathbf{v}\|^2 d^3v. \tag{A.67}$$

Applying the operator ∇_v in the term $\|\mathbf{v}\|^2$, the integral is rewritten as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -q_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{E} \cdot \left(2\mathbf{v} \cdot \nabla_v \mathbf{v}\right) d^3v. \tag{A.68}$$

By definition, the term $\nabla_v \mathbf{v}$ produces the identity matrix:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -2q_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{E} \cdot \left(\mathbf{v} \cdot \mathbb{I}\right) d^3v. \tag{A.69}$$

Taking the inner product of $\mathbf{v}$ and the identity matrix:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -2q_\alpha \int_{\mathbf{v}} f_\alpha \mathbf{E} \cdot \mathbf{v} d^3v. \tag{A.70}$$

Considering that $\mathbf{E}$ is independent of $\mathbf{v}$, the right side term is rewritten as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -2q_\alpha \mathbf{E} \cdot \int_{\mathbf{v}} f_\alpha \mathbf{v} d^3v. \tag{A.71}$$

Using the definition presented in Equation 2.12, the integral XI is calculated as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\mathbf{a} \cdot \nabla_v f_\alpha\right) d^3v = -2n_\alpha q_\alpha \mathbf{E} \cdot \mathbf{u}_\alpha. \tag{A.72}$$

Integral XII

Considering that the phase space velocity $\mathbf{v}$ is independent of t, the integral XII can be rewritten as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = m_\alpha \int_{\mathbf{v}} \left(\frac{\delta f_\alpha \|\mathbf{v}\|^2}{\delta t}\right)_{coll} d^3v. \tag{A.73}$$

The integral operator in the right side can be reordered, obtaining:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = \left(\frac{\delta}{\delta t} m_\alpha \int_{\mathbf{v}} f_\alpha \|\mathbf{v}\|^2 d^3v\right)_{coll}. \tag{A.74}$$

This integral term can be rewritten as an average value:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = \left(\frac{\delta}{\delta t} m_\alpha n_\alpha < \|\mathbf{v}\|^2 >_\alpha\right)_{coll} \tag{A.75}$$

Using the average value $< \|\mathbf{v}\|^2 >_\alpha$ as calculated in Equation A.50:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = \left(\frac{\delta}{\delta t} m_\alpha n_\alpha \left(\|\mathbf{u}_\alpha\|^2 + \frac{N_d k T_\alpha}{m_\alpha}\right)\right)_{coll}, \tag{A.76}$$

which can be splitted into:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = \left(\frac{\delta}{\delta t} m_\alpha n_\alpha \|\mathbf{u}_\alpha\|^2\right)_{coll} + \left(\frac{\delta}{\delta t} n_\alpha N_d k T_\alpha\right)_{coll}. \tag{A.77}$$

Using the derivative product rule in the first term of the right side, the Integral XII became:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = 2\mathbf{u}_\alpha \cdot \left(\frac{\delta}{\delta t} m_\alpha n_\alpha \mathbf{u}_\alpha\right)_{coll} + \left(\frac{\delta}{\delta t} n_\alpha N_d k T_\alpha\right)_{coll} \tag{A.78}$$

Using the definition of the term $\mathbf{A}_\alpha$, presented in Equation A.42, the integral XII is calculated as:

$$m_\alpha \int_{\mathbf{v}} \|\mathbf{v}\|^2 \left(\frac{\delta f_\alpha}{\delta t}\right)_{coll} d^3v = 2\mathbf{u}_\alpha \cdot \mathbf{A}_\alpha + 2Q_\alpha \tag{A.79}$$

where the term Q_α is defined as:

$$Q_\alpha := \frac{1}{2}\left(\frac{\delta}{\delta t} n_\alpha N_d k T_\alpha\right)_{coll}. \tag{A.80}$$

This term represents the generated heat per unit of volume of the particles α due to collisions with another particles. The plasma heating may occur due to many effects such as resistivity, viscosity, energy production from thermonuclear fusion, etc. In opposition, there are effects such as radiation, which causes heat loss.

Assembling the* 2^{nd} *moment integrals

Substituting the Equations A.51, A.60, A.72 and A.79 into the Equation A.44, and dividing those terms by two, the Equation 2.20c is obtained:

$$\frac{1}{2}\frac{\partial n_\alpha m_\alpha \|\mathbf{u}_\alpha\|^2}{\partial t} + \frac{\partial N_d k n_\alpha T_\alpha}{\partial t} + \nabla\cdot\left(\frac{1}{2} n_\alpha m_\alpha \|\mathbf{u}_\alpha\|^2 \mathbf{u}_\alpha + \frac{1}{2} N_d k n_\alpha T_\alpha \mathbf{u}_\alpha + \mathbf{u}_\alpha\cdot\mathbb{P}_\alpha + \mathbf{h}_\alpha\right) - n_\alpha q_\alpha \mathbf{E}\cdot\mathbf{u}_\alpha = \mathbf{u}_\alpha\cdot\mathbf{A}_\alpha + Q_\alpha \tag{A.81}$$

Alternatively, in Goedbloed and Poedts (2004), this equation is rewritten in the form:

$$\frac{N_d}{2} n_\alpha k\left(\frac{\partial T_\alpha}{\partial t} + \mathbf{u}_\alpha\cdot\nabla T_\alpha\right) + \mathbb{P}_\alpha : \nabla\mathbf{u}_\alpha + \nabla\cdot\mathbf{h}_\alpha = Q_\alpha \tag{A.82}$$

□

Theorem 2, page 15: Applying the Ideal MHD properties to the fluid formulation obtained in Section 2.2.2 produces the following equations:

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0 \tag{A.83a}$$

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla\cdot\left[\rho\mathbf{u}\mathbf{u} + \left(p + \frac{\mathbf{B}\cdot\mathbf{B}}{2}\right)\mathbb{I} - \mathbf{B}\mathbf{B}\right] = \mathbf{0} \tag{A.83b}$$

$$\frac{\partial\mathcal{E}}{\partial t} + \nabla\cdot\left[\left(\mathcal{E} + p + \frac{\mathbf{B}\cdot\mathbf{B}}{2}\right)\mathbf{u} - (\mathbf{u}\cdot\mathbf{B})\,\mathbf{B}\right] = 0 \tag{A.83c}$$

$$\frac{\partial\mathbf{B}}{\partial t} + \nabla\cdot[\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u}] = \mathbf{0}, \tag{A.83d}$$

where this system is completed with the internal energy equation, given by the

combination of the hydrodynamic and magnetic energies:

$$\mathcal{E} = \frac{p}{\gamma - 1} + \rho\frac{\mathbf{u}\cdot\mathbf{u}}{2} + \frac{\mathbf{B}\cdot\mathbf{B}}{2}, \tag{A.84}$$

and γ is the adiabatic index.

Proof. This proof is divided into four parts. The first three consists in summing the effects of every type of particle α in every equation obtained from the Theorem 1. The last part consist in obtaining an equation for the evolution of $\mathbf{B}$ in order to close the system.

Proof part 1: MHD Continuity equation

Summing the terms of the Equation 2.20a for every type of particle α among the plasma, the following relation is obtained:

$$\sum_{\alpha}\left[\frac{\partial n_\alpha m_\alpha}{\partial t} + \nabla\cdot(n_\alpha m_\alpha \mathbf{u}_\alpha)\right] = \sum_{\alpha} S_\alpha. \tag{A.85}$$

Assuming the mass conservation of the system, the sum $\sum_\alpha S_\alpha$ must vanish. Besides that, the summation in the left side can be rearranged as:

$$\frac{\partial \sum_\alpha n_\alpha m_\alpha}{\partial t} + \nabla\cdot\left(\sum_{\alpha} n_\alpha m_\alpha \mathbf{u}_\alpha\right) = 0 \tag{A.86}$$

Using the definitions presented in the Equations 3.1a and 3.1c, the Continuity equation for MHD is obtained as:

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0. \tag{A.87}$$

Proof part 2: MHD Momentum equation

Summing the terms of the Equation 2.20b for every type of particle α among the plasma, the following relation is obtained:

$$\sum_{\alpha}\left[\frac{\partial\left(n_\alpha m_\alpha \mathbf{u}_\alpha\right)}{\partial t} + \nabla\cdot\left[n_\alpha m_\alpha \mathbf{u}_\alpha\mathbf{u}_\alpha + \mathbb{P}_\alpha\right] - n_\alpha q_\alpha\left(\mathbf{E} + \mathbf{u}_\alpha\times\mathbf{B}\right)\right] = \sum_{\alpha}\mathbf{A}_\alpha \tag{A.88}$$

Assuming the momentum conservation after the collision of particles of different types, the sum $\sum_\alpha \mathbf{A}_\alpha$ vanishes. Besides that, the summation in the left side can be rearranged as:

$$\frac{\partial\left(\sum_\alpha n_\alpha m_\alpha \mathbf{u}_\alpha\right)}{\partial t} + \nabla\cdot\left[\sum_\alpha n_\alpha m_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha + \sum_\alpha \mathbb{P}_\alpha\right] - \mathbf{E}\sum_\alpha n_\alpha q_\alpha - \left(\sum_\alpha n_\alpha q_\alpha \mathbf{u}_\alpha\right)\times\mathbf{B} = \mathbf{0}. \tag{A.89}$$

According to the macroscopic neutrality assumption, the number of ions and electrons in the system are similar, *i.e.*, $n_i \approx n_e$. Considering that these particles have opposite charges, *i.e.*, $q_i = -q_e$, the sum $\sum_\alpha n_\alpha q_\alpha$ is approximately zero. Using the definitions presented in the Equations 3.1c and 3.1d, this equation is rewritten as:

$$\frac{\partial\left(\rho\mathbf{u}\right)}{\partial t} + \nabla\cdot\left[\sum_\alpha n_\alpha m_\alpha \mathbf{u}_\alpha \mathbf{u}_\alpha + \sum_\alpha \mathbb{P}_\alpha\right] - \mathbf{J}\times\mathbf{B} = \mathbf{0}, \tag{A.90}$$

where the current density $\mathbf{J}$ can be substituted using the Ampère's law presented in Equation 2.21c. Besides that, neglecting the terms multiplied by the electron mass and assuming the isotropic pressure, this equation became:

$$\frac{\partial\left(\rho\mathbf{u}\right)}{\partial t} + \nabla\cdot\left[\rho_i \mathbf{u}_i \mathbf{u}_i + \mathbb{I}\sum_\alpha p_\alpha\right] - \left(\nabla\times\mathbf{B}\right)\times\mathbf{B} = \mathbf{0}. \tag{A.91}$$

Assuming $\rho \approx \rho_i$ and $\mathbf{u} \approx \mathbf{u}_i$, then applying the pressure definition presented in Equation 3.1e:

$$\frac{\partial\left(\rho\mathbf{u}\right)}{\partial t} + \nabla\cdot\left[\rho\mathbf{u}\mathbf{u} + p\mathbb{I}\right] - \left(\nabla\times\mathbf{B}\right)\times\mathbf{B} = \mathbf{0}. \tag{A.92}$$

Using some vector calculus, the last term can be rewritten in a divergence form. Thus:

$$\frac{\partial\left(\rho\mathbf{u}\right)}{\partial t} + \nabla\cdot\left[\rho\mathbf{u}\mathbf{u} + p\mathbb{I}\right] - \nabla\cdot\left(\mathbf{B}\mathbf{B} - \frac{\mathbf{B}\cdot\mathbf{B}}{2}\mathbb{I}\right) = \mathbf{0}, \tag{A.93}$$

which can be rearranged in order to obtain the MHD momentum equation:

$$\frac{\partial\left(\rho\mathbf{u}\right)}{\partial t} + \nabla\cdot\left[\rho\mathbf{u}\mathbf{u} + \left(p + \frac{\mathbf{B}\cdot\mathbf{B}}{2}\right)\mathbb{I} - \mathbf{B}\mathbf{B}\right] = \mathbf{0}. \tag{A.94}$$

Proof part 3: MHD Energy equation

Summing the terms of the Equation 2.20c for every type of particle α among the

plasma, the following relation is obtained:

$$\sum_{\alpha}\Big[\frac{1}{2}\frac{\partial n_\alpha m_\alpha\|\mathbf{u}_\alpha\|^2}{\partial t}+\frac{\partial N_d k n_\alpha T_\alpha}{\partial t}+\nabla\cdot\Big(\frac{1}{2}n_\alpha m_\alpha\|\mathbf{u}_\alpha\|^2\mathbf{u}_\alpha+\frac{1}{2}N_d k n_\alpha T_\alpha\mathbf{u}_\alpha \\ +\mathbf{u}_\alpha\cdot\mathbb{P}_\alpha+\mathbf{h}_\alpha\Big)-n_\alpha q_\alpha\mathbf{E}\cdot\mathbf{u}_\alpha\Big]=\sum_{\alpha}\left[\mathbf{u}_\alpha\cdot\mathbf{A}_\alpha+Q_\alpha\right] \tag{A.95}$$

Considering $\mathbf{u}\approx u_i\approx u_e$, the term $\sum_\alpha[\mathbf{u}_\alpha\cdot\mathbf{A}_\alpha]$ can be neglected due to the momentum conservation assumption. Furthermore, the diffusive term $\mathbf{h}_\alpha$ and the term Q_α are also neglected due to the adiabatic assumption. Thus, this equation can be rearranged as:

$$\frac{1}{2}\frac{\partial\sum_\alpha n_\alpha m_\alpha\|\mathbf{u}_\alpha\|^2}{\partial t}+\frac{N_d}{2}\frac{\partial\sum_\alpha k n_\alpha T_\alpha}{\partial t}+\nabla\cdot\Big(\frac{1}{2}\sum_\alpha n_\alpha m_\alpha\|\mathbf{u}_\alpha\|^2\mathbf{u}_\alpha+ \\ \frac{N_d}{2}\sum_\alpha k n_\alpha T_\alpha\mathbf{u}_\alpha+\sum_\alpha\mathbf{u}_\alpha\cdot\mathbb{P}_\alpha\Big)-\mathbf{E}\cdot\sum_\alpha n_\alpha q_\alpha\mathbf{u}_\alpha=0 \tag{A.96}$$

Neglecting the terms multiplied by the electron mass and assuming the isotropic pressure, this equation became:

$$\frac{1}{2}\frac{\partial n_i m_i\|\mathbf{u}_i\|^2}{\partial t}+\frac{N_d}{2}\frac{\partial\sum_\alpha k n_\alpha T_\alpha}{\partial t}+\nabla\cdot\Big(\frac{1}{2}n_i m_i\|\mathbf{u}_i\|^2\mathbf{u}_i+ \\ \frac{N_d}{2}\sum_\alpha k n_\alpha T_\alpha\mathbf{u}_\alpha+\sum_\alpha\mathbf{u}_\alpha\cdot p_\alpha\mathbb{I}\Big)-\mathbf{E}\cdot\sum_\alpha n_\alpha q_\alpha\mathbf{u}_\alpha=0 \tag{A.97}$$

Assuming $\rho\approx\rho_i$ and $\mathbf{u}\approx\mathbf{u}_i$ from the Equations 3.1a and 3.1c and using the pressure definition presented in Equation 2.17:

$$\frac{1}{2}\frac{\partial\rho\|\mathbf{u}\|^2}{\partial t}+\frac{N_d}{2}\frac{\partial\sum_\alpha p_\alpha}{\partial t}+\nabla\cdot\Big(\frac{1}{2}\rho\|\mathbf{u}\|^2\mathbf{u}+\frac{N_d}{2}\sum_\alpha p_\alpha\mathbf{u}_\alpha+\sum_\alpha\mathbf{u}_\alpha\cdot p_\alpha\mathbb{I}\Big)-\mathbf{E}\cdot\mathbf{J}=0 \tag{A.98}$$

Using the pressure definition presented in Equation 3.1e, this equation can be rewritten as:

$$\frac{1}{2}\frac{\partial\rho\|\mathbf{u}\|^2}{\partial t}+\frac{N_d}{2}\frac{\partial p}{\partial t}+\nabla\cdot\Big(\frac{1}{2}\rho\|\mathbf{u}\|^2\mathbf{u}+\frac{N_d}{2}p\mathbf{u}+p\mathbf{u}\Big)-\mathbf{E}\cdot\mathbf{J}=0 \tag{A.99}$$

For simplicity, the term $\frac{N_d}{2}$ is calculated based on its inverse:

$$\frac{2}{N_d} = \frac{2}{N_d} + \frac{N_d}{N_d} - \frac{N_d}{N_d} = \frac{2+N_d}{N_d} - 1 = \gamma - 1, \tag{A.100}$$

hence

$$\frac{N_d}{2} = \frac{1}{\gamma - 1}. \tag{A.101}$$

Substituting this value into Equation A.99 and reordering some terms, the following relation is obtained:

$$\frac{1}{2}\frac{\partial \rho \|\mathbf{u}\|^2}{\partial t} + \frac{1}{\gamma - 1}\frac{\partial p}{\partial t} + \nabla \cdot \left[\left(\frac{\rho \|\mathbf{u}\|^2}{2} + \frac{p}{\gamma - 1} + p\right)\mathbf{u}\right] - \mathbf{E} \cdot \mathbf{J} = 0. \tag{A.102}$$

The last term in this equation can be rewritten using the Ampère's law, presented in Equation 2.21c:

$$\mathbf{E} \cdot \mathbf{J} = \mathbf{E} \cdot (\nabla \times \mathbf{B}). \tag{A.103}$$

The product in the left side can be decomposed by using vector calculus identities, obtaining:

$$\mathbf{E} \cdot \mathbf{J} = -\nabla \cdot (\mathbf{E} \times \mathbf{B}) + (\nabla \times \mathbf{E}) \cdot \mathbf{B} \tag{A.104}$$

Applying the Faraday law, presented in Equation 2.21b, this relation became:

$$\mathbf{E} \cdot \mathbf{J} = -\nabla \cdot (\mathbf{E} \times \mathbf{B}) - \frac{\partial \mathbf{B}}{\partial t} \cdot \mathbf{B}, \tag{A.105}$$

which can be rewritten using the derivative product rule as:

$$\mathbf{E} \cdot \mathbf{J} = -\nabla \cdot (\mathbf{E} \times \mathbf{B}) - \frac{1}{2}\frac{\partial \|\mathbf{B}\|^2}{\partial t} \tag{A.106}$$

Substituting this term into Equation A.102:

$$\frac{1}{2}\frac{\partial \rho \|\mathbf{u}\|^2}{\partial t} + \frac{1}{\gamma - 1}\frac{\partial p}{\partial t} + \nabla \cdot \left[\left(\frac{\rho \|\mathbf{u}\|^2}{2} + \frac{p}{\gamma - 1} + p\right)\mathbf{u}\right] + \nabla \cdot (\mathbf{E} \times \mathbf{B}) + \frac{1}{2}\frac{\partial \|\mathbf{B}\|^2}{\partial t} = 0 \tag{A.107}$$

The terms of this equation can be reordered as:

$$\frac{\partial}{\partial t}\left(\frac{\rho\|\mathbf{u}\|^2}{2}+\frac{p}{\gamma-1}+\frac{\|\mathbf{B}\|^2}{2}\right)+\nabla\cdot\left[\left(\frac{\rho\|\mathbf{u}\|^2}{2}+\frac{p}{\gamma-1}+p\right)\mathbf{u}+\mathbf{E}\times\mathbf{B}\right]=0 \tag{A.108}$$

The terms inside the time derivative defines the total internal energy:

$$\mathcal{E}:=\frac{p}{\gamma-1}+\rho\frac{\mathbf{u}\cdot\mathbf{u}}{2}+\frac{\mathbf{B}\cdot\mathbf{B}}{2} \tag{A.109}$$

Substituting this definition into Equation A.108, the relation became:

$$\frac{\partial\mathcal{E}}{\partial t}+\nabla\cdot\left[\left(\frac{\rho\|\mathbf{u}\|^2}{2}+\frac{p}{\gamma-1}+p\right)\mathbf{u}+\mathbf{E}\times\mathbf{B}\right]=0 \tag{A.110}$$

The electric field $\mathbf{E}$ can be rewritten using the Ohm's law presented in Equation 2.22, obtaining:

$$\frac{\partial\mathcal{E}}{\partial t}+\nabla\cdot\left[\left(\frac{\rho\|\mathbf{u}\|^2}{2}+\frac{p}{\gamma-1}+p\right)\mathbf{u}+(\eta\mathbf{J}-\mathbf{u}\times\mathbf{B})\times\mathbf{B}\right]=0 \tag{A.111}$$

Distributing the last term and applying a vector calculus identity, this equation became:

$$\frac{\partial\mathcal{E}}{\partial t}+\nabla\cdot\left[\left(\frac{\rho\|\mathbf{u}\|^2}{2}+\frac{p}{\gamma-1}+p\right)\mathbf{u}+(\eta\mathbf{J})\times\mathbf{B}-(\mathbf{u}\cdot\mathbf{B})\,\mathbf{B}+\|\mathbf{B}\|^2\mathbf{u}\right]=0. \tag{A.112}$$

These terms can be reordered as:

$$\frac{\partial\mathcal{E}}{\partial t}+\nabla\cdot\left[\left(\frac{\rho\|\mathbf{u}\|^2}{2}+\frac{p}{\gamma-1}+p+\|\mathbf{B}\|^2\right)\mathbf{u}-(\mathbf{u}\cdot\mathbf{B})\,\mathbf{B}+(\eta\mathbf{J})\times\mathbf{B}\right]=0. \tag{A.113}$$

Substituting the definition of internal energy $\mathcal{E}$, given in Equation A.109, into this equation produces:

$$\frac{\partial\mathcal{E}}{\partial t}+\nabla\cdot\left[\left(\mathcal{E}+p+\frac{\|\mathbf{B}\|^2}{2}\right)\mathbf{u}-(\mathbf{u}\cdot\mathbf{B})\,\mathbf{B}+(\eta\mathbf{J})\times\mathbf{B}\right]=0 \tag{A.114}$$

Neglecting the resistivity terms, the MHD energy equation is obtained:

$$\frac{\partial \mathcal{E}}{\partial t} + \nabla \cdot \left[\left(\mathcal{E} + p + \frac{\|\mathbf{B}\|^2}{2} \right) \mathbf{u} - (\mathbf{u} \cdot \mathbf{B})\, \mathbf{B} \right] = 0 \tag{A.115}$$

Proof part 4: MHD Induction equation

The MHD Induction equation is obtained by taking the curl of the Ohm's law:

$$\nabla \times \mathbf{E} + \nabla \times (\mathbf{u} \times \mathbf{B}) = \nabla \times (\eta \mathbf{J})\,. \tag{A.116}$$

Using the Faraday's law, presented in Equation 2.21b, and some vector calculus identities, this equation is rewritten as:

$$-\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{B}\mathbf{u} - \mathbf{u}\mathbf{B}) = \nabla \times (\eta \mathbf{J})\,. \tag{A.117}$$

Neglecting the resistive term and reordering the terms, the MHD Induction equation is obtained:

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u}) = \mathbf{0}. \tag{A.118}$$

□

Theorem 3, page 15: The pressure equation for Ideal MHD is given by:

$$\frac{\partial p}{\partial t} + (\mathbf{u} \cdot \nabla)\, p = -\gamma p \nabla \cdot \mathbf{u} \tag{A.119}$$

Proof. The pressure equation for Ideal MHD is obtained using an analogous procedure as performed to obtain the MHD energy equation. However, the pressure equation is derived from the alternative formulation of the second moment of the Boltzmann Equation, as presented in Equation A.82. Summing the terms of the Equation A.82 for every type of particle α among the plasma, the following relation is obtained:

$$\sum_{\alpha} \left[\frac{N_d}{2} n_\alpha k \left(\frac{\partial T_\alpha}{\partial t} + \mathbf{u}_\alpha \cdot \nabla T_\alpha \right) + \mathbb{P}_\alpha : \nabla \mathbf{u}_\alpha + \nabla \cdot \mathbf{h}_\alpha \right] = \sum_{\alpha} Q_\alpha \tag{A.120}$$

This equation is rearranged by multiplying its terms for $\frac{2}{N_d}$ and neglecting the terms

$\mathbf{h}_\alpha$ and Q_α, obtaining:

$$\sum_\alpha n_\alpha k \frac{\partial T_\alpha}{\partial t} + \sum_\alpha n_\alpha k \mathbf{u}_\alpha \cdot \nabla T_\alpha + \frac{2}{N_d} \sum_\alpha \mathbb{P}_\alpha : \nabla \mathbf{u}_\alpha = 0. \tag{A.121}$$

Assuming the isotropic pressure and splitting first term using the derivative product rule, the equation became:

$$\sum_\alpha \left(\frac{\partial n_\alpha k T_\alpha}{\partial t} - k T_\alpha \frac{\partial n_\alpha}{\partial t} \right) + \sum_\alpha n_\alpha k \mathbf{u}_\alpha \cdot \nabla T_\alpha + \frac{2}{N_d} \sum_\alpha p_\alpha \mathbb{I} : \nabla \mathbf{u}_\alpha = 0 \tag{A.122}$$

Using the definition for p_α presented in Equation 2.17 and a relation for the term $\frac{\partial n_\alpha}{\partial t}$ obtained from Equation 2.20a:

$$\frac{\partial \sum_\alpha p_\alpha}{\partial t} - \sum_\alpha k T_\alpha \left(\frac{S_\alpha}{m_\alpha} - \nabla \cdot (n_\alpha \mathbf{u}_\alpha) \right) + \sum_\alpha n_\alpha k \mathbf{u}_\alpha \cdot \nabla T_\alpha + \frac{2}{N_d} \sum_\alpha p_\alpha \nabla \cdot \mathbf{u}_\alpha = 0 \tag{A.123}$$

Using the pressure definition presented in Equation 3.1e and distributing the summation in the second term, this equation can be rewritten as:

$$\frac{\partial p}{\partial t} - \sum_\alpha \frac{S_\alpha}{m_\alpha} k T_\alpha + \underbrace{\sum_\alpha k T_\alpha \nabla \cdot (n_\alpha \mathbf{u}_\alpha)}_{I} + \underbrace{\sum_\alpha n_\alpha k \mathbf{u}_\alpha \cdot \nabla T_\alpha}_{II} + \frac{2}{N_d} \sum_\alpha p_\alpha \nabla \cdot \mathbf{u}_\alpha = 0 \tag{A.124}$$

The terms S_α are neglected due to the mass conservation assumption. Furthermore, the terms I and II can be grouped by the derivative product rule, obtaining:

$$\frac{\partial p}{\partial t} + \sum_\alpha \nabla \cdot (n_\alpha k T_\alpha \mathbf{u}_\alpha) + \frac{2}{N_d} \sum_\alpha p_\alpha \nabla \cdot \mathbf{u}_\alpha = 0. \tag{A.125}$$

Using the pressure definition presented in Equation 3.1e, this equation became:

$$\frac{\partial p}{\partial t} + \sum_\alpha \nabla \cdot (p_\alpha \mathbf{u}_\alpha) + \frac{2}{N_d} \sum_\alpha p_\alpha \nabla \cdot \mathbf{u}_\alpha = 0 \tag{A.126}$$

Assuming $\mathbf{u} \approx u_i \approx u_e$, the second term is rewritten, obtaining:

$$\frac{\partial p}{\partial t} + \nabla \cdot (\mathbf{u} \sum_\alpha p_\alpha) + \frac{2}{N_d} \sum_\alpha p_\alpha \nabla \cdot \mathbf{u}_\alpha = 0 \tag{A.127}$$

Using the definition for the single fluid isotropic pressure:

$$\frac{\partial p}{\partial t} + \nabla \cdot (p\mathbf{u}) + \frac{2}{N_d} p \nabla \cdot \mathbf{u} = 0. \tag{A.128}$$

The second term can be decomposed by the derivative product rule, obtaining:

$$\frac{\partial p}{\partial t} + p\nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla p + \frac{2}{N_d} p \nabla \cdot \mathbf{u} = 0 \tag{A.129}$$

These terms can be rearranged as:

$$\frac{\partial p}{\partial t} + \mathbf{u} \cdot \nabla p + \frac{2 + N_d}{N_d} p \nabla \cdot \mathbf{u} = 0 \tag{A.130}$$

Using the definition of the adiabatic index γ, presented in Equation 2.15, the Ideal MHD pressure equation is obtained:

$$\frac{\partial p}{\partial t} + \mathbf{u} \cdot \nabla p + \gamma p \nabla \cdot \mathbf{u} = 0 \tag{A.131}$$

□

Theorem 4, page 18: The resistive induction equation, obtained in Equation A.117, can be rewritten in the conservative form as:

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot \left[\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u} + \eta \left((\nabla \mathbf{B})^T - \nabla \mathbf{B} \right) \right] = 0 \tag{A.132}$$

Proof. Applying the Ampère's law into Equation A.117, the induction equation became:

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u}) = -\nabla \times (\eta \nabla \times \mathbf{B}) . \tag{A.133}$$

To represent this equation in the conservative formulation, the source term must be written in the form $\nabla \cdot \mathbb{R}$, then be included in the flux term. The matrix $\mathbb{R}$ is constructed based on the expansion of the term $\nabla \times (\eta \nabla \times \mathbf{B})$.

Initially, is considered the curl of $\mathbf{B}$:

$$\nabla \times \mathbf{B} = \mathbf{i} \left(\frac{\partial B_z}{\partial y} - \frac{\partial B_y}{\partial z} \right) + \mathbf{j} \left(\frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x} \right) + \mathbf{k} \left(\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y} \right) \tag{A.134}$$

In order to simplify the calculations, the values ξ_1, ξ_2 and ξ_3 are used to represent the components of the curl of $\mathbf{B}$ so that:

$$\nabla \times \mathbf{B} = \xi_1 \mathbf{i} + \xi_2 \mathbf{j} + \xi_3 \mathbf{k}. \tag{A.135}$$

Using this formulation, the source term is rewritten:

$$\begin{aligned} \nabla \times (\eta \nabla \times \mathbf{B}) = \mathbf{i} \left[\frac{\partial (\eta \xi_3)}{\partial y} - \frac{\partial (\eta \xi_2)}{\partial z} \right] + \mathbf{j} \left[\frac{\partial (\eta \xi_1)}{\partial z} - \frac{\partial (\eta \xi_3)}{\partial x} \right] \\ + \mathbf{k} \left[\frac{\partial (\eta \xi_2)}{\partial x} - \frac{\partial (\eta \xi_1)}{\partial y} \right] \end{aligned} \tag{A.136}$$

These terms can be expressed in the form of a row matrix as:

$$\nabla \times (\eta \nabla \times \mathbf{B}) = \left[\frac{\partial (\eta \xi_3)}{\partial y} - \frac{\partial (\eta \xi_2)}{\partial z}; \; \frac{\partial (\eta \xi_1)}{\partial z} - \frac{\partial (\eta \xi_3)}{\partial x}; \; \frac{\partial (\eta \xi_2)}{\partial x} - \frac{\partial (\eta \xi_1)}{\partial y} \right]. \tag{A.137}$$

This column matrix can be rewritten as the divergence of a 3×3 matrix:

$$\nabla \times (\eta \nabla \times \mathbf{B}) = \nabla \cdot \begin{bmatrix} 0 & -\eta \xi_3 & \eta \xi_2 \\ \eta \xi_3 & 0 & -\eta \xi_1 \\ -\eta \xi_2 & \eta \xi_1 & 0 \end{bmatrix}. \tag{A.138}$$

Substituting the values ξ into the matrix:

$$\nabla \times (\eta \nabla \times \mathbf{B}) = \nabla \cdot \begin{bmatrix} 0 & \eta \left(\frac{\partial B_x}{\partial y} - \frac{\partial B_y}{\partial x} \right) & \eta \left(\frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x} \right) \\ \eta \left(\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y} \right) & 0 & \eta \left(\frac{\partial B_y}{\partial z} - \frac{\partial B_z}{\partial y} \right) \\ \eta \left(\frac{\partial B_z}{\partial x} - \frac{\partial B_x}{\partial z} \right) & \eta \left(\frac{\partial B_z}{\partial y} - \frac{\partial B_y}{\partial z} \right) & 0 \end{bmatrix}. \tag{A.139}$$

The terms in the diagonal can be rewritten as:

$$\nabla \times (\eta \nabla \times \mathbf{B}) = \nabla \cdot \begin{bmatrix} \eta \left(\frac{\partial B_x}{\partial x} - \frac{\partial B_x}{\partial x} \right) & \eta \left(\frac{\partial B_x}{\partial y} - \frac{\partial B_y}{\partial x} \right) & \eta \left(\frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x} \right) \\ \eta \left(\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y} \right) & \eta \left(\frac{\partial B_y}{\partial y} - \frac{\partial B_y}{\partial y} \right) & \eta \left(\frac{\partial B_y}{\partial z} - \frac{\partial B_z}{\partial y} \right) \\ \eta \left(\frac{\partial B_z}{\partial x} - \frac{\partial B_x}{\partial z} \right) & \eta \left(\frac{\partial B_z}{\partial y} - \frac{\partial B_y}{\partial z} \right) & \eta \left(\frac{\partial B_z}{\partial z} - \frac{\partial B_z}{\partial z} \right) \end{bmatrix}. \tag{A.140}$$

This matrix is splitted into:

$$\nabla \times (\eta \nabla \times \mathbf{B}) = \nabla \cdot \left(\begin{bmatrix} \eta\frac{\partial B_x}{\partial x} & \eta\frac{\partial B_x}{\partial y} & \eta\frac{\partial B_x}{\partial z} \\ \eta\frac{\partial B_y}{\partial x} & \eta\frac{\partial B_y}{\partial y} & \eta\frac{\partial B_y}{\partial z} \\ \eta\frac{\partial B_z}{\partial x} & \eta\frac{\partial B_z}{\partial y} & \eta\frac{\partial B_z}{\partial z} \end{bmatrix} - \begin{bmatrix} \eta\frac{\partial B_x}{\partial x} & \eta\frac{\partial B_y}{\partial x} & \eta\frac{\partial B_z}{\partial x} \\ \eta\frac{\partial B_x}{\partial y} & \eta\frac{\partial B_y}{\partial y} & \eta\frac{\partial B_z}{\partial y} \\ \eta\frac{\partial B_x}{\partial z} & \eta\frac{\partial B_y}{\partial z} & \eta\frac{\partial B_z}{\partial z} \end{bmatrix} \right) \tag{A.141}$$

Using the vector notation, this term became:

$$\nabla \times (\eta \nabla \times \mathbf{B}) = \nabla \cdot \left[\eta \left[(\nabla \mathbf{B})^T - \nabla \mathbf{B} \right] \right]. \tag{A.142}$$

Substituting this source term in the Equation A.133,

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{uB} - \mathbf{Bu}) = -\nabla \cdot \left[\eta \left[(\nabla \mathbf{B})^T - \nabla \mathbf{B} \right] \right]. \tag{A.143}$$

Reordering these terms, the resistive induction equation in the conservative formulation is obtained:

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot \left[\mathbf{uB} - \mathbf{Bu} + \eta \left((\nabla \mathbf{B})^T - \nabla \mathbf{B} \right) \right] = 0 \tag{A.144}$$

□

ANNEX B - NERK LOCAL TIME STEPPING

In the following is annexed an article published in the Journal of Computational Physics that presents the proposed extension of the Local Time stepping approach to higher order formulations, together with the algorithms concerning the required synchronisations in time and a discussion concerning the numerical stability of these models. It is referred as (MOREIRA LOPES et al., 2019), namely:

- MOREIRA LOPES, M.; DOMINGUES, M.O.; SCHNEIDER, K.; MENDES, O. Local time-stepping for adaptive multiresolution using natural extensions for Runge–Kutta methods. **Journal of Computational Physics**, v. 382, p. 291-318, 2019.

Journal of Computational Physics 382 (2019) 291–318

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Local time-stepping for adaptive multiresolution using natural extension of Runge–Kutta methods

Müller Moreira Lopes [a,d,*], Margarete Oliveira Domingues [b,d], Kai Schneider [e], Odim Mendes [c,d]

[a] Graduate program in Applied Computing (CAP), Brazil
[b] Associate Laboratory of Applied Computing and Mathematics (LAC), Coordination of the Associated Laboratories (CTE), Brazil
[c] Space Geophysics Division (DGE), Coordination of Space Sciences (CEA), Brazil
[d] National Institute for Space Research (INPE), Av. dos Astronautas 1758, 12227-010 São José dos Campos, São Paulo, Brazil
[e] Institut de Mathématiques de Marseille (I2M), Aix-Marseille Université, CNRS, Centrale Marseille, 39 rue F. Joliot–Curie, 13453 Marseille Cedex 13, France

ARTICLE INFO

Article history:
Received 7 March 2018
Received in revised form 20 August 2018
Accepted 22 October 2018
Available online 17 January 2019

Keywords:
Multiresolution analysis
Finite volume
Local time-stepping
Runge–Kutta

ABSTRACT

A space–time fully adaptive multiresolution method for evolutionary non-linear partial differential equations is presented introducing an improved local time-stepping method. The space discretisation is based on classical finite volumes, endowed with cell average multiresolution analysis for triggering the dynamical grid adaptation. The explicit time scheme features a natural extension of Runge–Kutta methods which allow local time-stepping while guaranteeing accuracy. The use of a compact Runge–Kutta formulation permits further memory reduction. The precision and computational efficiency of the scheme regarding CPU time and memory compression are assessed for problems in one, two and three space dimensions. As application Burgers equation, reaction–diffusion equations and the compressible Euler equations are considered. The numerical results illustrate the efficiency and superiority of the proposed local time-stepping method with respect to the reference computations.

1. Introduction

Multiresolution (MR) methods improve the computational performance of numerical solvers of evolutionary partial differential equations when the solution exhibits localised structures, point-wise singularities, boundary layers, shocks, coherent vortices such as encountered in combustion and turbulent flow applications [12,19,26].

In these methods, the fast wavelet transform is the key ingredient to speed-up computations. The wavelet coefficients are employed to measure the local smoothness of the solution. Then, a thresholding strategy is used to remove non-significant coefficients, obtaining a grid adapted to the solution. This grid is coarser in smooth regions and finer there where structures and steep gradients are present. The locally refined grid can be rebuilt interactively to a regular grid with an expected error directly related to the chosen threshold. With the use of this adaptive grid, the number of interface flux computations during the time evolution can be significantly reduced, while controlling the error. The adaptive grid is checked before each

* Corresponding author at: National Institute for Space Research (INPE), Av. dos Astronautas 1758, 12227-010 São José dos Campos, São Paulo, Brazil.
E-mail addresses: muller.lopes@inpe.br (M. Moreira Lopes), margarete.domingues@inpe.br, margarete.oliveira.domingues@gmail.com (M.O. Domingues), kai.schneider@univ-amu.fr (K. Schneider), odim.mendes@inpe.br (O. Mendes).

https://doi.org/10.1016/j.jcp.2018.10.052
0021-9991/

time step to guarantee that it is sufficiently refined to represent possible new structures in the solution. Therefore, the adaptive grid is dynamically adapted to track the solution in scale and space.

The next step to improve the computational performance associated with these adaptive grids is to use a local time-stepping (LT) approach, especially when explicit time schemes are used. In this work, the combination of MR methods with the local time-stepping approach is denoted as MRLT. LT consists in performing the time evolution of each cell on the adaptive grid independently according to its required time step. Therefore, each cell must have a time step proportional to its refinement and consequently, a larger cell performs a larger time step.

Local time-stepping methods for space adaptive discretisations of partial differential equations have a long tradition, going back to the early work of Osher and Sanders [21]. Related to multiresolution, Müller and Stiriba [20] presented some general MRLT schemes that could be applied either to an explicit time scheme, based on Lagrange projection, or an implicit time scheme, for a reference finite-volume method in space. In [20] they applied LT to one-dimensional scalar conservation laws. Following this work, Coquel et al. [3] presented MRLT methods with both explicit and implicit Lagrange-projection schemes, the latter is the novelty concerning [20]. Hejazialhosseini et al. combined in [13] first and second order RK schemes to propose an LT approach for MR in blocks with finite volumes for multi-phase compressible flows implemented on multicore architectures.

More recently, LT methods have been proposed using different approaches for time interpolation required in the algorithms for providing values at intermediate time steps. In [24], the authors use a high-order Taylor type integrator, while a discontinuous Galerkin method with spectral elements is used for spatial discretization. In the context of finite elements methods, Rietmann et al. [25] proposed an energy conserving LT algorithm based on a second-order leap-frog scheme. Discussions on the proper choice of the time-step in LT schemes are still an important topic of discussion. To this end *a posteriori* error estimators are used in [1,18], while Gnedin et al. (2018) [11] investigate their effect by enforcing the CFL condition locally over every cell. In the context of finite volumes methods, the use of high-order schemes for local time-stepping on adaptive mesh refinement grids were previously discussed in [8]. In that work, the accuracy order in space is obtained through a WENO reconstruction, while the accuracy in the time discretization is achieved by a local space–time discontinuous Galerkin predictor method. This approach yields up to fourth order for compressible Euler equations in 2D. Similar and related works in this context were carried out in [15] and [2].

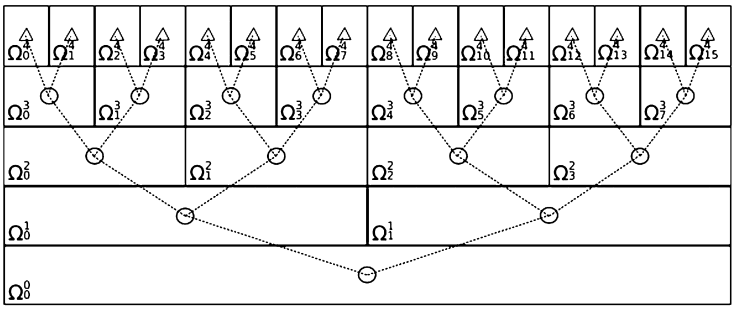
A detailed discussion on the stability of those MRLT schemes is presented in [14]. In the context of adaptive numerical methods for partial differential equations, the derivation of explicit LT methods based on standard RK schemes typically stays at orders smaller or equal to two [4–6]. The reason why LT methods are limited to low order in time is because a time synchronization is required; for a discussion we refer to [6]. Recently, higher-order LT schemes have been proposed in the context of discontinuous Galerkin methods. The works of Gassner et al. [9,10] in this context are based on natural continuous extensions for Runge–Kutta methods (NERK). Such schemes have been introduced initially by Zennaro in the late 1980's for solving general ODEs, with application to delay equations [30]. The idea is to interpolate the intermediate stages of the Runge–Kutta scheme to obtain the values at the requested intermediate time instants required for the time synchronisation in LT schemes. This method is also called Natural Continuous Extension in [30], Continuous Extension Runge–Kutta in [23], and in a general way Continuous Runge–Kutta, as discussed in [28].

Recently, an alternative for higher-order LT methods, again in the context of discontinuous Galerkin spectral element schemes, has been proposed by Winters and Kopriva [29]. The underlying ideas are Adams–Bashforth multi-step schemes.

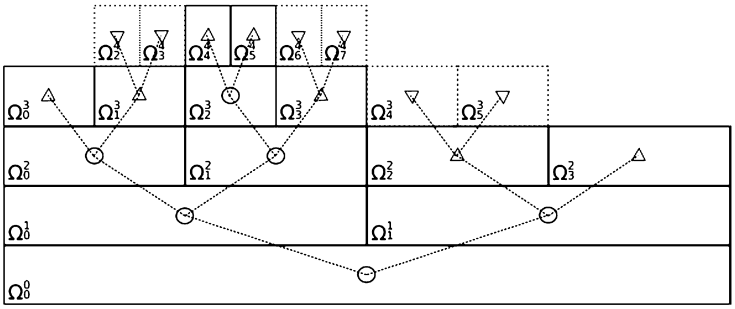
Moreover, Gassner et al. (2011) considered a family of explicit one-step time discretisations for finite volume (FV) and discontinuous Galerkin schemes, which are based on a predictor–corrector formulation [9].

The aim here is to use the idea of Gassner et al. [10] using NERK for the first time in the context of MRLT methods to perform the synchronisation. Thus, an improvement in time accuracy can be obtained. In the current work, only second and third-order schemes are used, but the extension to higher-order is in principle possible. Gassner et al. [10] discussed that NERK is 10% slower than the Cauchy–Kovalevskaya scheme. However, to work with the latter is complicated as the analytic solution of some nonlinear PDEs is required. The goal of the proposed approach is to perform simulations using NERK schemes. Therewith, the synchronisation required for the LT approach is possible, and this new class of MRLT methods, named MRLT/NERK, is created. This work presents the application and implementation of the MRLT/NERK method for the second (RK2) and third (RK3) order time evolution, named MRLT/NERK2 and MRLT/NERK3, respectively. The method proposed in this work is applied for solving the two-dimensional Burgers equation, one and three-dimensional reaction–diffusion problems and the two-dimensional compressible Euler equations. The obtained results and CPU times are compared with the MR method using classical RK2 and RK3 time evolution, named MR/RK2 and MR/RK3, respectively. The results are also compared with the MRLT approach given in [6] based on RK2 time evolution, named MRLT/RK2.

In Section 2, we summarise the adaptive multiresolution method proposed in [12]. Then, in Section 3, we discuss the Runge–Kutta methods and the NERK methods used in the current work to perform the proposed MRLT/NERK approach given in Section 4. A convergence analysis is conducted in Section 4.4. Performance comparisons, considering CPU time and errors, among the FV, MR and MRLT approaches given in [6] and the MRLT/NERK approach introduced here, are presented in Section 5. Conclusions are drawn in Section 6.



a) Grid hierarchy for an unidimensional domain.



b) Tree structure for an adapted grid.

Fig. 1. Dyadic grid hierarchy of MR methods and its implementation in a tree data structure. a) Example of nested unidimensional grids, the circles represent the internal nodes of the tree structure, while the triangles represent the leaves of the tree. b) Example of an adapted grid using the tree structure from a). The virtual leaves are represented by the triangles $\triangledown$. Adapted from [26].

2. Adaptive multiresolution methods using finite volumes

The following initial value problem for a vector-valued function $\mathbf{Q}(\mathbf{x}, t)$ written in divergence form is considered:

$$\frac{\partial \mathbf{Q}}{\partial t} = -\nabla \cdot \mathbf{F}(\mathbf{Q}, \nabla \mathbf{Q}) + \mathbf{S}(\mathbf{Q}), \quad \text{for} \quad (\mathbf{x}, t) \in \Omega \times [0, +\infty),\ \Omega \subset \mathbb{R}^d. \tag{1}$$

This problem is given in d space dimensions, completed with initial conditions $\mathbf{Q}(\mathbf{x}, t = 0) = \mathbf{Q}_0(\mathbf{x})$ and appropriate boundary conditions. The terms $\nabla \cdot \mathbf{F}(\mathbf{Q}, \nabla \mathbf{Q})$ and $\mathbf{S}(\mathbf{Q})$ denote the divergence and source term, respectively. The flux $\mathbf{F}$ can be decomposed into advective and diffusive contributions, *i.e.* $\mathbf{F}(\mathbf{Q}, \nabla \mathbf{Q}) = \mathbf{f}(\mathbf{Q}) - \nu \nabla \mathbf{Q}$, where the diffusion coefficient ν is positive and assumed to be constant.

To discretise Equation (1) in space, we use a classical finite volume formulation written in standard form. The domain Ω corresponds to a rectangular parallelepiped in $d = 1$, 2 or 3 dimensions in Cartesian geometry. It is partitioned into cells $(\Omega_i)_{i \in \Lambda}$, $\Lambda = \{0, \ldots, i_{\max}\}$ with $\Omega = \bigcup_i \Omega_i$.

Defining the volume of the cell by $|\Omega_i| = \int_{\Omega_i} d\mathbf{x}$, we compute the cell-average $\bar{\mathbf{q}}_i(t)$ of a given quantity $\mathbf{Q}$ on Ω_i at time instant t by,

$$\bar{\mathbf{q}}_i(t) = \frac{1}{|\Omega_i|} \int_{\Omega_i} \mathbf{Q}(\mathbf{x}, t)\, d\mathbf{x}.$$

Considering the one-dimensional case, Ω_i is an interval $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ of length $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$. Integrating Equation (1) on Ω_i then yields:

$$\frac{d\bar{\mathbf{q}}_i}{dt}(t) = -\frac{1}{\Delta x_i}\left(\bar{\mathbf{F}}_{i+\frac{1}{2}} - \bar{\mathbf{F}}_{i-\frac{1}{2}}\right) + \bar{\mathbf{S}}_i, \tag{2}$$

where $\bar{\mathbf{F}}$ is the numerical flux and $\bar{\mathbf{S}}_i$ is the source term of the cell Ω_i. This formulation can be extended to two- and three dimensional problems correspondingly.

The source term is approximated by $\bar{\mathbf{S}}_i \approx \mathbf{S}(\bar{\mathbf{q}}_i)$, which yields also second-order accuracy in case of a linear source term.

The adaptive multiresolution (MR) analysis in the cell average context, proposed by Harten [12], consists in decomposing the cell-averages of the solution into a multilevel representation. This representation, illustrated in Fig. 1a, consists of a hierarchy of nested grids Ω^ℓ, where ℓ is the grid refinement level. Each grid Ω^ℓ consists of a regular grid, as defined for the FV formulation, with $2^{d\ell}$ cells. A cell of refinement level ℓ and position i is denoted by Ω_i^ℓ.

Fig. 1a shows the implementation of this structure as a binary tree, where the nodes of level ℓ generate the grid Ω^ℓ. For the two- and three-dimensional cases, this idea is extended by using a quadtree and an octree, respectively.

This sequence of nested grids corresponds to a scheme where a finer scale represents its subsequent coarser scale plus a sum of details between these levels. These details are the wavelet coefficients. The construction of the adapted grid in the MR method consists in representing the grid using only cells with significant wavelet coefficients. For this, the leaves of the most refined level are checked, if their parent cell has a significant wavelet coefficient, larger than a predetermined threshold ϵ, if not, the leaves are deleted. This process is repeated recursively starting from the finest level, going to coarser and coarser levels. The procedure to obtain the projection of the solution on the coarser levels and the wavelet coefficients is given in Section 2.1.

The grid generation procedure is mathematically supported by the fact that the magnitude of the wavelet coefficients are small in regions where the solution is smooth, while they are significant in regions where the solution exhibits steep gradients. Hence, high compression rates are expected when only a small number of cell-averages is present on the finest scales.

In this work, the data structure for representing the solution is organized as a dynamic graded tree. This tree organization requires that no hole is admitted inside the tree, which means that the connectivity in the tree structure has to be ensured. Moreover, the tree can change in time to track the space-scale evolution of the solution as nodes can be added or removed while guaranteeing its gradedness.

This dynamic graded tree organization implies that neighbours of each cell can have a difference of one refinement level at most. This restriction allows the use of virtual leaves for the flux computations between leaves at different refinement levels. These virtual leaves are auxiliary leaves placed as children nodes of the leaves which have an interface with finer leaves. Their values are predicted using the prediction procedure used to compute the wavelet coefficients. A unidimensional adaptive grid represented in a graded tree structure is shown in Fig. 1b.

The flux computations in the MR scheme are performed individually for each leaf. The numerical fluxes are computed between cells at the same refinement level. Using the adaptive grid of Fig. 1b, the following interface scenarios for computing the numerical fluxes of a leaf can be identified:

- **Leaf/Leaf or Virtual leaf:** When both leaves belong to the same refinement level, the flux computation is performed in the same way as in the FV method. The following cases can occur: Leaf/Leaf example: Flux between cells Ω_0^3 and Ω_1^3; Leaf/Virtual leaf example: Flux between cells Ω_5^4 and Ω_6^4.
- **Leaf/Internal node:** In this scenario, the current leaf has an interface with a finer leaf. To perform the flux computation in this case, the numerical flux is computed using the virtual children of the leaf and its adjacent leaves. Example: Flux between cells Ω_2^2 and Ω_1^2.

The construction of the adaptive grid guarantees that the current solution is well represented. However, after the time evolution process, the new solution should also be well represented on this grid, which a priori cannot be ensured. In order to guarantee that the new solution after the time evolution is still well represented on this grid, the leaves with finer neighbours are refined and neighbour cells are added. Further details of the MR scheme and its implementation can be found in [26]. The Algorithms 1 and 2 given in Appendix A describe the adaptive grid creation and its update, respectively.

2.1. Projection and prediction operators

In order to perform the MR method, some operations for projection and prediction are required. For the MR scheme with finite volumes, where the cell values are local averages, a coarser cell Ω_i^ℓ has its value estimated using the finer values and a unique projection operator $P_{\ell+1\to\ell} : \bar{\mathbf{q}}^{\ell+1} \mapsto \bar{\mathbf{q}}^{\ell}$. In this scheme, the projection operator to obtain the solution of a coarser cell is given by the average value of its children. For the unidimensional case, the projection is performed by:

$$\bar{\mathbf{q}}_i^\ell = P_{\ell+1\to\ell}\left(\bar{\mathbf{q}}_{2i}^{\ell+1}, \bar{\mathbf{q}}_{2i+1}^{\ell+1}\right) = \frac{1}{2}\left(\bar{\mathbf{q}}_{2i}^{\ell+1} + \bar{\mathbf{q}}_{2i+1}^{\ell+1}\right), \tag{3}$$

where $\bar{\mathbf{q}}_i^\ell$ are the average value of the cell Ω_i^ℓ. The same idea is extended for the two and three-dimensional cases.

The prediction operators are used to perform the opposite path of the projection operators, they allow to obtain the values of the finer cells using the values of the coarser ones. For each child cell to be predicted, there is a different prediction operator, represented by $P^{i}_{\ell\to\ell+1} : \bar{\mathbf{q}}^{\ell} \mapsto \bar{\mathbf{q}}^{\ell+1}$ for the one-dimensional case, $P^{i,j}_{\ell\to\ell+1} : \bar{\mathbf{q}}^{\ell} \mapsto \bar{\mathbf{q}}^{\ell+1}$ for the two-dimensional case and $P^{i,j,k}_{\ell\to\ell+1} : \bar{\mathbf{q}}^{\ell} \mapsto \bar{\mathbf{q}}^{\ell+1}$ for the three-dimensional case. These operators yield a non-unique approximation of $\bar{\mathbf{q}}_i^{\ell+1}$ by interpolation. We use polynomial interpolation of second degree on the cell-averages, as proposed by Harten [12], which yields third-order accuracy. For the one-dimensional case, it follows that,

$$\tilde{\mathbf{q}}_{2i}^{\ell+1} = P^0_{\ell\to\ell+1}\left(\bar{\mathbf{q}}_{i-1}^{\ell}, \bar{\mathbf{q}}_{i}^{\ell}, \bar{\mathbf{q}}_{i+1}^{\ell}\right) = \bar{\mathbf{q}}_i^\ell - \frac{1}{8}(\bar{\mathbf{q}}_{i+1}^{\ell} - \bar{\mathbf{q}}_{i-1}^{\ell}) \tag{4a}$$

$$\tilde{\mathbf{q}}_{2i+1}^{\ell+1} = P^1_{\ell\to\ell+1}\left(\bar{\mathbf{q}}_{i-1}^{\ell}, \bar{\mathbf{q}}_{i}^{\ell}, \bar{\mathbf{q}}_{i+1}^{\ell}\right) = \bar{\mathbf{q}}_i^\ell + \frac{1}{8}(\bar{\mathbf{q}}_{i+1}^{\ell} - \bar{\mathbf{q}}_{i-1}^{\ell}), \tag{4b}$$

where $\tilde{\mathbf{q}}_i^\ell$ is an approximation of the value $\bar{\mathbf{q}}_i^\ell$. Interpolation operators for higher dimensions can be found in [26]. The operator must satisfy the properties of locality, requiring the interpolation for a child cell to be computed from the cell-averages of its parent and its nearest uncle cells in each direction; and consistency, $P_{\ell+1\to\ell} \circ P_{\ell\to\ell+1} = \text{Identity}$.

The prediction operator is used to obtain the wavelet coefficients $\mathbf{d}_i^\ell$ of the finer cells. These coefficients are given by the difference between the cell average $\bar{\mathbf{q}}_i^\ell$ and the predicted value $\tilde{\mathbf{q}}_i^\ell$:

$$\mathbf{d}_i^\ell = \bar{\mathbf{q}}_i^\ell - \tilde{\mathbf{q}}_i^\ell. \tag{5}$$

The values $\mathbf{d}_i^\ell$ are also used for reconstructing the finer levels without interpolation errors. Their norm yields the local approximation error. Moreover, the information of the cell-average value of the two children is equivalent to the knowledge of the cell-average value of the parent and one independent detail. This can be expressed by $\left(\bar{\mathbf{q}}_{2i}^{\ell+1}, \bar{\mathbf{q}}_{2i+1}^{\ell+1}\right) \longleftrightarrow \left(\mathbf{d}_{2i}^{\ell+1}, \bar{\mathbf{q}}^\ell i\right)$. This procedure can be applied recursively from level L down to the level 0 creating thus a multiresolution transform of the cell-average values as proposed by Harten [12]. Therefore, we have

$$\bar{\mathbf{q}}^L \longmapsto (\bar{D}^L, \bar{D}^{L-1}, \ldots, \bar{D}^1, \bar{\mathbf{q}}^0), \tag{6}$$

where $\bar{D}^\ell$ is the set of wavelet coefficients at level ℓ. Accordingly, the information of the cell-average values of all the leaves is equivalent to the knowledge of the cell-average value of the root and the wavelet coefficients of all the other nodes of the tree structure. For two and three dimensions, respectively, the information of the cell-averages of four and eight children is equivalent to the knowledge of three and seven wavelet coefficients in the different directions and the node value [12,26].

3. Runge–Kutta methods

After discretising in space the initial value problem given in Eq. (1), the following system of ordinary differential equations in time is obtained:

$$\frac{d\bar{\mathbf{q}}}{dt} = f(t, \bar{\mathbf{q}}), \tag{7}$$

where $\mathbf{Q}(t=0) = \bar{\mathbf{q}}^0$ is the given initial condition. This system yields an equation for each leaf of the grid. By abuse of or to simplify notation the space discretised solution will be denoted again by $\bar{\mathbf{q}}$. The general formulation for an explicit s-stage Runge–Kutta (RK) method can be expressed at time t^{n+1} as:

$$\bar{\mathbf{q}}^{n+1} = \bar{\mathbf{q}}^n + \sum_{i=1}^{s} b_i k_i, \tag{8}$$

with

$$k_i = \Delta t^n f\left(t^n + c_i \Delta t^n, \bar{\mathbf{q}}^n + \sum_{j=1}^{i-1} a_{ij} k_j\right), \tag{9}$$

where a_{ij}, b_i and c_i are the Runge–Kutta coefficients, and Δt^n is the time-step used to perform the time evolution from the instant t^n to t^{n+1}. The actual convergence order of the RK method depends of the number of stages and a set of well selected RK coefficients.

In this work we consider second order RK methods (RK2) with coefficients c_i and a_{ij} in such a way that they are the same coefficients used in the first and second steps of the RK3 method. Namely, the values of these coefficients are $c_1 = 0$ and $c_2 = a_{21} = 1$. The other coefficients for RK2 are $b_1 = b_2 = \frac{1}{2}$. To perform RK3, further coefficients are $c_3 = \frac{1}{2}$, $a_{31} = a_{32} = \frac{1}{4}$, $b_1 = b_2 = \frac{1}{6}$ and $b_3 = \frac{2}{3}$.

3.1. Natural continuous Extension Runge–Kutta (NERK) method

The NERK method, originally introduced by [30], produces an approximation of the solution in the time interval $[t^n; t^n + \Delta t^n]$ using the same coefficients a_{ij} and c_i as in the standard Runge–Kutta methods. The difference between NERK and RK is the use of polynomials β_i instead of the constant coefficients b_i.

The NERK method can be expressed as,

$$\bar{\mathbf{q}}(t^n + \theta \Delta t^n) = \bar{\mathbf{q}}^n + \sum_{i=1}^{s} \beta_i(\theta)\, k_i, \ \theta \in (0, 1], \tag{10}$$

where the polynomials β_i are given as a function of the coefficients b_i of the original RK method.

Using the proposed RK2 coefficients, the following polynomials for the two-stage NERK method is obtained using the methodology given in [23]:

$$\beta_1(\theta) = -\frac{1}{2}\theta^2 + \theta, \quad \beta_2(\theta) = \frac{1}{2}\theta^2. \tag{11}$$

In this work, the application of the NERK method consists in producing approximations of a solution at some intermediate time instants inside the interval $[t^n, t^n + \Delta t^n]$, depending on the choice for RK2 or RK3 time evolution.

3.2. Compact formulation

In order to reduce the memory allocation per cell when performing the time evolution, the compact formulation of the RK methods is of particular interest, used for example in [26]. Based on the standard RK methods, we can obtain the following compact formulation for the two and three stage methods,

- RK2: $\bar{\mathbf{q}}^* = \bar{\mathbf{q}}^n + \Delta t^n f(t^n, \bar{\mathbf{q}}^n), \qquad \bar{\mathbf{q}}^{n+1} = \frac{1}{2}\bar{\mathbf{q}}^n + \frac{1}{2}\bar{\mathbf{q}}^* + \frac{1}{2}\Delta t^n f\left(t^n + \Delta t^n, \bar{\mathbf{q}}^*\right).$
- RK3: $\bar{\mathbf{q}}^* = \bar{\mathbf{q}}^n + \Delta t^n f(t^n, \bar{\mathbf{q}}^n), \qquad \bar{\mathbf{q}}^{**} = \frac{3}{4}\bar{\mathbf{q}}^n + \frac{1}{4}\bar{\mathbf{q}}^* + \frac{1}{4}\Delta t^n f\left(t^n + \Delta t^n, \bar{\mathbf{q}}^*\right),$
  $\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\bar{\mathbf{q}}^n + \frac{2}{3}\bar{\mathbf{q}}^{**} + \frac{2}{3}\Delta t^n f\left(t^n + \frac{1}{2}\Delta t^n, \bar{\mathbf{q}}^{**}\right).$

When performing the time integration using a local time-stepping approach, *cf.* Section 4, some intermediate values are necessary. In this work, we propose to use the NERK method for that. However, the values k_i are not stored in the RK compact formulation. Hence the NERK solution must be adapted to be compatible with the compact RK method.

The following steps produce a NERK/RK2 approximation at the time instant $t^n + \frac{1}{2}\Delta t^n$. Due to the memory management of the numerical code, where the fluxes are stored at the same memory allocation, each one of the following steps is executed immediately after its corresponding compact RK step:

$$\bar{\mathbf{q}}^*_{\theta=\frac{1}{2}} = \bar{\mathbf{q}}^n + \frac{3}{8}\Delta t^n f(t^n, \bar{\mathbf{q}}^n), \qquad \bar{\mathbf{q}}_{\theta=\frac{1}{2}} = \bar{\mathbf{q}}^*_{\theta=\frac{1}{2}} + \frac{1}{8}\Delta t^n f\left(t^n + \Delta t^n, \bar{\mathbf{q}}^*\right), \tag{12}$$

where the values $f(t^n, \bar{\mathbf{q}}^n)$ and $f\left(t^n + \Delta t^n, \bar{\mathbf{q}}^*\right)$ are the same as those obtained for the compact RK formulation. In this formulation, the values $\bar{\mathbf{q}}_{\theta=\frac{1}{2}}$ and $\bar{\mathbf{q}}^*_{\theta=\frac{1}{2}}$ are stored at the same memory allocation.

For the RK3 time evolution, second order approximations at the time instants $t^n + \frac{1}{4}\Delta t^n$ and $t^n + \frac{3}{4}\Delta t^n$ become necessary. These approximations are required to compute the third step of the RK3. However, the NERK/RK3 method only yields this information after the third step. The proposed solution for this problem is to obtain these approximations via NERK/RK2. Then, it is possible to obtain approximations at these desired instants immediately after the second step of the RK method, under the condition that the coefficients a_{ij} and c_i of both, RK2 and RK3, methods are the same. Therefore we use the following approximations:

- at $t^n + \frac{1}{4}\Delta t^n$: $\bar{\mathbf{q}}^*_{\theta=\frac{1}{4}} = \bar{\mathbf{q}}^n + \frac{7}{32}\Delta t^n f(t^n, \bar{\mathbf{q}}^n), \qquad \bar{\mathbf{q}}_{\theta=\frac{1}{4}} = \bar{\mathbf{q}}^*_{\theta=\frac{1}{4}} + \frac{1}{32}\Delta t^n f\left(t^n + \Delta t^n, \bar{\mathbf{q}}^*\right),$
- at $t^n + \frac{3}{4}\Delta t^n$: $\bar{\mathbf{q}}^*_{\theta=\frac{3}{4}} = \bar{\mathbf{q}}^n + \frac{15}{32}\Delta t^n f(t^n, \bar{\mathbf{q}}^n), \qquad \bar{\mathbf{q}}_{\theta=\frac{3}{4}} = \bar{\mathbf{q}}^*_{\theta=\frac{3}{4}} + \frac{9}{32}\Delta t^n f\left(t^n + \Delta t^n, \bar{\mathbf{q}}^*\right).$

These approximations can be performed using similar memory management ideas as for the RK2 evolution.

4. Local time-stepping

To improve further the computational efficiency of the MR method, a local time-stepping approach was proposed in [6]. This approach consists in using an adapted time-step for each leaf individually. This time step is obtained accordingly to the spatial size of the leaf. Thus, small time steps are only used for fine scale leaves, while large time-steps can be used for coarser leaves. This is possible without violating the stability condition of the explicit time discretisation, as shown in [6].

Solutions with point-like singularities are well adapted and yield highly efficient multiresolution representations. For those, the MRLT approach is found to be most efficient. Besides, the adaptive grid created for the MRLT scheme is the same graded tree structure used in the MR scheme [26]. The update procedure for the trees during MRLT schemes is discussed in Section 4.3.1.

We suppose that the CFL condition implies a time-step Δt for the most refined level. In the LT scheme, each cell of level ℓ performs its time evolution with a proper time step given by:

$$\Delta t_\ell = 2^{L-\ell}\Delta t, \tag{13}$$

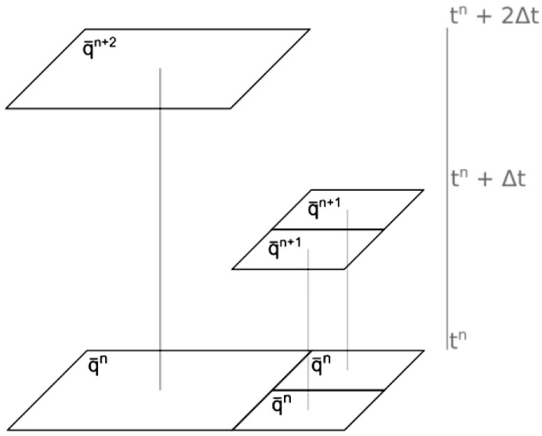


Fig. 2. LT evolution of adjacent cells at different scales. The finer cells in this representation need a second time evolution to reach the instant t^{n+2}.

where L is the finest level of the grid. From this point, the notation is adjusted in order to facilitate the understanding. Moreover, considering the Courant number σ depending on the ratio between Δt_ℓ and Δx_ℓ, and using the relations between the cell size and the time-step of different levels, the value σ obtained for every cell will thus be the same.

As illustrated in Fig. 2, due to the scale dependent time-stepping of the LT scheme, not all leaves of the coarser scales will be evolved during an iteration of the time evolution. We define the coarsest level where the leaves must be evolved in a certain iteration n to be the minimum scale level in which the modulo operator between n and $2^{L-\ell}$ is zero and we denote it as $\ell_{\min}$.

The above approach is restricted to second order time accuracy, because the internal steps of standard Runge–Kutta schemes are not compatible with the dyadic grid size. The reason is that the intermediate time steps of higher order RK schemes (order larger than two) do not correspond to the time instants of the solution obtained by the RK method when using the dyadic grid size [6]. A possible solution to overcome this limitation is to use NERK schemes. Their polynomial approximation in time is used to evaluate the solution at the intermediate time instants imposed by the dyadic spatial grid size.

The implementation of high order LT schemes leads to three synchronization challenges during the time evolution of a leaf with a neighbour at a different refinement level. Those challenges are discussed in the following subsections. The whole LT method, as proposed in this work, is performed as given in Algorithm 3.

4.1. Synchronization during the Runge–Kutta iteration

Considering the restriction that during an iteration of LT schemes only leaves of refinement level greater or equal $\ell_{\min}$ are evolved, we know that all of these leaves have their solutions at the same time instant t^n, requiring no synchronization to perform the first RK step. Thus, the first RK step can be done with the MR method using the proper Δt_ℓ value. After performing the first step of the Runge–Kutta method, the leaves at each level are evolved with their own time step, obtaining a first order solution at time instant $t^n + \Delta t_\ell$. Using the fluxes obtained in this step, the values $\bar{\mathbf{q}}^*_{\theta=\frac{1}{2}}$, for RK2, or $\bar{\mathbf{q}}^*_{\theta=\frac{1}{4}}$ and $\bar{\mathbf{q}}^*_{\theta=\frac{3}{4}}$ for RK3, are computed at the leaves of every level evolved in this iteration.

However, for the next RK steps, due to the different time step size, the solution values after the RK step are given in a different time instant for each refinement level. This implies that some synchronization has to be performed.

4.1.1. Second Runge–Kutta step synchronization

In order to perform the second step of the RK method, it is necessary to compute the flux $f(t^n + \Delta t_\ell, \bar{\mathbf{q}}^*)$. This flux can be interpreted as the flux between leaves at the time instant $t^n + \Delta t_\ell$, where the leaf value is a first order approximation $\bar{\mathbf{q}}^* = \bar{\mathbf{q}}^n + a_{21}k_1$. The challenge here lies in the fact that when this approximation is obtained at a finer scale, due to the different time step size used, it is located at an earlier time instant with respect to the approximation at a coarser scale. This situation implies that the flux computation for each scale has to be performed at a different time instant, as shown in the scheme presented in Fig. 3. The values required for both cases of the second RK step are not available after the first RK step, requiring thus a synchronization procedure to obtain those values for each scale. Then the second RK step in this scale can be performed. To obtain the missing values to compute the fluxes in the situation given in Fig. 3(a), a tree refreshing procedure, described in Section 4.2, must be performed in order to project the solution for scales $\ell > \ell_{\min}$ at time instant $t^n + \Delta t_\ell$ onto the scale $\ell - 1$ at instant $t^n + \Delta t_{\ell-1}$.

The proposed synchronization methodology to perform the second RK step consists in using a first order approximation $\bar{\mathbf{q}}^n_\ell + \frac{1}{2}a_{21}k_1$ as a solution at the time instant $t^n + \frac{1}{2}\Delta t_\ell$ for every $\ell \neq L$. This approximation is in the same time instant as the solution in the next finer level. The use of this approximation consists in predicting the values of the virtual leaves at level $\ell + 1$, at the proper time instant, before performing the flux computations of the leaves at level $\ell + 1$. This approach is illustrated for the situation presented in Fig. 3(a).

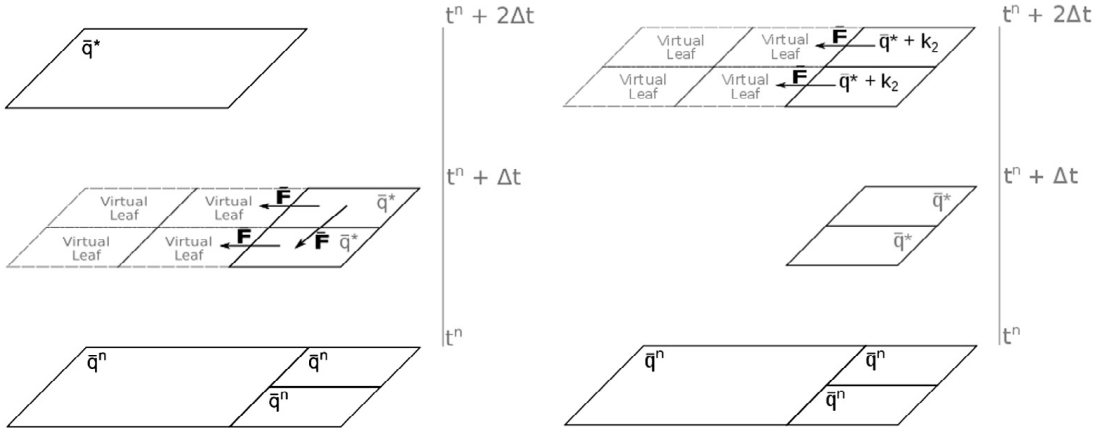


Fig. 3. Schematic view of the synchronization issues for the second RK step. a) Presents the evaluation of flux computations $\bar{\mathbf{F}}$ at instant $t^n + \Delta t$. For adjacent leaves at the same level we can compute the fluxes directly. However, to compute the fluxes between different levels, we need the virtual leaves of the coarser leaf at the same time instant of the finer leaves. For that, first, we interpolate the coarser leaf $\bar{\mathbf{q}}$ at instance $t^n + \Delta t$ from its respective values at t^n and $t^n + 2\Delta t$. Then, we predict its value at the finer level and obtain its virtual leaves. After that, we evaluate the flux, and then, the RK step for the finer leaves. b) Presents the evolution of the flux at $t^n + 2\Delta t$ of the coarser leaf. First, we predict the virtual leaves of the coarse leaf, then we compute $\bar{\mathbf{q}}^* + k_2$, *i.e.* a RK1 step of the finer leaves, which yields a first order approximation. After that, we can compute the flux and evaluate the leaves.

In order to simplify the algorithm, the prediction of the virtual leaves at the coarser scales at the proper time instant, necessary to update the next finer level, is performed during the tree refreshing process, explained in detail for every RK step, in Section 4.2. In this Section, we focus on the update of the virtual leaves at level $\ell + 1$ in order to perform the flux computations on leaves of level ℓ, as presented in Fig. 3(b).

Initially, the flux computations for the second RK step are performed on the leaves of level L. This choice is due to the fact that at this level, there are no interfaces of the type leaf/internal node, avoiding thus the situation shown in Fig. 3(b).

Having updated the level L, the same process is repeated recursively for the levels $L-1$ down to $L = \ell_{\min}$. However, due to the leaf/internal node scenario, the leaves and virtual leaves of the previously updated level $\ell + 1$ must be synchronized at the instant $t^n + \Delta t_\ell$, which is equivalent to the instant $t^n + 2\Delta t_{\ell+1}$, in order to perform the flux computations of the next coarser level. For that we propose the use of an extrapolated value, resulting in the situation presented in Fig. 3(b).

In this work, we suggest to obtain an extrapolation at time instant $t^n + 2\Delta t_{\ell+1}$, which allows to compute the fluxes for the second step of the RK method. To this end, the value k_2 is used as the value k_1 in a second RK1 time evolution. The first order approximation of the solution at the instant $t^n + 2\Delta t_{\ell+1}$ is computed as:

$$\bar{\mathbf{q}}_{\ell+1}\left(t^n + 2\Delta t_{\ell+1}\right) = \bar{\mathbf{q}}^*_{\ell+1} + \Delta t_{\ell+1} f\left(t + \Delta t_{\ell+1};\ \bar{\mathbf{q}}^*_{\ell+1}\right) = \bar{\mathbf{q}}^n_{\ell+1} + k_1 + k_2 \tag{14}$$

where $\bar{\mathbf{q}}^*_{\ell+1}$ is the first RK step, and $\Delta t_{\ell+1} f\left(t + \Delta t_{\ell+1};\ \bar{\mathbf{q}}^*_{\ell+1}\right)$ is the flux of the second RK step. Note that this approximation is only possible due to the choice of the coefficients $a_{11} = b_1 = 1$.

Once the leaves of level $\ell + 1$ are extrapolated to the instant $t^n + 2\Delta t_{\ell+1}$, the virtual leaves of this level are also updated. However, in order to predict the value of the virtual leaves at level $\ell + 1$, the values of the virtual leaves and internal nodes of level ℓ must be synchronized at instant $t^n + \Delta t_\ell$ first.

The solution of the virtual leaves and internal nodes of level ℓ in this time instant is obtained during the tree refreshing process. However, in order to improve the solution of the internal nodes, their values at time instant $t^n + \Delta t_\ell$ are updated after the evolution of the leaves of level $\ell + 1$. This update is performed via projection of the extrapolated solution at instant $t^n + 2\Delta t_{\ell+1}$ from the level $\ell + 1$ to ℓ. In contrast to the leaves, which have an approximation at this instant, the internal nodes do not. The values for the internal nodes are obtained via linear extrapolation:

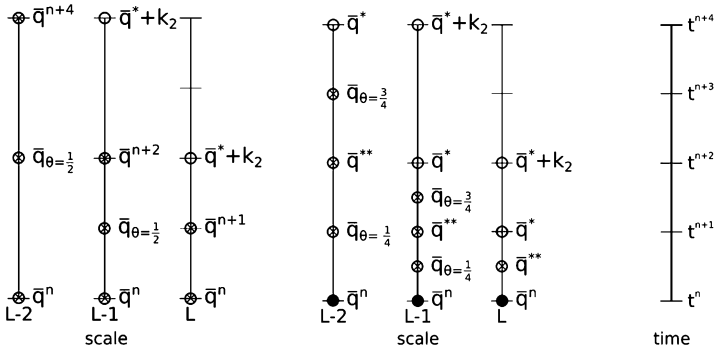
$$\bar{\mathbf{q}}_{\ell+1}\left(t^n + 2\Delta t_{\ell+1}\right) = 2\bar{\mathbf{q}}^*_{\ell+1} - \bar{\mathbf{q}}^n_{\ell+1}. \tag{15}$$

This approximation is used to perform the projection of the solution of the internal node to the next finer level ℓ. The projection procedure is given in Algorithm 4. After the projection procedure, the virtual leaves of level $\ell + 1$ have their predicted values. Then, the flux computations and the update of the leaves at level ℓ can be performed. The second step of the RK method is given in Algorithm 5.

During the second RK step, the fluxes are also used for computing the second step of the NERK approximation with second order at the time instants $t^n + \frac{1}{2}\Delta t_\ell$ for RK2 or $t^n + \frac{1}{4}\Delta t_\ell$ and $t^n + \frac{3}{4}\Delta t_\ell$ for RK3 time evolution.

As stated before, the NERK approximation for the RK2 time evolution is used to solve the synchronization according the time evolution problem given in Section 4.3. For the RK3 time evolution, the NERK approximations are used to perform the third RK step.

After the second step of the RK evolution, a solution at the time instant $t^n + \Delta t_\ell$ for the RK2 method, and at $t^n + \frac{1}{2}\Delta t_\ell$ for the RK3 method is obtained. In order to prepare the tree structure for the next RK2 iteration and for the next RK3



a) RK2 evolution. b) RK3 evolution.

Fig. 4. Scheme of the obtained values and its respective time instants for cells in each level after performing the second RK step in the LT method: a) for RK2 and b) for the RK3 time evolution. The filled, crossed and clear circles represent an available solution, of first, second and third order, respectively, at the corresponding time instant.

step, another tree refreshing procedure must be applied. This procedure is given in detail in Section 4.2. After this process, the information obtained for the cells (internal nodes, leaves and virtual leaves) in every refinement level are illustrated in Fig. 4.

4.1.2. Third Runge–Kutta step synchronization

The third step is done only in the case of the RK3 time evolution. As for the second step of the RK evolution, in the third step, the evolution must be done first on the finest scale in order to obtain the necessary values to perform the evolution on the next coarser scale. However there are, indeed, slightly different synchronization issues in this case which we discuss in the following.

As performed in the second RK step, the virtual leaves and the internal nodes have their values updated during the tree refreshing process, the details are discussed in the next section.

To perform the third RK step, the fluxes of level L are computed in the proper time instant $t^n + \frac{1}{2}\Delta t_L$. Then, the third RK step yields a third order solution at the time instant $t^n + \Delta t_L$.

Again as for the second step, this solution is projected onto the next coarser level in order to update the virtual leaves at level L at the instant $t^n + \Delta t_L$, where the fluxes for the third step of the level $L-1$ shall be computed.

The projection procedure for the third RK step is, as mentioned above, slightly different from the projection procedure in the second step. In this case, the solution at the leaves at level ℓ after the third RK step matches with the time instant where the fluxes of level $\ell - 1$ are computed. Then, the solution is projected from level ℓ onto level $\ell - 1$ by simple averaging, obtaining thus a solution at $t^n + \frac{1}{2}\Delta t_{\ell-1}$. This solution is used as an approximation for $\bar{\mathbf{q}}^{**}$. Then, the value of the internal nodes at the instant $t^n + \Delta t_{\ell-1}$ should be updated with a second order solution in order to continue the projections recursively during the third RK step. This update is performed using the following relation obtained from the NERK scheme:

$$\bar{\mathbf{q}}_\ell(t^n + \Delta t_\ell) = \bar{\mathbf{q}}_\ell^n + 2\left(\bar{\mathbf{q}}_{\ell,\theta=\frac{3}{4}} - \bar{\mathbf{q}}_{\ell,\theta=\frac{1}{4}}\right). \tag{16}$$

This projection procedure inside the third RK step is given in Algorithm 6, and the execution of the third step of the RK evolution in Algorithm 7.

4.2. Tree refreshing with time synchronization

In the MR scheme, due to the same time step for every scale, the leaves of every scale store values corresponding to the same time instant. Therefore, projections of the leaves onto internal nodes receive values at the same time instant. However, in the LT approach, due to fact that leaves of different scales are evolved with different time steps, the tree refreshing may project values at different time instants related to the leaves at the same scale. Accordingly, we need a procedure before each RK step internally to the time cycle, to project the solution to an internal node at its corresponding time instant.

Moreover, these synchronizations are necessary to predict the values of the virtual leaves for the next steps of the RK method, at the required time instant. Only after that, the flux computations can be performed.

Before the first RK step, every scale of level $\ell_{\min}$ or greater, which must be evolved, has its solution at the same time instant. Therefore, the values of the virtual leaves can be predicted normally, except at level $\ell_{\min}$, which has its virtual leaves predicted using the solution at $t^n + \frac{1}{2}\Delta t_{\ell_{\min}-1}$ from the level $\ell_{\min} - 1$.

After the first step of the RK method, each leaf yields a first order solution at the time instant $t^n + \Delta t_\ell$, which corresponds to the time instant $t^n + \frac{1}{2}\Delta t_{\ell-1}$. This implies that the projection of the leaves from level ℓ to $\ell - 1$ produces an

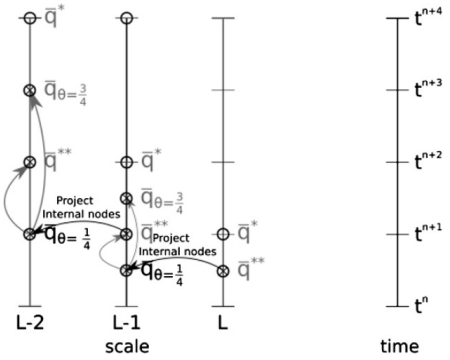


Fig. 5. Projection scheme before the third RK step.

approximation at that time instant. The value of an internal node at this instant is given by the mean value of its children:

$$\bar{\mathbf{q}}_{\ell-1}\left(t^n+\frac{1}{2}\Delta t_{\ell-1}\right)=\bar{\mathbf{q}}^n_{\ell-1}+\frac{1}{2}\Delta t_{\ell-1}f\left(t^n,\bar{\mathbf{q}}^n_{\ell-1}\right)=\frac{1}{2^d}\sum_{i=1}^{2^d}\left[\bar{\mathbf{q}}^n_{\ell,\,i}+\Delta t_\ell f\left(t^n,\bar{\mathbf{q}}^n_{\ell,\,i}\right)\right] \tag{17}$$

where d is the number of dimensions of the problem and $\bar{\mathbf{q}}^n_{\ell,\,i}$ is the solution of the children cell i. This value is stored in order to predict the values of the virtual leaves of level ℓ for the second RK step. Furthermore, this value must be extrapolated to the instant $t^n+\Delta t_{\ell-1}$, obtaining the value $\bar{\mathbf{q}}^*_{\ell-1}$. This value, used to update the virtual leaves of level ℓ during the second RK step, is obtained by linear extrapolation:

$$\bar{\mathbf{q}}^*_{\ell-1}=2\bar{\mathbf{q}}_{\ell-1}\left(t^n+\frac{1}{2}\Delta t_{\ell-1}\right)-\bar{\mathbf{q}}^n_{\ell-1}, \tag{18}$$

where the value $\bar{\mathbf{q}}^n_{\ell-1}$ is the solution before the time evolution. This value should be stored for every node before the RK procedure. The tree refreshing procedure after the first RK step is given in Algorithm 8.

When performing the RK3 time evolution, before the third RK step, besides the solution, the values $\bar{\mathbf{q}}_{\theta=\frac{1}{4}}$ and $\bar{\mathbf{q}}_{\theta=\frac{3}{4}}$ must be obtained for the internal nodes. The solutions in those instants are required to predict the solution of the virtual leaves in the next finer level at the instant required for this RK step.

Those values are obtained through projections from the level L down to the level $\ell_{\min}$, initially by projecting the solution $\bar{\mathbf{q}}^{**}_\ell$ at the time instant $t^n+\frac{1}{2}\Delta t_\ell$, obtaining thus a value at $t^n+\frac{1}{4}\Delta t_{\ell-1}$, which is used as an approximation for $\bar{\mathbf{q}}_{\ell-1,\,\theta=\frac{1}{4}}$.

Using this solution, the Equation (10) and the value $\bar{\mathbf{q}}^*_\ell$, computed during the projection before the second RK step, we obtain the following relation to reconstruct the solution inside the interval $[t^n;t^n+\Delta t^n]$:

$$\bar{\mathbf{q}}_\ell\left(t^n+\theta\Delta t_\ell\right)=\left[1-\theta-12\theta^2\right]\bar{\mathbf{q}}^n_\ell+\left[\theta-4\theta^2\right]\bar{\mathbf{q}}^*_\ell+16\theta^2\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}}. \tag{19}$$

This expression is based on the NERK scheme presented in Section 3.1. Using the value $\theta=\frac{3}{4}$, we have:

$$\bar{\mathbf{q}}_{\ell,\,\theta=\frac{3}{4}}=-\frac{13}{2}\bar{\mathbf{q}}^n_\ell-\frac{3}{2}\bar{\mathbf{q}}^*_\ell+9\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}} \tag{20}$$

Subsequently, using these values, the value $\bar{\mathbf{q}}^{**}$ can be computed as:

$$\bar{\mathbf{q}}^{**}_\ell=-\frac{11}{2}\bar{\mathbf{q}}^n_\ell-\frac{3}{2}\bar{\mathbf{q}}^*_\ell+8\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}} \tag{21}$$

The solution $\bar{\mathbf{q}}^{**}_\ell$ allows to continue the projection procedure recursively by obtaining the value $\bar{\mathbf{q}}_{\ell-1,\,\theta=\frac{1}{4}}$ at the next coarser level, while the solution $\bar{\mathbf{q}}_{\ell,\,\theta=\frac{3}{4}}$ is used to compute the value of the virtual leaves at level $\ell+1$ on the following iterations, as explained in Section 4.1.

The tree refreshing procedure, illustrated in Fig. 5, consists in projecting the solution $\bar{\mathbf{q}}^{**}_L$ from the level L onto the level $L-1$. This solution is approximated as $\bar{\mathbf{q}}_{L-1,\,\theta=\frac{1}{4}}$. Then, using the previously obtained value $\bar{\mathbf{q}}^*_{L-1}$, the values $\bar{\mathbf{q}}_{L-1,\,\theta=\frac{3}{4}}$ and $\bar{\mathbf{q}}^{**}_{L-1}$ can be obtained using Equations (20) and (21). This procedure is recursively repeated down to the coarsest scale $\ell_{\min}$.

In the third RK step, the fluxes are computed at the intermediary time instant $t^n+\frac{1}{2}\Delta t_\ell$. For an adjacent coarser leaf, this instant is $t^n+\frac{1}{4}\Delta t_{\ell-1}$ or $t^n+\frac{3}{4}\Delta t_{\ell-1}$, depending on the current iteration number. Basically, if the current scale is $\ell_{\min}$ then the NERK value used to compute the virtual leaves is available at time instant $t^n+\frac{3}{4}\Delta t_{\ell_{\min}-1}$. Otherwise, the value at the time instant $t^n+\frac{1}{4}\Delta t_{\ell-1}$ is used.

The choice which value of the NERK approximation shall be used in the flux computations is shown in Fig. 6, where we choose $\ell_{\min}=L-2$.

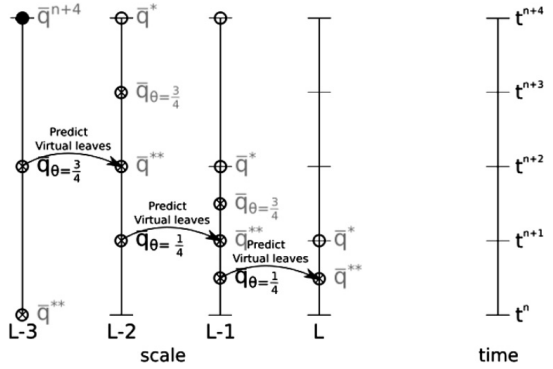


Fig. 6. Scheme of the choice of the NERK solution to predict the virtual leaves for the third RK step in a LT iteration with $\ell_{\min} = L - 2$. The solution $\bar{\mathbf{q}}_{\theta=\frac{3}{4}}$ is used to predict the $\bar{\mathbf{q}}^{**}$ values at level $\ell_{\min}$. For the other scales, the solution $\bar{\mathbf{q}}^{**}$ is predicted with the values $\bar{\mathbf{q}}_{\theta=\frac{1}{4}}$.

Once the projections have been performed from the level L until the level $\ell_{\min}$, the virtual leaves have their values $\bar{\mathbf{q}}^{**}_{\ell}$ predicted using the values $\bar{\mathbf{q}}_{\ell-1,\,\theta=\frac{3}{4}}$, in case of $\ell = \ell_{\min}$; or using $\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}}$, otherwise.

Then, the values $\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}}$ and $\bar{\mathbf{q}}_{\ell,\,\theta=\frac{3}{4}}$ for these virtual leaves are obtained using the Equations (20) and (21). Those values are required here to predict the values of the virtual leaves at the next finer scale.

The tree refreshing procedure before the third RK step is detailed in Algorithm 9.

4.3. Synchronization after the time evolution

After finishing the RK steps, the leaves at level ℓ perform a complete time evolution with a time step that is twice the time step of the leaves at level $\ell + 1$. In order to perform the next time evolution at scale ℓ, the finer scale leaves must reach the same time instant. For that, a second time evolution is required for this finer scale leaf.

Moreover, in the next iteration of the time evolution, a new value for $\ell_{\min}$ is obtained. To predict the virtual leaves at the new scale $\ell_{\min}$ at the proper time instant, the values $\bar{\mathbf{q}}_{\ell_{\min}-1,\,\theta=\frac{1}{2}}$ for RK2 or $\bar{\mathbf{q}}^{**}_{\ell_{\min}-1}$ for RK3 are used. For the scales finer than $\ell_{\min}$, the virtual leaves are predicted with the MR method.

To solve this synchronization challenge and to maintain the approximation order, a high order approximation at time step $t^n + \frac{1}{2}\Delta t_\ell$, where ℓ is the refinement level of the leaf, is required. In this case, in the tree terminology, it means that for the coarser leaves, an approximation of its virtual children at the instant $t^n + \frac{1}{2}\Delta t_\ell$ is needed.

To obtain these values, it is necessary to obtain approximations at the instant $t^n + \frac{1}{2}\Delta t_\ell$ during the Runge–Kutta evolution for every cell of refinement level $\ell \neq L$ and its neighbours, including internal nodes. Then, the values of the virtual children are obtained by the prediction procedure.

For the RK2 time evolution, we use the NERK approach to obtain the approximation with $\theta = \frac{1}{2}$. However, the implementation of the MRLT/NERK approach for the RK3 time evolution requires extra memory. Hence we use the approximation $\bar{\mathbf{q}}^{**}$, obtained after the second step of the compact RK, which is at the instant $t^n + \frac{1}{2}\Delta t_\ell$, instead of the NERK approximation with $\theta = \frac{1}{2}$ in order to reduce the number of variables to be stored in the problem.

Besides those values, the updated solution, at time instant $t^n + \Delta t_\ell$, is projected onto the coarser level at the time instant $t^n + \frac{1}{2}\Delta t_{\ell-1}$ by simple averaging. This solution is then used as an approximation for $\bar{\mathbf{q}}_{\ell-1,\,\theta=\frac{1}{2}}$, in the RK2 time evolution, or $\bar{\mathbf{q}}^{**}_{\ell-1}$ for the RK3 time evolution. Using the NERK equations given in Section 3.2 and the current value of the internal node at $t^n + \Delta t_{\ell-1}$ (RK first step), the solution at instant $t^n + \Delta t_\ell$ is obtained by the following relation:

$$\bar{\mathbf{q}}_\ell\left(t^n + \Delta t_\ell\right) = -2\bar{\mathbf{q}}^n_\ell - \bar{\mathbf{q}}^*_\ell + 4\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{2}}. \tag{22}$$

This value is used to compute the value of the virtual leaves at level $\ell_{\min} + 1$ in the following iterations at the second RK step, as explained in Section 4.1.1.

4.3.1. Adapting the grid during local time-stepping

During the LT scheme, in most parts of the numerical simulation, some scales have results at different time instants. Therefore, in order to avoid errors to be caused by converting a cell with a solution value at a determined time instant to another scale where its cells are in a different time instant, there is a need for a criterion when adapting the solution during the LT scheme.

This criterion consists in combining or splitting only leaves from a scale to another scale whose leaves are at the same time instant. These scales can be easily detected due to the fact that the time evolution procedure is applied only in scales at the same time instant. In other words, to split a cell into more refined ones, both scales must be included into the current time evolution iteration. The same is valid to combine cells into a coarser one.

Moreover, beyond this restriction, the remeshing process is identical to the remeshing process of the multiresolution method.

4.4. Some remarks on the convergence and stability of MRLT/NERK schemes

In this section, we perform a stability analysis to check the convergence order of the MRLT/NERK3 method. For that, we compute the interpolation and extrapolation errors in each approximation required for the method, considering the interface between cell of different levels.

To analyze the stability typically a simple linear ODE,

$$\frac{dq}{dt} = \lambda u \tag{23}$$

where a constant complex-valued coefficient $\lambda \in \mathbb{C}$ with negative or zero real part, is considered. The above equation is completed with an initial condition. Using Fourier analysis of linear PDEs it can be shown that imaginary values of λ correspond to pure advection, while real (negative) values correspond to pure diffusion equations.

One-step schemes, including Runge–Kutta methods can then be written in the following form,

$$\hat{\mathbf{q}}^{n+1} = g(\lambda\Delta t)\,\hat{\mathbf{q}}^{n} \tag{24}$$

where g is a polynomial. In particular, for RK1, RK2 and RK3 we respectively have $g(\lambda\Delta t) = 1 + \lambda\Delta t$, $g(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}\lambda^2\Delta t^2$ and $g(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}\lambda^2\Delta t^2 + \frac{1}{6}\lambda^3\Delta t^3$, which correspond to the truncated Taylor series of $\exp(\lambda\Delta t)$. A method is called of convergence order n if its polynomial $g(\lambda\Delta t)$ reconstructs the Taylor series up to the $\lambda^n\Delta t^n$ term.

To compute the polynomial g for the MRLT/NERK methods, the analysis is performed by computing and inserting each error ϵ_i obtained in every approximation required to perform the MRLT method.

The errors obtained in those approximations are inserted into the flux computations of the stability model as follows:

$$f(\bar{\mathbf{q}}) = f(\hat{\mathbf{q}} - \epsilon)\,, \tag{25}$$

where $\bar{\mathbf{q}}$ is an approximation obtained during the MRLT/NERK method and $\hat{\mathbf{q}}$ is the solution that would be obtained by the regular RK method.

In the following, we obtain the errors ϵ for each approximation used for the first RK evolution on both finer and coarser scales. In this evolution, both scales are initially at the same time instant, so there is no approximation error for the first RK step (k_1). In other words, it means that $\bar{\mathbf{q}}^n = \hat{\mathbf{q}}^n$.

In this section, we consider the RK3 scheme as given in Section 3.2.

Considering that the solution $\bar{\mathbf{q}}^n$ is at the same instant for every scale to be updated in the time evolution, the first RK stage reads,

$$\bar{\mathbf{q}}^{*} = \bar{\mathbf{q}}^{n} + \Delta t^n f(t^n, \bar{\mathbf{q}}^n) \tag{26}$$

and yields no interpolation errors due to the LT approach for both scales. That leads to $\bar{\mathbf{q}}^* = \hat{\mathbf{q}}^*$.

4.4.1. Approximations after the first RK step

Here we compute the errors obtained from the predictions and projections after the first RK step, as performed in Section 4.1.1. The solution of the intermediary solution $\bar{\mathbf{q}}^*$ is at a different time instant for each refinement level, requiring thus a series of interpolations and extrapolations, which introduces errors ϵ_1 and ϵ_2 into the scheme.

- ϵ_1: Prediction error in the finer leaf.

Error obtained in the prediction of $\bar{\mathbf{q}}^*$ in the finer leaf using the values of the solution in the lower level with half of the time-step.

$$\epsilon_1 = \hat{\mathbf{q}}^{*}_{\ell} - P_{\ell-1\to\ell}\left[\bar{\mathbf{q}}^{n}_{\ell-1} - \frac{\Delta t_{\ell-1}}{2} f(\bar{\mathbf{q}}^{n}{}_{\ell-1})\right] = 0. \tag{27}$$

Proof. Considering $f(\bar{\mathbf{q}}^{n}{}_{\ell-1}) = \lambda\bar{\mathbf{q}}^{n}{}_{\ell-1}$

$$\epsilon_1 = \hat{\mathbf{q}}^{n}_{\ell} + \Delta t_\ell \lambda \hat{\mathbf{q}}^{n}_{\ell} - P_{\ell-1\to\ell}\left[\bar{\mathbf{q}}_{\ell-1} - \frac{\Delta t_{\ell-1}}{2}\lambda\bar{\mathbf{q}}_{\ell-1}\right] \tag{28}$$

$$\epsilon_1 = \hat{\mathbf{q}}^{n}_{\ell} + \Delta t_\ell \lambda \hat{\mathbf{q}}^{n}_{\ell} - \bar{\mathbf{q}}_{\ell} - \frac{\Delta t_{\ell-1}}{2}\lambda\bar{\mathbf{q}}_{\ell} \tag{29}$$

$$\epsilon_1 = 0 \tag{30}$$

- ϵ_2: Projection error in the coarser leaf.
  Error obtained in the projection to obtain $\bar{\mathbf{q}}^*$ in the coarser level. This solution is a first order extrapolation to the time instant $t^n + \Delta t_{\ell-1}$ for the leaves in the finer level.

$$\epsilon_2 = \hat{\mathbf{q}}^*_\ell - P_{\ell+1\to\ell}\left[\underbrace{\bar{\mathbf{q}}^n_{\ell+1} + \Delta t_{\ell+1} f(\bar{\mathbf{q}}^n_{\ell+1})}_{\text{First RK step on finer leaf}} + \underbrace{\Delta t_{\ell+1} f(\bar{\mathbf{q}}^*_{\ell+1})}_{\text{extrapolation to } t^n+\Delta t_{\ell-1}}\right] = -\frac{\Delta t_\ell^2 \lambda^2}{4}\hat{\mathbf{q}}^n_\ell \tag{31}$$

Proof. Here we consider the errors caused in the previous approximations in the flux terms. Using $\mathbf{q}^* = \mathbf{q} + \Delta t f(\mathbf{q}) - \epsilon_1$ inside the flux term:

$$\epsilon_2 = \hat{\mathbf{q}}^n_\ell + \Delta t_\ell f(\hat{\mathbf{q}}^n_\ell) - P_{\ell+1\to\ell}\left[\bar{\mathbf{q}}^n_{\ell+1} + \Delta t_{\ell+1} f(\bar{\mathbf{q}}^n_{\ell+1}) + \Delta t_{\ell+1} f(\hat{\mathbf{q}}^n_{\ell+1} + \Delta t_{\ell+1} f(\hat{\mathbf{q}}^n_{\ell+1}) - \epsilon_1)\right] \tag{32}$$

Considering $f(\bar{\mathbf{q}^n}_{\ell+1}) = \lambda \bar{\mathbf{q}^n}_{\ell+1}$ we have

$$\epsilon_2 = \hat{\mathbf{q}}^n_\ell + \Delta t_\ell \lambda \hat{\mathbf{q}}^n_\ell - P_{\ell+1\to\ell}\left[\hat{\mathbf{q}}^n_{\ell+1} + \Delta t_{\ell+1}\lambda \hat{\mathbf{q}}^n_{\ell+1} + \Delta t_{\ell+1}\lambda(\hat{\mathbf{q}}^n_{\ell+1} + \Delta t_{\ell+1}\lambda(\hat{\mathbf{q}}^n_{\ell+1}))\right] \tag{33}$$

Applying the projection operator we get

$$\epsilon_2 = \hat{\mathbf{q}}^n_\ell + \Delta t_\ell \lambda \hat{\mathbf{q}}^n_\ell - \hat{\mathbf{q}}^n_\ell - \Delta t_{\ell+1}\lambda \hat{\mathbf{q}}^n_\ell - \Delta t_{\ell+1}\lambda\left(\hat{\mathbf{q}}^n_\ell + \Delta t_{\ell+1}\lambda \hat{\mathbf{q}}^n_\ell\right) \tag{34}$$

$$\epsilon_2 = \Delta t_\ell \lambda \hat{\mathbf{q}}^n_\ell - \frac{\Delta t_\ell}{2}\lambda \hat{\mathbf{q}}^n_\ell - \frac{\Delta t_\ell}{2}\lambda\left(\hat{\mathbf{q}}^n_\ell + \frac{\Delta t_\ell}{2}\lambda \hat{\mathbf{q}}^n_\ell\right) \tag{35}$$

$$\epsilon_2 = -\frac{\Delta t_\ell^2 \lambda^2}{4}\hat{\mathbf{q}}^n_\ell \tag{36}$$

Once the interpolation errors are obtained, the second Runge–Kutta step is performed for the finer scale:

$$\bar{\mathbf{q}}^{**} = \frac{3}{4}\bar{\mathbf{q}}^n + \frac{1}{4}\bar{\mathbf{q}}^* + \frac{1}{4}\Delta t^n f\left(t^n + \Delta t^n, \bar{\mathbf{q}}^*\right) \tag{37}$$

$$\bar{\mathbf{q}}^{**} = \frac{3}{4}\hat{\mathbf{q}}^n + \frac{1}{4}\hat{\mathbf{q}}^* + \frac{1}{4}\Delta t^n f\left(\hat{\mathbf{q}}^* - \epsilon_1\right) \tag{38}$$

$$\bar{\mathbf{q}}^{**} = \frac{3}{4}\hat{\mathbf{q}}^n + \frac{1}{4}\hat{\mathbf{q}}^* + \frac{1}{4}\Delta t^n f\left(\hat{\mathbf{q}}^*\right) \tag{39}$$

$$\bar{\mathbf{q}}^{**} = \hat{\mathbf{q}}^{**} \tag{40}$$

We observe that the $\bar{\mathbf{q}}^{**}$ solution does not have any interpolation errors due to the LT scheme. Then the second order RK step is performed for the coarser scale:

$$\bar{\mathbf{q}}^{**} = \frac{3}{4}\bar{\mathbf{q}}^n + \frac{1}{4}\bar{\mathbf{q}}^* + \frac{1}{4}\Delta t^n f\left(t^n + \Delta t^n, \bar{\mathbf{q}}^*\right) \tag{41}$$

$$\bar{\mathbf{q}}^{**} = \frac{3}{4}\hat{\mathbf{q}}^n + \frac{1}{4}\hat{\mathbf{q}}^* + \frac{1}{4}\Delta t^n f\left(\hat{\mathbf{q}}^* - \epsilon_2\right) \tag{42}$$

$$\bar{\mathbf{q}}^{**} = \frac{3}{4}\hat{\mathbf{q}}^n + \frac{1}{4}\hat{\mathbf{q}}^* + \frac{1}{4}\Delta t^n \lambda \hat{\mathbf{q}}^* - \frac{1}{4}\Delta t^n \lambda \epsilon_2 \tag{43}$$

$$\bar{\mathbf{q}}^{**} = \frac{3}{4}\hat{\mathbf{q}}^n + \frac{1}{4}\hat{\mathbf{q}}^* + \frac{z}{4}\hat{\mathbf{q}}^* - \frac{\Delta t_\ell \lambda}{4}\epsilon_2 \tag{44}$$

$$\bar{\mathbf{q}}^{**} = \hat{\mathbf{q}}^{**} - \frac{\Delta t_\ell \lambda}{4}\epsilon_2 \tag{45}$$

Here, the approximation has an error of $-\frac{\Delta t_\ell \lambda}{4}\epsilon_2$.

4.4.2. Approximations after the second RK step

In the following, we compute the errors obtained from the predictions and projections after the second RK step, as performed in Section 4.1.2. The approximations here are the NERK solution at $t^n + \frac{1}{4}\Delta t_{\ell-1}$ to predict the solution $\bar{\mathbf{q}}^{**}_\ell$ on finer scales and the projections of the RK3 evolution of the finer leaves to approximate $\bar{\mathbf{q}}^{**}_{\ell-1}$. The first one is given by:

- ϵ_3: Prediction error using NERK approximation

$$\epsilon_3 = \hat{\mathbf{q}}^{**}_\ell - P_{\ell-1\to\ell}\left[\bar{\mathbf{q}}^n_{\ell-1,\theta=\frac{1}{4}}\right] = \left(\frac{\Delta t_\ell^2 \lambda^2}{8} - \frac{\Delta t_\ell^3 \lambda^3}{16}\right)\hat{\mathbf{q}}^n_\ell \tag{46}$$

Proof. Considering the NERK equation for $\theta = \frac{1}{4}$ given in Section 3.2 we get,

$$\epsilon_3 = \hat{\mathbf{q}}_\ell^{**} - P_{\ell-1\to\ell}\left[\bar{\mathbf{q}}_{\ell-1}^n + \frac{7}{32}\Delta t_{\ell-1} f(\bar{\mathbf{q}}_{\ell-1}^n) + \frac{1}{32}\Delta t_{\ell-1} f(\bar{\mathbf{q}}_{\ell-1}^*)\right] \tag{47}$$

Then, we consider the error ϵ_2 in the flux computations for the coarser scale:

$$\epsilon_3 = \hat{\mathbf{q}}_\ell^n\left(1 + \frac{\Delta t_\ell\lambda}{2} + \frac{\Delta t_\ell^2\lambda^2}{4}\right) - P_{\ell-1\to\ell}\left[\bar{\mathbf{q}}_{\ell-1}^n + \frac{7}{32}\Delta t_{\ell-1}\lambda\bar{\mathbf{q}}_{\ell-1}^n + \frac{1}{32}\Delta t_{\ell-1}\lambda\left(\hat{\mathbf{q}}_{\ell-1}^* - \epsilon_2\right)\right] \tag{48}$$

$$\epsilon_3 = \hat{\mathbf{q}}_\ell^n\left(\frac{\Delta t_\ell\lambda}{16} + \frac{\Delta t_\ell^2\lambda^2}{4}\right) - \frac{\Delta t_\ell\lambda}{16}P_{\ell-1\to\ell}\left[\hat{\mathbf{q}}_{\ell-1}^* - \epsilon_2\right] \tag{49}$$

$$\epsilon_3 = \hat{\mathbf{q}}_\ell^n\left(\frac{\Delta t_\ell\lambda}{16} + \frac{\Delta t_\ell^2\lambda^2}{4}\right) - \frac{\Delta t_\ell\lambda}{16}P_{\ell-1\to\ell}\left[\hat{\mathbf{q}}_{\ell-1}^n + \Delta t_{\ell-1} f(\hat{\mathbf{q}}_{\ell-1}^n) + \frac{\Delta t_{\ell-1}^2\lambda^2}{4}\hat{\mathbf{q}}_{\ell-1}^n\right] \tag{50}$$

$$\epsilon_3 = \frac{\Delta t_\ell^2\lambda^2}{4}\hat{\mathbf{q}}_\ell^n - \frac{\Delta t_\ell^2\lambda^2}{8}\hat{\mathbf{q}}_\ell^n - \frac{\Delta t_\ell\lambda}{16}P_{\ell-1\to\ell}\left[\frac{\Delta t_{\ell-1}^2\lambda^2}{4}\hat{\mathbf{q}}_{\ell-1}^n\right] \tag{51}$$

$$\epsilon_3 = \left(\frac{\Delta t_\ell^2\lambda^2}{8} - \frac{\Delta t_\ell^3\lambda^3}{16}\right)\hat{\mathbf{q}}_\ell^n \tag{52}$$

Afterwards we perform the third RK step for the finer leaf.

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\bar{\mathbf{q}}^n + \frac{2}{3}\bar{\mathbf{q}}^{**} + \frac{2}{3}\Delta t^n f\left(t^n + \frac{1}{2}\Delta t^n, \bar{\mathbf{q}}^{**}\right) \tag{53}$$

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\hat{\mathbf{q}}^n + \frac{2}{3}\hat{\mathbf{q}}^{**} + \frac{2z}{3}\left(\hat{\mathbf{q}}^{**} - \epsilon_3\right) \tag{54}$$

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\hat{\mathbf{q}}^n + \frac{2}{3}\left(1 + \frac{\Delta t_\ell\lambda}{2} + \frac{\Delta t_\ell^2\lambda^2}{4}\right)\hat{\mathbf{q}}^n + \frac{2\Delta t_\ell\lambda}{3}\left[\left(1 + \frac{\Delta t_\ell\lambda}{2} + \frac{\Delta t_\ell^2\lambda^2}{4}\right)\hat{\mathbf{q}}^n - \epsilon_3\right] \tag{55}$$

$$\bar{\mathbf{q}}^{n+1} = \left(1 + \Delta t_\ell\lambda + \frac{\Delta t_\ell^2\lambda^2}{2} + \frac{\Delta t_\ell^3\lambda^3}{6}\right)\hat{\mathbf{q}}^n - \frac{2\Delta t_\ell\lambda}{3}\left(\frac{\Delta t_\ell^2\lambda^2}{8} - \frac{\Delta t_\ell^3\lambda^3}{16}\right)\hat{\mathbf{q}}^n \tag{56}$$

$$\bar{\mathbf{q}}^{n+1} = \left(1 + \Delta t_\ell\lambda + \frac{\Delta t_\ell^2\lambda^2}{2} + \frac{\Delta t_\ell^3\lambda^3}{12} + \frac{\Delta t_\ell^4\lambda^4}{24}\right)\hat{\mathbf{q}}^n \tag{57}$$

This shows that the third order accuracy is lost in the evolution of the finer leaf.

- ϵ_4: Projection error using finer leaf evolution.

$$\epsilon_4 = \hat{\mathbf{q}}_\ell^{**} - P_{\ell+1\to\ell}\left[\bar{\mathbf{q}}_{\ell+1}^{n+1}\right] = \left(\frac{\Delta t_\ell^2\lambda^2}{8} - \frac{\Delta t_\ell^3\lambda^3}{96} - \frac{\Delta t_\ell^4\lambda^4}{384}\right)\hat{\mathbf{q}}_\ell^n \tag{58}$$

Proof. Using Equation (57) to represent the time evolution for the finer leaf, we have,

$$\epsilon_4 = \hat{\mathbf{q}}_\ell^{**} - P_{\ell+1\to\ell}\left[\left(1 + \Delta t_{\ell+1}\lambda + \frac{(\Delta t_{\ell+1}\lambda)^2}{2} + \frac{(\Delta t_{\ell+1}\lambda)^3}{12} + \frac{(\Delta t_{\ell+1}\lambda)^4}{24}\right)\hat{\mathbf{q}}_{\ell+1}^n\right] \tag{59}$$

$$\epsilon_4 = \hat{\mathbf{q}}_\ell^{**} - P_{\ell+1\to\ell}\left[\left(1 + \frac{\Delta t_\ell\lambda}{2} + \frac{(\Delta t_\ell\lambda)^2}{8} + \frac{(\Delta t_\ell\lambda)^3}{96} + \frac{(\Delta t_\ell\lambda)^4}{384}\right)\hat{\mathbf{q}}_{\ell+1}^n\right] \tag{60}$$

$$\epsilon_4 = \left(1 + \frac{\Delta t_\ell\lambda}{2} + \frac{\Delta t_\ell^2\lambda^2}{4}\right)\hat{\mathbf{q}}_\ell^n - \left(1 + \frac{\Delta t_\ell\lambda}{2} + \frac{\Delta t_\ell^2\lambda^2}{8} + \frac{\Delta t_\ell^3\lambda^3}{96} + \frac{\Delta t_\ell^4\lambda^4}{384}\right)\hat{\mathbf{q}}_\ell^n \tag{61}$$

$$\epsilon_4 = \left(\frac{\Delta t_\ell^2\lambda^2}{8} - \frac{\Delta t_\ell^3\lambda^3}{96} - \frac{\Delta t_\ell^4\lambda^4}{384}\right)\hat{\mathbf{q}}_\ell^n \tag{62}$$

And finally, performing the third RK step for the coarser leaf, we get

Table 1
Convergence order for the FV, MR and MRLT methods.

Δt ($\times 10^{-4}$)	Method						
	FV/RK2	MR/RK2	MRLT/RK2	MRLT/NERK2	FV/RK3	MR/RK3	MRLT/NERK3
1.6	1.9991	1.9984	1.0794	2.0154	3.0000	3.0030	1.7895
0.8	2.0003	2.0010	0.9433	2.0002	3.0077	3.0017	1.9152

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\bar{\mathbf{q}}^{n} + \frac{2}{3}\bar{\mathbf{q}}^{**} + \frac{2}{3}\Delta t_\ell f\left[t^n + \frac{1}{2}\Delta t_\ell, \bar{\mathbf{q}}^{**}\right] \tag{63}$$

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\hat{\mathbf{q}}^{n} + \frac{2}{3}\left(\hat{\mathbf{q}}^{**} - \frac{\Delta t_\ell \lambda}{4}\epsilon_2\right) + \frac{2\Delta t_\ell \lambda}{3}\left[\hat{\mathbf{q}}^{**} - \epsilon_4\right] \tag{64}$$

$$\bar{\mathbf{q}}^{n+1} = \frac{1}{3}\hat{\mathbf{q}}^{n} + \frac{2}{3}\left(\hat{\mathbf{q}}^{**} + \frac{\Delta t_\ell^3 \lambda^3}{16}\hat{\mathbf{q}}_\ell^n\right) + \frac{2\Delta t_\ell \lambda}{3}\left[\hat{\mathbf{q}}^{**} - \left(\frac{\Delta t_\ell^2 \lambda^2}{8} - \frac{\Delta t_\ell^3 \lambda^3}{96} - \frac{\Delta t_\ell^4 \lambda^4}{384}\right)\hat{\mathbf{q}}_\ell^n\right] \tag{65}$$

$$\bar{\mathbf{q}}^{n+1} = \hat{\mathbf{q}}^{n+1} + \frac{\Delta t_\ell^3 \lambda^3}{24}\hat{\mathbf{q}}_\ell^n + \left(-\frac{\Delta t_\ell^3 \lambda^3}{12} + \frac{\Delta t_\ell^4 \lambda^4}{144} + \frac{\Delta t_\ell^5 \lambda^5}{576}\right)\hat{\mathbf{q}}_\ell^n \tag{66}$$

$$\bar{\mathbf{q}}^{n+1} = \left(1 + \Delta t_\ell \lambda + \frac{\Delta t_\ell^2 \lambda^2}{2} + \frac{\Delta t_\ell^3 \lambda^3}{8} + \frac{\Delta t_\ell^4 \lambda^4}{144} + \frac{\Delta t_\ell^5 \lambda^5}{576}\right)\hat{\mathbf{q}}^n. \tag{67}$$

This result also shows a loss of the third order accuracy, since the third order term of the Taylor series $\frac{\Delta t_\ell^3 \lambda^3}{6}$ is not found.

4.5. Discussion on the numerical convergence

To study the numerical convergence of local time stepping we consider the advection equation

$$\frac{\partial Q}{\partial t} + \frac{\partial Q}{\partial x} = 0 \qquad x \in [0, 1] \tag{68}$$

with periodic boundary conditions and a Gaussian initial condition, given by $Q(x) = \exp\left(-100\,(x - 0.25)^2\right)$. The simulation is performed for one time cycle. In order to avoid errors due to the MR scheme, we performed the adaptive simulations using two fixed grids in the domain. The first one in the interval $[0, 0.5]$ with $\Delta x = 1/512$, and the second in the interval $[0.5, 1.0]$ with $\Delta x = 1/256$, are both fixed during the entire simulation. We also use a centred numerical flux.

The convergence orders is computed using a self convergence method, obtaining the rate for which the solution of the MR and MRLT methods converges to a solution as $\Delta t \to 0$. For that, we perform simulations using subsequently smaller time steps, each one having half of the time step used in the previous simulation. The convergence rate is obtained from the following ratio:

$$\left\| \frac{\bar{\mathbf{q}}_{\Delta t} - \bar{\mathbf{q}}_{\frac{\Delta t}{2}}}{\bar{\mathbf{q}}_{\frac{\Delta t}{2}} - \bar{\mathbf{q}}_{\frac{\Delta t}{4}}} \right\| = \frac{C\Delta t^p - C\left(\frac{\Delta t}{2}\right)^p + O(\Delta t^{p+1})}{C\left(\frac{\Delta t}{2}\right)^p - C\left(\frac{\Delta t}{4}\right)^p + O(\Delta t^{p+1})} = \frac{1 - 2^{-p} + O(\Delta t)}{2^{-p} - 2^{-2p} + O(\Delta t)} = 2^p + O(\Delta t) \tag{69}$$

where $\bar{\mathbf{q}}_{\Delta t}$ is the solution of a simulation using a time step Δt and p is the order of the method, which is approximated using the logarithm:

$$p \approx \log_2 \left\| \frac{\bar{\mathbf{q}}_{\Delta t} - \bar{\mathbf{q}}_{\frac{\Delta t}{2}}}{\bar{\mathbf{q}}_{\frac{\Delta t}{2}} - \bar{\mathbf{q}}_{\frac{\Delta t}{4}}} \right\|. \tag{70}$$

The convergence order obtained for the FV, MR and MRLT methods are given in Table 1. In order to check that $p \to 0$ as $\Delta t \to 0$, we perform this test using two different values for the coarsest Δt. We observe that FV/RK2 and FV/RK3 yield second and third order time discretisations, respectively, as expected. For MRLT/NERK2 and MRLT/NERK3 we obtained the expected second order, in particular for the MRLT/NERK3 the second order is justified by the approximation errors ϵ_3 and ϵ_4 which caused the loss of the third order, as shown in the Equations (57) and (67). Another reason for this loss in accuracy was the fact that due to the order barrier for NERK methods [22], a third order solution at instant $t^n + \frac{\Delta t_\ell}{2}$ could not be produced by a three stage method. Thus, when a leaf performs its second time evolution inside the LT cycle, it may use the second order solution from a coarser leaf, causing the loss from third to second order. This issue also justifies the observed loss in accuracy for the MRLT/RK2 method. Here, the coarser leaf produces a first order solution at instant $t^n + \frac{\Delta t_\ell}{2}$ to be used in the second evolution of the finer leaf.

5. Numerical experiments

In this section we present some comparative results of the proposed MRLT/NERK method with the MR, MRLT methods given in [6] and also the traditional FV method on a uniform grid. These methods are applied to solve the two-dimensional Burgers equation, one and three-dimensional reaction–diffusion equations and finally the two-dimensional compressible Euler equations. We use the AUSM+ scheme [17] to compute the numerical flux in Burgers and Euler equations. To compute the advective term for the reaction–diffusion equation, we use the McCormack scheme [27]. The errors are computed in the discrete L_1 norm on the fine grid as:

$$e_{L_1}^{\text{method}} = \frac{1}{2^{Ld}} \sum_{i=0}^{2^{Ld}-1} \left\| \bar{\mathbf{q}}_i^{\text{ref}} - \bar{\mathbf{q}}_i^{\text{method}} \right\| \tag{71}$$

where $\bar{\mathbf{q}}^{\text{ref}}$ is the FV/RK3 reference solution of the corresponding problem and $\bar{\mathbf{q}}^{\text{method}}$ is the solution obtained with the analyzed method.

To compare the performance of two adaptive methodologies in terms of CPU time reduction versus accuracy loss, a cost value μ^{method} is defined for each adaptive method as:

$$\mu^{\text{method}} = \frac{e_{L_1}^{\text{method}} \cdot t_{\mathbf{CPU}}^{\text{method}}}{t_{\mathbf{CPU}}^{\text{FV}}}, \tag{72}$$

where $t_{\mathbf{CPU}}^{\text{method}}$ is the CPU time obtained for the analyzed method and $t_{\mathbf{CPU}}^{\text{FV}}$ is the CPU time of the FV method with the same number of scales L and Runge–Kutta of the same order.

The ratio between the cost of different adaptive methods yields the parameter λ, used to measure the advantage of one method compared with the other. In this work, the parameter λ is used to compare the proposed MRLT/NERK methods with the MR and MRLT methods, defined as:

$$\lambda_{\text{MRLT/NERK}}^{\text{method}} = \frac{\mu^{\text{method}}}{\mu^{\text{MRLT/NERK}}}. \tag{73}$$

If the parameter λ is larger than 1, the MRLT/NERK approach is considered to be advantageous over the other method. In case of $\lambda < 1$, the MRLT/NERK approach is considered to be disadvantageous over the other method, and in case of $\lambda = 1$, the methods are considered to be equivalent.

All the simulations are performed using a fixed threshold value ϵ in order to simplify the experiments. In [7] computations are presented using a MR methodology with ϵ values which depend of the refinement level.

5.1. Two-dimensional Burgers equation

The Burgers equation is a non-linear PDE which represents a simple model for turbulence and is used in astrophysical applications. The inviscid model, in the two-dimensional case, is given by the following equation:

$$\frac{\partial Q}{\partial t} + \frac{1}{2}\left(\frac{\partial (Q^2)}{\partial x} + \frac{\partial (Q^2)}{\partial y} \right) = 0 \qquad (x, y) \in \Omega = [0, 1] \times [0, 1]. \tag{74}$$

The initial condition used in this work is $Q_0(x, y) = \sin(2\pi x)\sin(2\pi y)$, with Dirichlet boundary conditions given by $Q(x, 0) = Q(x, 1) = Q(0, y) = Q(1, y) = 0$.

All simulations are performed with a Courant number $\sigma = 0.5$ and a threshold $\epsilon = 0.01$ until the time instant $t_f = 0.9$. The reference solution for this case is obtained using refinement level $L = 12$.

Fig. 7 shows the reference solution and the solution obtained with the MRLT/RK2 and MRLT/NERK2. The respective difference, in modulus, with respect to the reference solution and adaptive grids at the end of the computation are also shown. The method MRLT/RK2 case exhibits larger errors close to the shocks, especially in the peak of the structure and in its background. The other methods present solutions closer to the one obtained with the MRLT/NERK2 method.

The results are compared with the solution in an uniform grid at the same level using a L_1 norm, showing perturbation errors. These errors, CPU time and memory compression are summarized in Table 2. The CPU time and memory of the adaptive methods are given in percentage of the number of leaves used in the FV method with the same number of scales and the same Runge–Kutta scheme. For the two-dimensional Burgers equation, the proposed MRLT/NERK methods present a slight gain in precision and a significant gain in CPU time in relation to the other adaptive methodologies. However, the MRLT schemes with NERK time integration require more memory, which decreases when increasing the resolution, *i.e.* for increasing L.

The parameters λ obtained for the MRLT/NERK methods compared to the MR and MRLT methods are presented in Table 3. For most of the experiments, the proposed methods yield values of λ between 2 and 3. This shows that the NERK-based methods are significantly more efficient than the RK-based MR and MRLT methods.

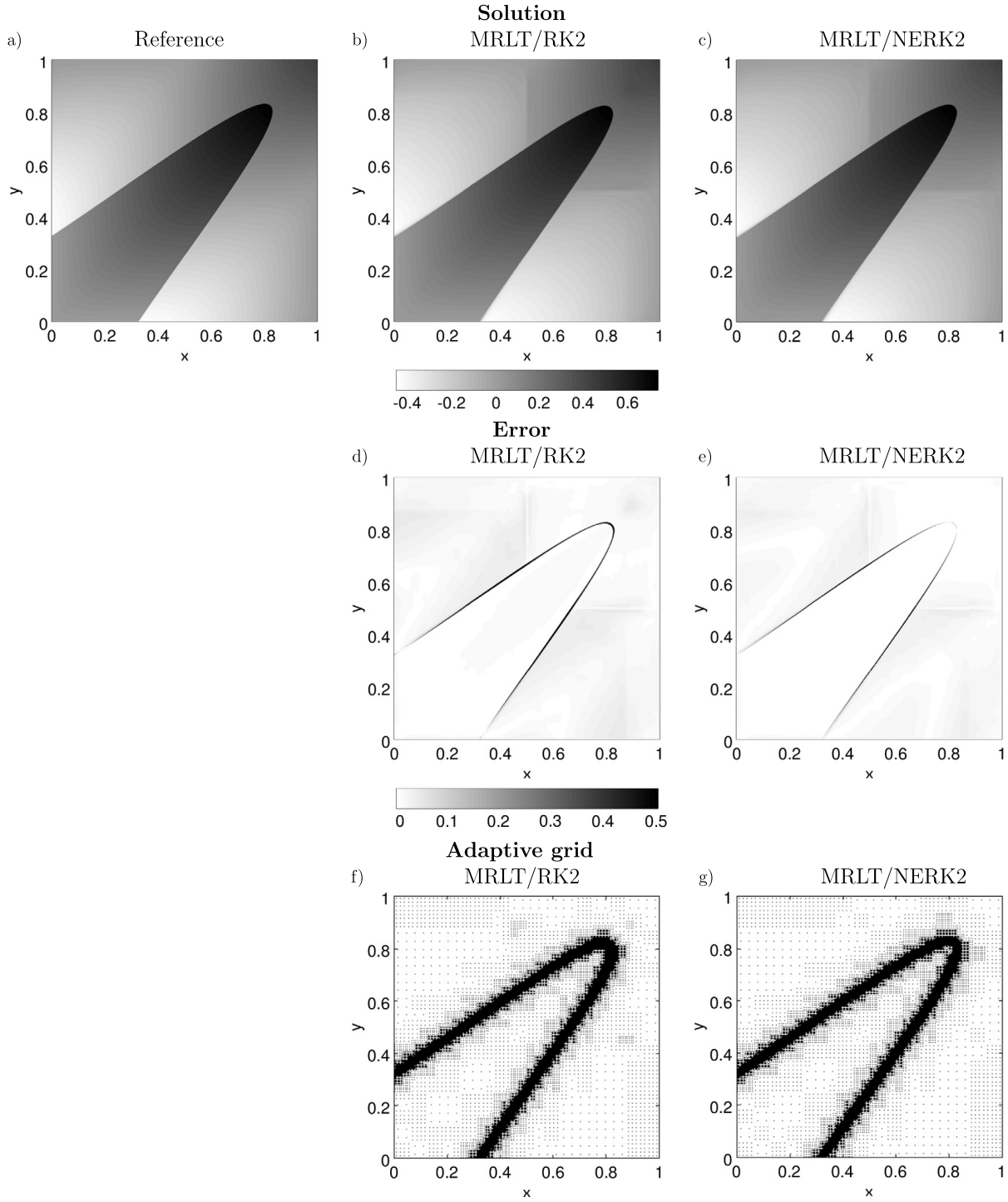


Fig. 7. Reference solution for the two-dimensional Burgers equation a), the solutions obtained by the MRLT/RK2 b) and MRLT/NERK2 c) methods with its respective errors and the corresponding adaptive grids. For all cases we use $L = 10$ and the time instant is $t_f = 0.9$.

5.2. Reaction–diffusion equations

We consider reaction–diffusion equations in one and three space dimensions and study the formation of a flame front ignited with a spark in an environment with flammable premixed gas. This kind of problem is a prototype of non-linear parabolic equations with a non-linearity in the source term, see e.g. [6,26].

5.2.1. One-dimensional case

In the one-dimensional case, considering equal mass and heat diffusion, this problem can be modelled by the following equation:

Table 2
Two-dimensional Burgers equation: L_1 errors, CPU time and memory compression.

Finest scale level	Method	Error ($\times 10^{-2}$) Q	CPU time (% FV)	Memory (% FV)
$L = 8$	MR/RK2	1.5032	68.4	19.5
	MRLT/RK2	1.7701	67.8	19.5
	MRLT/NERK2	1.2958	40.3	24.2
	MR/RK3	1.5045	184.9	19.4
	MRLT/NERK3	1.2932	94.7	24.2
$L = 9$	MR/RK2	1.3615	34.7	8.9
	MRLT/RK2	1.6645	32.5	8.9
	MRLT/NERK2	1.1457	16.7	10.3
	MR/RK3	1.3614	95.4	8.9
	MRLT/NERK3	1.1423	42.7	10.3
$L = 10$	MR/RK2	1.2890	14.7	4.1
	MRLT/RK2	1.6016	13.5	4.1
	MRLT/NERK2	1.0740	6.7	4.4
	MR/RK3	1.2893	15.3	4.1
	MRLT/NERK3	1.0699	6.4	4.4

Note: All adaptive computations use $\epsilon = 10^{-2}$; the final time is $t_f = 0.9$. The computations have been carried out on an Intel Core™ *i*7 CPU 2.67 GHz. FV/RK2 CPU time: 2.0 min ($L = 8$); 15.7 min ($L = 9$); 2.3 h ($L = 10$). FV/RK3 CPU time: 1.2 min ($L = 8$); 8.0 min ($L = 9$); 3.3 h ($L = 10$).

Table 3
Two-dimensional Burgers equation: computational gain λ of the MRLT/NERK methods with respect to the MR and MRLT methods.

Finest scale level		MR/RK2	MRLT/RK2	MR/RK3
$L = 8$	MRLT/NERK2	1.96	2.29	–
	MRLT/NERK3	–	–	2.27
$L = 9$	MRLT/NERK2	2.46	2.82	–
	MRLT/NERK3	–	–	2.66
$L = 10$	MRLT/NERK2	2.63	3.00	–
	MRLT/NERK3	–	–	2.88

$$\frac{\partial T}{\partial t} + v_f \frac{\partial T}{\partial x} = \frac{\partial^2 T}{\partial x^2} + \omega(T) \quad \text{for } x \in (-15, 15) \tag{75}$$

where the function $T(x,t)$ is the dimensionless temperature normalized between 0 (unburned gas) and 1 (burned gas), $v_f = \int \omega \, dx$ is the flame velocity and $\omega(T)$ is the chemical reaction rate, given by:

$$\omega(T) = \frac{Ze^2}{2}(1-T)\exp\left(\frac{Ze(1-T)}{\tau(1-T)-1}\right) \tag{76}$$

where Ze is a dimensionless activation energy, known as Zeldovich number, and τ is the burnt-unburnt temperature ratio. In this one-dimensional case, where we use unitary Lewis number, the unburned gas concentration Y is defined as $Y = 1 - T$.

In the numerical experiments, the following initial condition is used:

$$T(x,0) = \begin{cases} 1, & \text{if } x \leq 1 \\ \exp(1-x), & \text{otherwise} \end{cases} \tag{77}$$

with boundary conditions given by:

$$\frac{\partial T}{\partial x}(-15,t) = 0, \qquad T(15,t) = 0 \tag{78}$$

The subsequent simulations are performed with the parameters $Ze = 10$, $\tau = 0.8$ and a Courant number $\sigma = 0.5$ until the final time instant $t_f = 5.0$. The adaptive simulations are performed with a threshold $\epsilon = 0.01$. The reference solution for this case is obtained using the same Courant number and a refinement level $L = 13$. Fig. 8 shows the reference solution and the solution obtained by the MRLT/RK2 and MRLT/NERK2 methods with its respective errors and the corresponding final grids. In this one-dimensional case, the adaptive grid is represented by the position of each cell (x-axis) and its refinement level (y-axis).

The adaptive methods present a larger error in the flame front region, especially for the variable ω. Among the adaptive methods, MRLT/RK2 has the smallest errors, while the other methods present very similar errors. However, the MRLT/NERK methods still requires the lowest CPU time. These times and L_1 errors are assembled in Table 4.

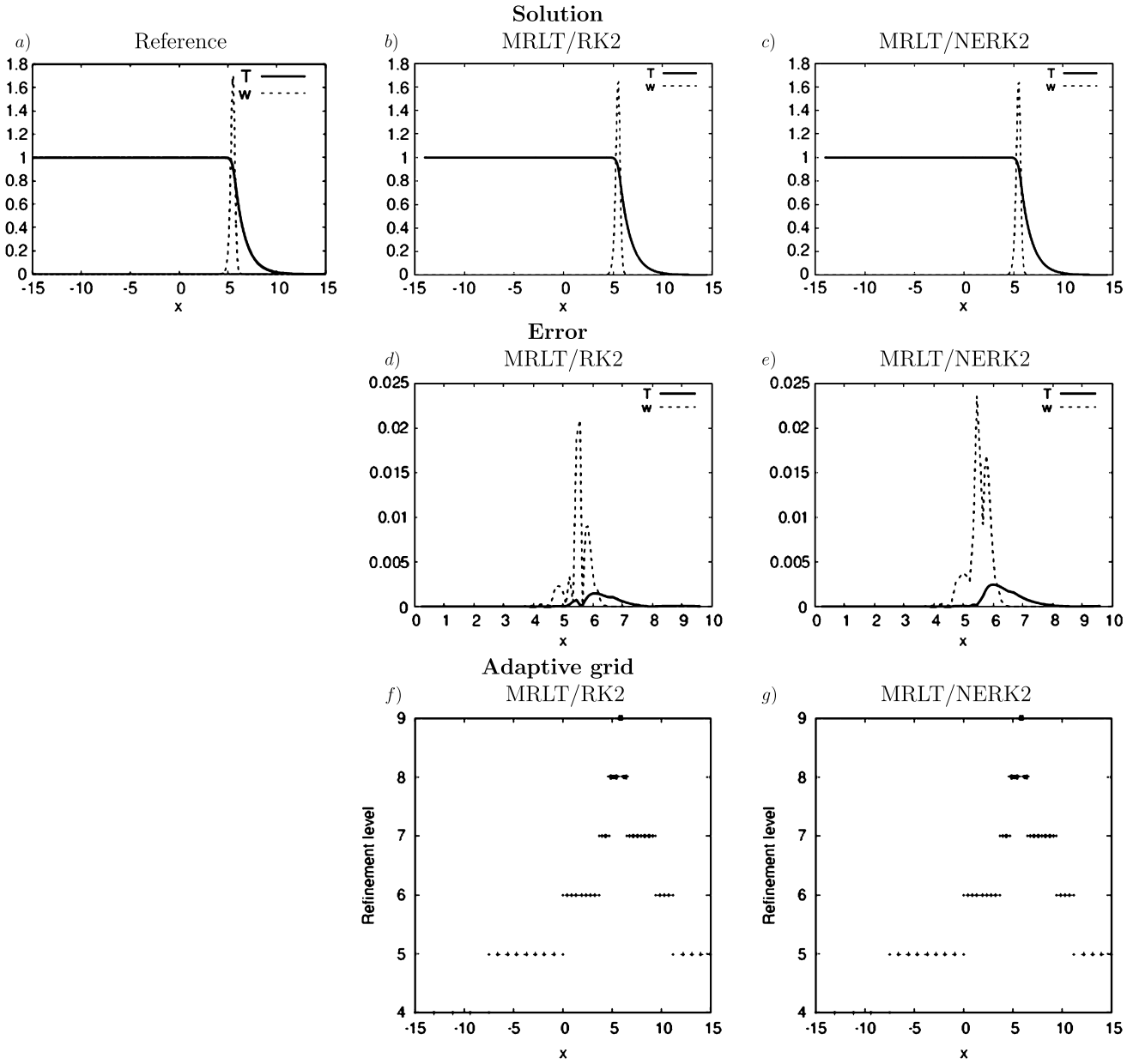


Fig. 8. Reference solution of the one-dimensional reaction–diffusion equations a) and the solutions obtained by the MRLT/RK2 (b) and MRLT/NERK2 (c) methods with its respective errors (d), (e) and final adaptive grids (f), (g). For all cases we use $L = 13$.

The computational gain of the MRLT/NERK methods compared to the MR and MRLT methods are given in Table 5. In this case, the MRLT/NERK methods yields the most expressive results compared to the MR methods in terms of computational gain. The gain of the MRLT/NERK2 method compared with the MRLT/RK2 method is small for the cases $L = 12$ and 13. The corresponding results are presented in Table 5.

5.2.2. Three-dimensional case

In the three-dimensional case, the reaction–diffusion equations read:

$$\frac{\partial T}{\partial t} = \nabla^2 T + \omega - s \tag{79a}$$

$$\frac{\partial Y}{\partial t} = \frac{1}{Le}\nabla^2 Y - \omega \tag{79b}$$

where Le denotes the Lewis number, which defines the ratio of mass and heat diffusion and with the chemical reaction rate ω:

$$\omega(T, Y) = \frac{Ze^2}{2Le} Y \exp\left(\frac{Ze(T-1)}{1+\tau(T-1)}\right) \tag{80}$$

Table 4
One-dimensional reaction–diffusion equations: L_1 errors, CPU time and memory compression.

Finest scale level	Method	Error ($\times 10^{-4}$)			CPU time	Memory
		T	Y	ω	(%FV)	
$L = 11$	MR/RK2	5.045	5.045	24.564	31.9	3.1
	MRLT/RK2	5.666	5.666	30.192	18.0	3.1
	MRLT/NERK2	4.380	4.380	21.308	7.2	3.1
	MR/RK3	5.045	5.045	24.566	31.3	3.1
	MRLT/NERK3	4.543	4.543	22.173	5.3	3.1
$L = 12$	MR/RK2	5.043	5.043	24.559	15.9	1.5
	MRLT/RK2	1.780	1.780	14.256	8.0	1.5
	MRLT/NERK2	4.728	4.728	23.107	2.8	1.5
	MR/RK3	5.043	5.043	24.560	16.0	1.5
	MRLT/NERK3	4.755	4.755	23.261	2.3	1.5
$L = 13$	MR/RK2	5.054	5.054	24.609	7.9	0.7
	MRLT/RK2	2.904	2.904	15.062	3.9	0.7
	MRLT/NERK2	4.900	4.900	23.903	1.4	0.7
	MR/RK3	5.054	5.054	24.609	7.9	0.7
	MRLT/NERK3	4.904	4.904	23.923	1.1	0.7

Note: All adaptive computations use $\epsilon = 10^{-2}$; final time: $t_f = 5.0$. Computed on an Intel Core™ *i*7 CPU 2.67 GHz. FV/RK2 CPU time: 51.4 min ($L = 11$); 6.9 h ($L = 12$); 54.6 h ($L = 13$). FV/RK3 CPU time: 51.4 min ($L = 11$); 6.9 h ($L = 12$); 54.6 h ($L = 13$).

Table 5
One-dimensional reaction–diffusion equations: computational gain, for the variable T, of the proposed MRLT/NERK methods compared to the MR and MRLT methods.

Finest scale level		MR/RK2	MRLT/RK2	MR/RK3
$L = 11$	MRLT/NERK2	5.10	3.23	–
	MRLT/NERK3	–	–	6.55
$L = 12$	MRLT/NERK2	6.05	1.07	–
	MRLT/NERK3	–	–	7.37
$L = 13$	MRLT/NERK2	5.82	1.65	–
	MRLT/NERK3	–	–	7.40

According to the Stefan–Boltzmann law, the heat loss due to radiation s is modelled by:

$$s(T) = \kappa \left[(T + \tau^{-1} - 1)^4 - (\tau^{-1} - 1)^4 \right] \tag{81}$$

where κ is a dimensionless radiation coefficient. In this work, we use $\kappa = 0.1$.

The initial condition, described in spherical coordinates, is:

$$T(r, 0) = \begin{cases} 1, & \text{if } r \leq r_0 \\ \exp\left(1 - \frac{r}{r_0}\right), & \text{otherwise} \end{cases} \tag{82}$$

$$Y(r, 0) = \begin{cases} 0, & \text{if } r \leq r_0 \\ 1 - e\left(Le(1 - \frac{r}{r_0})\right), & \text{otherwise} \end{cases} \tag{83}$$

where $r_0 = 1$ is the initial radius of the ellipsoidal flame ball and $r = \sqrt{\frac{X^2}{a^2} + \frac{Y^2}{b^2} + \frac{Z^2}{c^2}}$ with:

$$X = x\cos(\theta) - y\sin(\theta) \tag{84}$$

$$Y = [x\sin(\theta) + y\cos(\theta)]\cos(\phi) - z\sin(\phi) \tag{85}$$

$$Z = [x\sin(\theta) + y\cos(\theta)]\sin(\phi) + z\cos(\phi) \tag{86}$$

The boundary conditions are of homogeneous Neumann type. In these simulations we use the parameters $Ze = 10$, $\tau = 0.64$, $Le = 0.3$, $\theta = \frac{\pi}{3}$, $\phi = \frac{\pi}{4}$, $a = \frac{3}{2}$, $b = \frac{3}{2}$, $c = 3$, a threshold factor $\epsilon = 0.01$ and a Courant number $\sigma = 0.1$ until the final time instant $t_f = 5.0$. The reference solution for this case is obtained using the same Courant number and the refinement level $L = 7$. Fig. 9 shows the isosurface of T for the reference and the MRLT/NERK2 and MRLT/NERK3 methods. It also shows the solutions, difference from the reference solution in modulus, and projections of every cell centre at the plane xz. The adaptive methodologies present higher errors close to the flame ball fronts, which are more rounded in comparison to the FV solutions. The adaptive grids are similar for all adaptive methodologies, with a higher concentration of refined cells in the region of the front and inside the flame ball.

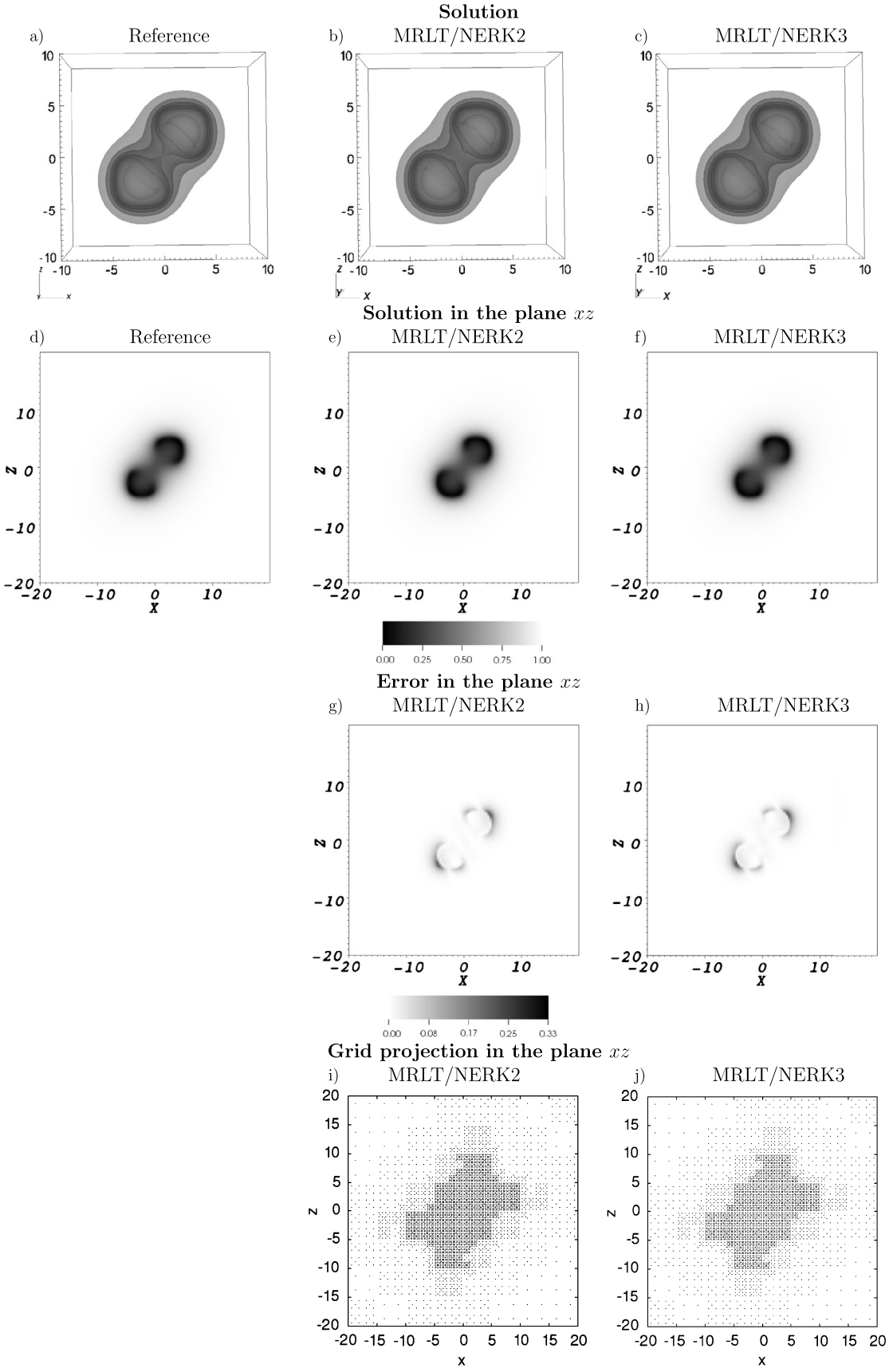


Fig. 9. Isosurface and xz plane reference solution for the variable T in three-dimensions at $t_f = 5.0$. The solutions are obtained with the MRLT/NERK2 and MRLT/NERK3 methods with its respective errors in the xz plane and projections of the centre of the adaptive grid cell onto the xz plane. The isosurface plots were built using 5 linearly scaled values.

Table 6
Three-dimensional reaction–diffusion equations: L_1 errors, CPU time and memory compression.

Finest scale level	Method	Error ($\times 10^{-4}$)			CPU time	Memory
		T	Y	ω	(%FV)	
$L=7$	MR/RK2	4.516	7.338	21.831	15.6	2.5
	MRLT/RK2	4.511	7.267	21.798	14.9	2.5
	MRLT/NERK2	4.513	7.636	22.038	10.9	7.7
	MR/RK3	4.516	7.338	21.831	14.4	2.5
	MRLT/NERK3	4.526	7.369	21.856	5.5	2.5

Note: All adaptive computations use $\epsilon = 10^{-2}$; final time: $t_f = 5.0$. Computed on a Quad-Core AMD Opteron™ CPU 2.4 GHz. FV/RK2 CPU time: 25.8 h ($L=7$). FV/RK3 CPU time: 38.5 h ($L=7$).

Table 7
Three-dimensional reaction–diffusion equations: computational gain, for the variable T, of the MRLT/NERK methods compared to the MR and MRLT methods.

Finest scale level		MR/RK2	MRLT/RK2	MR/RK3
$L=7$	MRLT/NERK2	1.43	1.36	–
	MRLT/NERK3	–	–	2.61

Table 8
Initial condition for two-dimensional Euler equations.

Variables	Quadrant			
	1st	2nd	3rd	4th
Density, ρ	1.00	2.00	1.00	3.00
Pressure, p	1.00	1.00	1.00	1.00
x-velocity, v_x	0.75	0.75	−0.75	−0.75
y-velocity, v_y	−0.50	0.50	0.50	−0.50

The L_1 errors, CPU time and memory compression are summarized in Table 6. In this case, the proposed MRLT/NERK methods present a loss in precision and memory usage with a gain in CPU time in relation to the other adaptive methodologies. However, the parameters λ obtained for the MRLT/NERK methods compared to the MR and MRLT methods, presented in Table 7, still yield favourable values, especially for the third order methods. Thus we find that the MRLT/NERK methods are slightly more efficient than the MR and MRLT methods for the three-dimensional flame ball case, besides a small loss in precision.

5.3. Compressible two-dimensional Euler equations

The Euler equations, which describe the dynamics of a non-ionized gas, are given by the following relations:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{87a}$$

$$\frac{\partial \rho \vec{v}}{\partial t} + \nabla \cdot (\rho \vec{v} \vec{v}) + \nabla p = 0 \tag{87b}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot \left(\vec{v}(E+p) \right) = 0 \tag{87c}$$

where $\vec{v} = (v_x, v_y)$ is the velocity, ρ is the fluid density, p is the pressure, and the energy per mass unity E is given by $E = \rho e + \frac{\rho \|\vec{v}\|^2}{2}$. This system is completed by the equation of state of an ideal gas $p = \frac{\rho T}{\gamma Ma^2}$, where γ is the specific heat ratio, T is the temperature and Ma is the Mach number.

The initial condition used in this numerical test is the classical Lax–Liu configuration #6 [7,16]. In this initial condition configuration, the domain is divided into four quadrants, ordered from 1st to 4th, and defined by the subdomains $[0.5; 1] \times [0.5; 1]$, $[0; 0.5] \times [0.5; 1]$, $[0; 0.5] \times [0; 0.5]$ and $[0.5; 1] \times [0; 0.5]$, respectively. The initial values for each quadrant are given in Table 8. This problem is simulated until the final instant $t_f = 0.25$, using homogeneous Neumann boundary conditions. The numerical parameters are the Courant number $\sigma = 0.5$, $Ma = 1$, and $\gamma = 1.4$ inside the domain $[0; 1] \times [0; 1]$.

Fig. 10 shows the reference solution and the solution obtained with the MRLT/RK2, MRLT/NERK2 methods and its respective difference, in modulus, from the reference. Furthermore, the corresponding adaptive grid using $L = 12$ with CFL $\sigma = 0.5$ is shown.

The L_1 errors, CPU time and memory compression are presented in Table 9. In this case, the proposed MRLT/NERK methods yield a slight gain in precision with a significant gain in CPU time in relation to the other adaptive methodologies. The memory usage of all adaptive methodologies is quite similar. Due to the gain in both precision and CPU time, the parameters λ for the MRLT/NERK methods compared with the MR and MRLT methods, presented in Table 10, have expressive values, around 3 for most of the experiments. Thus, using this metric, the proposed methods for the two-dimensional Euler case are significantly more efficient than the MR and MRLT methods.

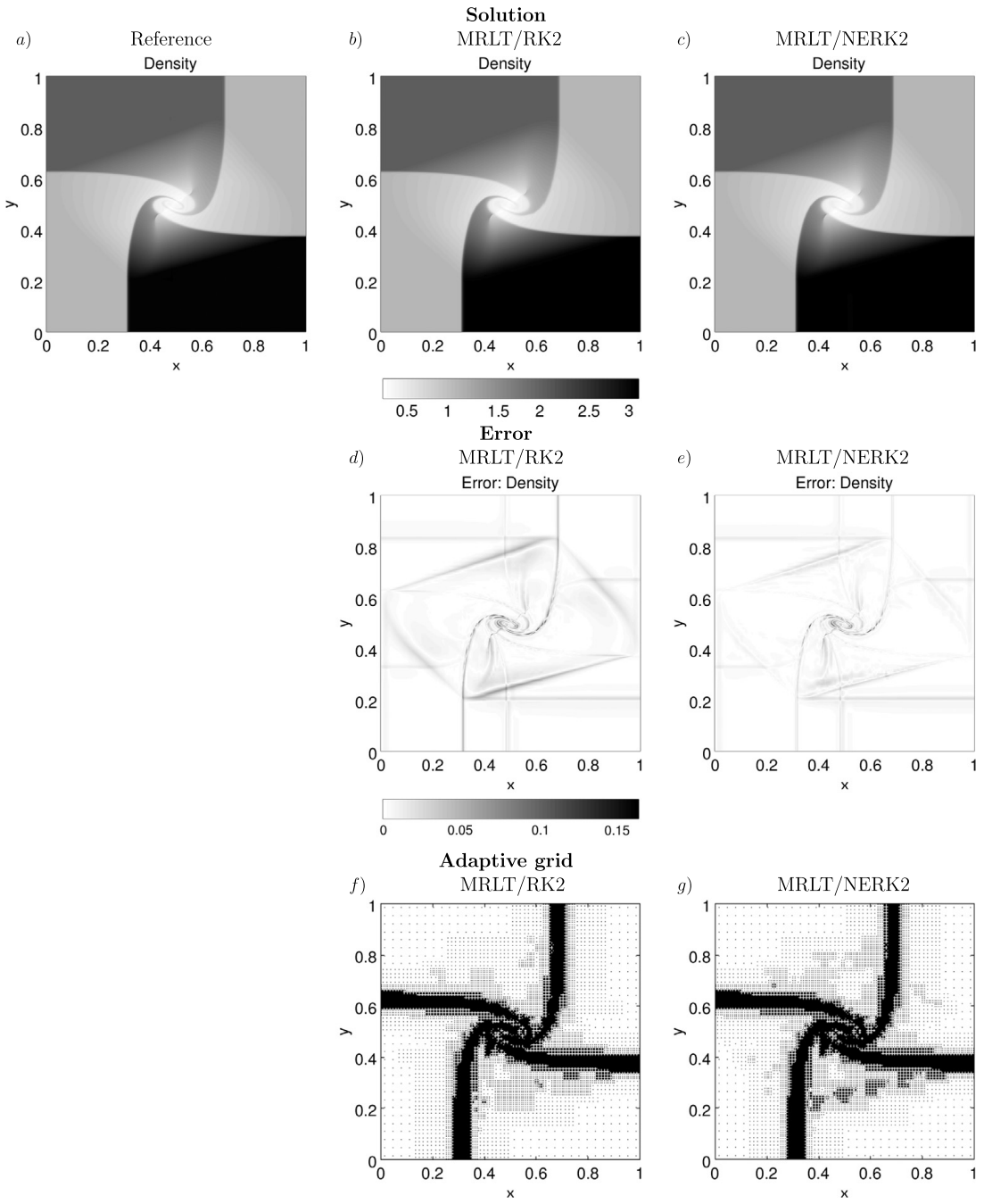


Fig. 10. Reference solution for the two-dimensional Euler equations (a), the solutions obtained by the MRLT/RK2 (b) and MRLT/NERK2 methods, using $L = 10$ scales, with its respective errors (d), (e), and corresponding adaptive grids (f), (g) at final time $t_f = 0.25$.

6. Conclusions

In this work, we introduced a new local time-stepping for adaptive multiresolution methods using NERK time integration schemes [30]. Interpolating values of the intermediate Runge–Kutta stages yield the required values at intermediate time steps, which are necessary for the time evolution. Hence the current limitation of local time stepping to second order schemes can be overcome, and the required synchronised solution can be obtained. The proposed new methodology has been implemented and validated for two and three stage NERK schemes. In principle the extension of MRLT/NERK schemes to even higher order is possible, but the computational cost would increase due to the order barrier discussed in [22]. For NERK methods of order larger or equal to three, Owren and Zennaro [23] have shown that the order of approximation is reduced concerning the underlying RK method. This order reduction implies that increasing the order of NERK schemes beyond three would become less efficient and thus further research is indeed necessary to obtain well performing local time-stepping schemes with order larger than three. With the presented numerical experiments we assessed the precision and efficiency of the proposed two and three stage MRLT/NERK approach for different classical nonlinear evolution equations, *i.e.*, for Burgers, reaction–diffusion and the compressible Euler equations considering Cartesian geometries in one, two

Table 9
Two-dimensional Euler equations: L_1 errors, CPU time and memory compression.

Finest scale level	Method	Error ($\times 10^{-1}$)						CPU time (%FV)	Memory
		ρ	p	T	E	v_x	v_y		
$L=8$	MR/RK2	2.2095	0.6075	0.9581	2.0499	1.7521	0.7180	40.7	26.4
	MRLT/RK2	2.2085	0.6049	0.9572	2.0463	1.7510	0.7167	33.7	26.0
	MRLT/NERK2	2.2096	0.6077	0.9582	2.0503	1.7521	0.7183	12.9	26.6
	MR/RK3	2.2093	0.6075	0.9580	2.0498	1.7520	0.7180	46.3	26.4
	MRLT/NERK3	2.2095	0.6078	0.9581	2.0504	1.7522	0.7182	14.6	26.6
$L=9$	MR/RK2	1.1365	0.3133	0.5217	1.0470	0.9012	0.3934	24.4	14.0
	MRLT/RK2	1.1369	0.3133	0.5219	1.0483	0.9012	0.3931	22.6	13.9
	MRLT/NERK2	1.1355	0.3128	0.5216	1.0458	0.9012	0.3931	7.4	14.2
	MR/RK3	1.1364	0.3133	0.5217	1.0469	0.9012	0.3934	27.3	14.0
	MRLT/NERK3	1.1355	0.3130	0.5216	1.0461	0.9013	0.3931	8.0	14.2
$L=10$	MR/RK2	0.5835	0.1604	0.2765	0.5353	0.4611	0.2115	14.7	7.2
	MRLT/RK2	0.5864	0.1643	0.2782	0.5441	0.4622	0.2131	12.8	7.2
	MRLT/NERK2	0.5821	0.1596	0.2762	0.5335	0.4609	0.2110	4.1	7.3
	MR/RK3	0.5834	0.1604	0.2765	0.5352	0.4611	0.2115	11.2	7.2
	MRLT/NERK3	0.5821	0.1598	0.2762	0.5338	0.4610	0.2109	3.6	7.3

Note: All adaptive computations use $\epsilon = 10^{-2}$; final time: $t_f = 0.25$. Computed on an Intel Core™ *i*7 CPU 2.67 GHz. FV/RK2 CPU time: 10.1 min ($L=8$); 73.3 min ($L=9$); 8.8 h ($L=10$). FV/RK3 CPU time: 11.7 min ($L=8$); 91.7 min ($L=9$); 13.8 h ($L=10$).

Table 10
Two-dimensional Euler equations: Computational gain, for the variable ρ, of the proposed MRLT/NERK methods compared with the MR and MRLT methods.

Finest scale level		MR/RK2	MRLT/RK2	MR/RK3
$L=8$	MRLT/NERK2	3.15	2.61	–
	MRLT/NERK3	–	–	3.17
$L=9$	MRLT/NERK2	3.30	3.05	–
	MRLT/NERK3	–	–	3.41
$L=10$	MRLT/NERK2	3.59	3.14	–
	MRLT/NERK3	–	–	3.11

and three space dimensions. In all adaptive computations, we observed a significant gain in CPU time in comparison with uniform grid computations where the efficiency is increasing with the grid resolution. Nevertheless, the precision of the uniform grid computations is controlled in the MRLT/NERK schemes, and the order of convergence is maintained. Regarding memory consumption, we observed for the MRLT/NERK schemes no necessary increase, compared to MR and classical MRLT methods.

The precision of the MRLT/NERK computations is very reasonable, and in all cases, we found errors about the same order of magnitude as for the MRLT computations using classical RK schemes.

In conclusion, we showed that the MRLT/NERK methods are advantageous compared to the MR and MRLT approaches in all studied cases, obtaining even significant performance gains in some examples. For the two-dimensional Euler equations, for instance, the MRLT/NERK simulations only required one-third of the CPU time necessary for the MR and MRLT computations. Most of these gains are due to the significant reduction in CPU time obtained in the MRLT/NERK methods.

Acknowledgements

The authors are indebted to Prof. Claus-Dieter Munz who motivated the use of NERK methods for local time-stepping. We thank Dr. Olivier Roussel for developing the original Carmen Code and fruitful scientific discussions during the years. ML thankfully acknowledges financial support from CAPES grant 88881.132489/2016-01 and CNPq grant 140626/2014-0, Brazil. MD thankfully acknowledges financial support from Ecole Centrale Marseille, CNPq grants 306038/2015-3, 302226/2018-4 and FAPESP grant 2015/25624-2. OM thankfully acknowledges financial support from CNPq grants 307083/2017-9 and 424352/2018-4, and FINEP grant 01.120527.00, Brazil. KS acknowledges financial support from the ANR Grant 15-CE40-0019 (AIFIT), France.

Appendix A. Algorithms

Algorithm 1 Construction of the adaptive grid.

Do Algorithm 2 {Compute the leaves that will belong to the adaptive grid}
– Refine each leaf with a finer neighbour leaf.
– Set these new leaves as virtual leaves;

Algorithm 2 Grid adaptation.

Require: Finest scale level L of the solution.
Require: Solution at level L.
Require: Threshold value ϵ.
{Selection of the nodes in the adaptive grid}
for $\ell = L-1 \rightarrow 0$, $\ell = \ell-1$ **do**
– Project the solution from the grid $\Omega^{\ell+1}$ to Ω^{ℓ};
– Predict the solution of the grid $\Omega^{\ell+1}$ based on the project solution in Ω^{ℓ} ;
– Compare the original solution of the grid $\Omega^{\ell+1}$ with the predicted one, and then obtain the wavelet coefficients $\tilde{D}^{\ell+1}$, as the difference with these solutions;
{Elimination of the unnecessary nodes and the imposition of a graded tree}
for every leaf $\in \Omega^{\ell+1}$ **do**
if $|\mathbf{d}^{\ell+1}| \leq \epsilon$ and adjacent leaves are inside $\Omega^{\ell+1}$ or Ω^{ℓ} **then**
– Remove the leaf from the grid $\Omega^{\ell+1}$;
end if
end for
end for

Algorithm 3 Single iteration of the LT scheme.

Require: Coarsest scale $\ell_{\min}$ to be evolved in this time evolution;
Require: Current iteration number n;
Compute $\ell_{\min} = \min_{\ell}[\ \mod(n, 2^{L-\ell}) = 0]$;
for $\ell = L \rightarrow \ell_{\min}$, $\ell = \ell-1$ **do**
for Every internal node $\in \Omega^{\ell}$ **do**
Obtain the solution $\tilde{\mathbf{q}}^{n}_{\ell}$ by projecting the solution from its child nodes via simple averaging;
end for
end for
if $\ell_{\min}$ is not the coarsest scale of the grid **then**
for Every internal node $\in \Omega^{\ell_{\min}-1}$ **do**
Obtain the NERK solution with $\theta = \frac{1}{2}$ by projecting $\tilde{\mathbf{q}}^{n}_{\ell_{\min}}$.
Extrapolate this solution to the instant $t^{n+2^{L-\ell_{\min}}}$ (Section 4.3).
end for
for Every virtual leaf $\in \Omega^{\ell_{\min}}$ **do**
Predict $\tilde{\mathbf{q}}^{n}_{\ell}$ using the NERK (RK2) or $\tilde{\mathbf{q}}^{**}_{\ell_{\min}-1}$ (RK3) solutions of the cells $\in \Omega^{\ell_{\min}-1}$ at instant t^{n}.
end for
end if
for $\ell = \ell_{\min}+1 \rightarrow L$, $\ell = \ell+1$ **do**
for Every virtual leaf $\in \Omega^{\ell}$ **do**
Predict $\tilde{\mathbf{q}}^{n}_{\ell}$ using the values of the cells $\in \Omega^{\ell-1}$ at time instant t^{n}.
end for
end for
Remeshing process of cells with refinement level greater or equal than $\ell_{\min}$;
for $\ell = L \rightarrow \ell_{\min}$, $\ell = \ell-1$ **do**
for Every leaf $\in \Omega^{\ell}$ **do**
Perform flux computations at instant t^{n};
First RK step;
First order interpolation of the RK evolution at instant $t^{n} + \frac{1}{2}\Delta t_{\ell}$;
if we are computing RK2 time evolution **then**
Compute $\tilde{\mathbf{q}}^{*}_{\theta=\frac{1}{2}}$;
else if we are computing RK3 time evolution **then**
Compute $\tilde{\mathbf{q}}^{*}_{\theta=\frac{1}{4}}$;
Compute $\tilde{\mathbf{q}}^{*}_{\theta=\frac{3}{4}}$;
end if
end for
end for
Tree refreshing before the second RK step (Algorithm 8):
Second RK step of the time evolution (Algorithm 5);
if Performing RK3 time evolution **then**
Tree refreshing before the third RK step (Section 4.2):
Third RK step of the time evolution (Algorithm 7);
end if

Algorithm 4 Projection procedure inside the second RK step.

Require: Scale ℓ to receive the projection.
Require: Number of dimensions d of the problem.
for Every internal node $\in \Omega^{\ell}$ **do**
  $\bar{\mathbf{q}}^{*}_{\ell} = 0$; {result after RK1, set here to zero to store the projection}
  for Every child cell $\in \Omega^{\ell+1}$ **do**
    if The cell is a leaf **then**
      $\bar{\mathbf{q}}^{*}_{\ell} \leftarrow \bar{\mathbf{q}}^{*}_{\ell} + \bar{\mathbf{q}}_{\ell+1}\left(t^n + 2\Delta t_{\ell+1}\right)$; {Add the already extrapolated value $\bar{\mathbf{q}}_{\ell+1}$ at $t^n + 2\Delta t_{\ell+1}$ }
    else if The cell is an internal node **then**
      $\bar{\mathbf{q}}^{*}_{\ell} \leftarrow \bar{\mathbf{q}}^{*}_{\ell} + 2\bar{\mathbf{q}}^{*}_{\ell+1} - \bar{\mathbf{q}}^{n}_{\ell+1}$; {Add a linear extrapolation of the values $\bar{\mathbf{q}}_{\ell+1}$ at $t^n + 2\Delta t_{\ell+1}$ }
    end if
  end for
  $\bar{\mathbf{q}}^{*}_{\ell} \leftarrow \dfrac{1}{2^d}\bar{\mathbf{q}}^{*}_{\ell}$;
end for

Algorithm 5 Performing the second RK step in the LT approach.

for $\ell = L \rightarrow \ell_{\min}$, $\ell = \ell - 1$ **do**
  if $\ell \neq L$ **then**
    Project the leaves and internal nodes from level $\ell + 1$ onto level ℓ (Algorithm 4);
    Predict the values of the virtual leaves of level $\ell + 1$ at instant $t^n + 2\Delta t_{\ell+1}$;
    {These two steps compute the update in time of the virtual leaves}
  end if
  for every leaf $\in \Omega^{\ell}$ **do**
    Flux computation using the values at instant $t^n + \Delta t_{\ell}$;
    Second step of compact RK;
    Perform the approximation at instant $t^n + 2\Delta t_{\ell}$ given in Equation (14);
    if we are computing RK2 time evolution **then**
      Compute $\bar{\mathbf{q}}_{\theta=\frac{1}{2}}$;
    else if we are computing RK3 time evolution **then**
      Compute $\bar{\mathbf{q}}_{\theta=\frac{1}{4}}$;
      Compute $\bar{\mathbf{q}}_{\theta=\frac{3}{4}}$;
    end if
  end for
end for

Algorithm 6 Projection procedure inside the third RK step.

Require: Scale ℓ to receive the projection.
Require: Number of dimensions d of the problem.
for every internal node $\in \Omega^{\ell}$ **do**
  $\bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right) = 0$; {result after RK2, set here to zero to store the projection}

  for Every child cell $\in \Omega^{\ell+1}$ **do**
    $\bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right) \leftarrow \bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right) + \bar{\mathbf{q}}_{\ell+1}\left(t^n + \Delta t_{\ell+1}\right)$; {Add value $\bar{\mathbf{q}}_{\ell+1}$ from RK3 3^{rd} step }
  end for

  $\bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right) \leftarrow \dfrac{1}{2^d}\bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right)$
  $\bar{\mathbf{q}}^{**}_{\ell} \approx \bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right)$;

  To obtain a 2^{nd} order approximation for $\bar{\mathbf{q}}_{\ell}(t^n + \Delta t_{\ell})$, use Equation (16).
end for

Algorithm 7 Performing the third RK step in the LT approach.

for $\ell = L \rightarrow \ell_{\min}$, $\ell = \ell - 1$ **do**
  if $\ell \neq L$ **then**
    – Project the leaves and internal nodes from level $\ell + 1$ onto level ℓ (Algorithm 6);
    – Predict the values of the virtual leaves of level $\ell + 1$ at instant $t^n + \Delta t_{\ell+1}$;
    {These two steps compute the update in time of the virtual leaves}
  end if
  for Every leaf $\in \Omega^{\ell}$ **do**
    – Flux computation using the values at instant $t^n + \frac{1}{2}\Delta t_{\ell}$;
    – Third step of compact RK;
  end for
end for

Algorithm 8 Tree refreshing before the second RK step.

Require: Scale ℓ to receive the projection.
Require: Number of dimensions d of the problem.
for $\ell = L-1 \rightarrow \ell_{\min}$, $\ell = \ell - 1$ **do**
  for Every internal node $\in \Omega^{\ell}$ **do**
    $\bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right) = 0$; {Set the solution equal zero to perform the averaging of its children cells.}
    for Every child cell $i \in \Omega^{\ell+1}$ **do**
      $\bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right) \leftarrow \bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right) + \bar{\mathbf{q}}^{*}_{\ell+1,\,i}$;
    end for
    $\bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right) \leftarrow \frac{1}{2^d}\bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right)$
    Obtain a 1st order approximation for $\bar{\mathbf{q}}^{*}_{\ell}$ using Equation (18).
  end for
end for
for $\ell = \ell_{\min} \rightarrow L$, $\ell = \ell + 1$ **do**
  for Every virtual leaf $\in \Omega^{\ell}$ **do**
    Use the approximated solution at level $\ell - 1$ at time instant $t^n + \frac{1}{2}\Delta t_{\ell-1}$ to predict the solution $\bar{\mathbf{q}}^{*}_{\ell}$.
    Obtain the solution $\bar{\mathbf{q}}_{\ell}\left(t^n + \frac{1}{2}\Delta t_{\ell}\right)$ by linear interpolation. {These two steps predict the solution of the virtual leaves in the proper time instant to update the level ℓ.}
  end for
end for

Algorithm 9 Tree refreshing before the third RK step.

Require: Scale ℓ to receive the projection.
Require: Number of dimensions d of the problem.
for $\ell = L-1 \rightarrow \ell_{\min}$, $\ell = \ell - 1$ **do**
  for Every internal node $\in \Omega^{\ell}$ **do**
    $\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}} = 0$;
    for Every child cell $i \in \Omega^{\ell+1}$ **do**
      $\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}} \leftarrow \bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}} + \bar{\mathbf{q}}^{**}_{\ell+1,\,i}$;
    end for
    $\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}} \leftarrow \frac{1}{2^d}\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}}$
    Compute $\bar{\mathbf{q}}_{\ell,\,\theta=\frac{3}{4}}$ using Equation (20).
    Compute $\bar{\mathbf{q}}^{**}_{\ell}$ using Equation (21).
  end for
end for
for $\ell = \ell_{\min} \rightarrow L$, $\ell = \ell + 1$ **do**
  for Every virtual leaf $\in \Omega^{\ell}$ **do**
    if $\ell = \ell_{\min}$ **then**
      Use the solution $\bar{\mathbf{q}}_{\ell-1,\,\theta=\frac{3}{4}}$, at time instant $t^n + \frac{1}{2}\Delta t_{\ell}$, to predict the solution $\bar{\mathbf{q}}^{**}_{\ell}$.
    else
      Use the solution $\bar{\mathbf{q}}_{\ell-1,\,\theta=\frac{1}{4}}$, at time instant $t^n + \frac{1}{2}\Delta t_{\ell}$, to predict the solution $\bar{\mathbf{q}}^{**}_{\ell}$.
    end if
    Compute $\bar{\mathbf{q}}_{\ell,\,\theta=\frac{1}{4}}$ using Equation (21).
    Compute $\bar{\mathbf{q}}_{\ell,\,\theta=\frac{3}{4}}$ using Equation (20).
  end for
end for

References

[1] W. Auzinger, O. Koch, An improved local error estimator for symmetric time-stepping schemes, Appl. Math. Lett. 82 (2018) 106–110.
[2] W. Boscheri, M. Dumbser, O. Zanotti, High order cell-centered Lagrangian-type finite volume schemes with time-accurate local time stepping on unstructured triangular meshes, J. Comput. Phys. 291 (2015) 120–150.
[3] F. Coquel, L.Q. Nguyen, M. Postel, Q.H. Tran, Local time stepping applied to implicit–explicit methods for hyperbolic systems, SIAM Multiscale Model. Simul. 8 (2) (2010) 540–570.
[4] J. Díaz, M.J. Grote, Conserving explicit local time-stepping for second-order wave equations, SIAM J. Sci. Comput. 31 (3) (2009) 1985–2014.
[5] J. Díaz, M.J. Grote, Multi-level explicit local time-stepping methods for second-order wave equations, Comput. Methods Appl. Mech. Eng. 291 (2015) 240–265.
[6] M.O. Domingues, S.M. Gomes, O. Roussel, K. Schneider, An adaptive multiresolution scheme with local time stepping for evolutionary PDEs, J. Comput. Phys. 227 (8) (2008) 3758–3780.
[7] M.O. Domingues, S.M. Gomes, O. Roussel, K. Schneider, Space–time adaptive multiresolution techniques for compressible Euler equations, in: C.A. de Moura, C.S. Kubrusly (Eds.), The Courant–Friedrichs–Lewy (CFL) Condition: 80 Years After Its Discovery, Birkhäuser, 2012, pp. 101–117.
[8] M. Dumbser, O. Zanotti, A. Hidalgo, D.S. Balsara, ADER–WENO finite volume schemes with space–time adaptive mesh refinement, J. Comput. Phys. 248 (2013) 257–286.
[9] G. Gassner, M. Dumbser, F. Hindenlang, C.-D. Munz, Explicit one-step time discretizations for discontinuous Galerkin and finite volume schemes based on local predictors, J. Comput. Phys. 230 (11) (2011) 4232–4247, Special issue High Order Methods for CFD Problems.
[10] G.J. Gassner, F. Hindenlang, C.-D. Munz, A Runge–Kutta based discontinuous Galerkin method with time accurate local time stepping, Adv. Comput. Fluid Dyn. 2 (2011) 95–118.

[11] N.Y. Gnedin, V.A. Semenov, A.V. Kravtsov, Enforcing the Courant–Friedrichs–Lewy condition in explicitly conservative local time stepping schemes, J. Comput. Phys. 359 (2018) 93–105.
[12] A. Harten, Multiresolution algorithms for the numerical solution of hyperbolic conservation laws, Commun. Pure Appl. Math. 48 (1995) 1305–1342.
[13] B. Hejazialhosseini, D. Rossinelli, M. Bergdorf, P. Koumoutsakos, High order finite volume methods on wavelet-adapted grids with local time-stepping on multicore architectures for the simulation of shock-bubble interactions, J. Comput. Phys. 229 (22) (2010) 8364–8383.
[14] N. Hovhannisyan, S. Müller, On the stability of fully adaptive multiscale schemes for conservation laws using approximate flux and source reconstruction strategies, IMA J. Numer. Anal. 30 (4) (2010) 1256–1295.
[15] L. Krivodonova, An efficient local time-stepping scheme for solution of nonlinear conservation laws, J. Comput. Phys. 229 (2010) 8537–8551.
[16] P.D. Lax, X-D. Liu, Solution of two-dimensional Riemann problems of gas dynamics by positive schemes, SIAM J. Sci. Comput. 19 (2) (1998) 319–340.
[17] M.-S. Liou, A sequel to ausm: ausm+, J. Comput. Phys. 129 (1996) 364–382.
[18] M. Mayr, W.A. Wall, M.W. Gee, Adaptive time stepping for fluid–structure interaction solvers, Finite Elem. Anal. Des. 141 (2018) 55–69.
[19] S. Müller, Adaptive Multiscale Schemes for Conservation Laws, Lecture Notes in Computational Science and Engineering, vol. 27, Springer, Heidelberg, 2003.
[20] S. Müller, Y. Stiriba, Fully adaptive multiscale schemes for conservation laws employing locally varying time stepping, J. Sci. Comput. 30 (3) (2007) 493–531.
[21] S. Osher, R. Sanders, Numerical approximations to nonlinear conservation laws with locally varying time and space grids, Math. Comput. 41 (1983) 321–336.
[22] B. Owren, M. Zennaro, Order barriers for continuous explicit Runge–Kutta methods, Math. Comput. 56 (1991) 645–661.
[23] B. Owren, M. Zennaro, Derivation of efficient, continuous, explicit Runge–Kutta methods, SIAM J. Sci. Statist. Comput. 13 (6) (1992) 1488–1501.
[24] H. Qi, X. Wang, J. Zhang, J. Wang, An ADER discontinuous Galerkin method with local time-stepping for transient electromagnetics, Comput. Phys. Commun. 229 (2018) 106–115.
[25] M. Rietmann, M. Grote, D. Peter, O. Schenk, Newmark local time stepping on high-performance computing architectures, J. Comput. Phys. 334 (2017) 308–326.
[26] O. Roussel, K. Schneider, A. Tsigulin, H. Bockhorn, A conservative fully adaptive multiresolution algorithm for parabolic PDEs, J. Comput. Phys. 188 (2003) 493–523.
[27] B. Van Leer, Towards the ultimate conservative difference scheme, iii: upstream-centered finite-difference schemes for ideal compressible flow, J. Comput. Phys. 23 (3) (1977) 263–275.
[28] R. Vermiglio, M. Zennaro, Multistep natural continuous extensions of Runge–Kutta methods: the potential for stable interpolation, Appl. Numer. Math. 12 (6) (1993) 521–546.
[29] A.R. Winters, D.A. Kopriva, High-order local time stepping on moving DG spectral element meshes, J. Sci. Comput. 58 (1) (2014) 176–202.
[30] M. Zennaro, Natural continuous extensions of Runge–Kutta methods, Math. Comput. 46 (173) (1986) 119–133.

ANNEX C - AMROC MHD SOLVER DISCUSSIONS

In the following is annexed an article published in the Computers and Fluids that presents the AMROC MHD solver and discuss about this performance for adaptive and parallel simulations for both bidimensional and tridimensional problems. It is referred as (MOREIRA LOPES et al., 2018a), namely:

- MOREIRA LOPES, M.; DEITERDING, R.; GOMES, A.; MENDES, O.; DOMINGUES, M. An ideal compressible magnetohydrodynamic solver with parallel block-structured adaptive mesh refinement. **Computers and Fluids**, v. 173, p. 293-298, 2018.

Computers and Fluids 173 (2018) 293–298

Contents lists available at ScienceDirect

Computers and Fluids

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

An ideal compressible magnetohydrodynamic solver with parallel block-structured adaptive mesh refinement

Muller Moreira Lopes[a], Ralf Deiterding[b,*], Anna Karina Fontes Gomes[a], Odim Mendes[a], Margarete O. Domingues[a]

[a] National Institute for Space Research (INPE), São José dos Campos, São Paulo, 12.227-010, Brazil
[b] Aerodynamics and Flight Mechanics Research Group, University of Southampton, SO17 1BJ, United Kingdom

ARTICLE INFO

Article history:
Received 31 October 2017
Accepted 23 January 2018
Available online 1 February 2018

Keywords:
AMROC
Magnetohydrodynamics
Finite-volume
Mesh refinement

ABSTRACT

We present an adaptive parallel solver for the numerical simulation of ideal magnetohydrodynamics in two and three space dimensions. The discretisation uses a finite volume scheme based on a Cartesian mesh and an explicit compact Runge–Kutta scheme for time integration. Numerically, a generalized Lagrangian multiplier approach with a mixed hyperbolic-parabolic correction is used to guarantee a control on the incompressibility of the magnetic field. We implement the solver in the AMROC (Adaptive Mesh Refinement in Object-oriented C++) framework that uses a structured adaptive mesh refinement (SAMR) method discretisation-independent and is fully parallelised for distributed memory systems. Moreover, AMROC is a modular framework providing manageability, extensibility and efficiency. In this paper, we give an overview of the ideal magnetohydrodynamics solver developed in this framework and its capabilities. We also include an example of this solver's verification with other codes and its numerical and computational performance.

1. Introduction

The numerical simulation of plasma plays an important role in astrophysics and space physics due to its scale variabilities [1]. For studying macroscopic space plasma phenomena, the ideal magnetohydrodynamics (MHD) theory is a handy tool that treats the plasma as a perfect electrically conducting fluid under the influence of a magnetic field [2]. Since the last decades, a variety of numerical algorithms for multidimensional MHD based on finite volume method have been developed, for instances the ones discussed in [3].

These algorithm have two significant extensions compared to the primary hydrodynamical case: the first is an extension of the Riemann solver used to compute the fluxes of each conserved quantity, and the second is the method that assures a control of the divergence-free constraint, *i.e.*, $\nabla \cdot \mathbf{B} = 0$. In general, realistic MHD simulations in the context of space weather forecast are performed under prohibitive computational costs, which remains a core challenge to be overcome. Therefore, in practice for multiscale space MHD applications very efficient frameworks are essential. As proposed here, a new MHD solver has been developed upon our serial Carmen–MHD code, cf. [4,5] and references therein. The current solver is an ideal compressible MHD model with finite volume adaptive mesh in two and three space dimensions developed in the parallel AMROC framework, incorporating recent advances in the area of SAMR.

We organise the content as follows: in Section 2, we describe the governing equations, the finite volume method, and briefly the main ideas of the block-based adaptive mesh refinement (AMR) approach as implemented in AMROC. In Section 3, we develop the numerical experiments. Then, we draw the conclusions in Section 4.

2. Numerical methods

2.1. Governing equations

In the solution of hyperbolic conservation laws given by partial differential equations such as $\partial_t \mathbf{q} + \nabla \cdot \mathbf{f}(\mathbf{q}) = 0$, different length scales are ubiquitous. It is well established that for nonlinear flux functions $\mathbf{f}(\mathbf{q})$ even continuous initial data can develop into discontinuities over time [6]. In particular, we consider the compressible inviscid ideal magnetohydrodynamic equations, in Cartesian coordinates [2], that describe the physics of an ideal conducting fluid under the influence of a magnetic field. Basically this system is

* Corresponding author.
E-mail addresses: muller.lopes@inpe.br (M.M. Lopes), r.deiterding@soton.ac.uk (R. Deiterding), anna.gomes@inpe.br (A.K.F. Gomes), odim.mendes@inpe.br (O. Mendes), margarete.domingues@inpe.br (M.O. Domingues).

https://doi.org/10.1016/j.compfluid.2018.01.032
0045-7930/

composed of the continuity, energy, momentum equations, and induction equations as

$$\mathbf{q} = \begin{pmatrix} \rho \\ E \\ \rho\mathbf{u} \\ \mathbf{B} \end{pmatrix}, \quad \mathbf{f}(\mathbf{q}) = \begin{pmatrix} \rho\mathbf{u} \\ \left(E + p + \dfrac{\mathbf{B}\cdot\mathbf{B}}{2}\right)\mathbf{u} - \mathbf{B}(\mathbf{u}\cdot\mathbf{B}) \\ \rho\mathbf{u}\mathbf{u} + \left(p + \dfrac{\mathbf{B}\cdot\mathbf{B}}{2}\right)\mathbf{I} - \mathbf{B}\mathbf{B} \\ \mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u} \end{pmatrix}. \tag{1}$$

In the latter, ρ is the density, $\mathbf{u}$ the fluid velocity vector, $\mathbf{B}$ the magnetic field vector, and E the total energy density defined as $E = \dfrac{p}{\gamma - 1} + \dfrac{\rho u^2}{2} + \dfrac{B^2}{2}$, where p is the gas pressure considering the state equation assuming an ideal gas plasma. All variables depend on the spatial location $\mathbf{x}$ and time t and these equations are represented in non-dimensional form such that the magnetic permeability μ is normalised to one.

Due to Gauss' law of magnetism, this system has also a physical constraint in the magnetic field, given by $\nabla\cdot\mathbf{B} = 0$. In numerical simulations the divergence constraint is not always satisfied as first realised by Brackbill and Barnes [7]. In order to minimize this effect, several types of corrections have been used, as described in [3] and references therein. Here, we have chosen a divergence transport approach known as parabolic-hyperbolic divergence cleaning proposed by Dedner et al. in [8] in combination with the correction proposed by Mignone and Tzeferacos in [9]. This correction consists in adding a differential operator $D = \dfrac{1}{c_h^2}\dfrac{\partial}{\partial t} + \dfrac{1}{c_p^2}$ to the divergence constraint of the $\mathbf{B}$ field, resulting in a new system composed basically of the continuity, energy, momentum equations, and the additional equations

$$\frac{\partial\mathbf{B}}{\partial t} + \nabla\cdot(\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u} + \psi\mathbf{I}) = 0, \quad \frac{\partial\psi}{\partial t} + c_h^2\nabla\cdot\mathbf{B} = -\frac{c_h^2}{c_p^2}\psi \tag{2}$$

where ψ is a scalar-valued function, $\mathbf{I}$ the identity tensor, and c_p and c_h are the parabolic and hyperbolic constants, respectively. The constant c_h is defined as

$$c_h = \max\left[\nu\frac{\Delta h}{\Delta t}, \max\left(|u_i| \pm c_f\right)\right], \tag{3}$$

where $\Delta h = \min(\Delta x, \Delta y, \Delta z)$, Δx, Δy, Δz are the mesh sizes in each direction, ν the Courant number, u_i is the velocity of the ith component, and c_f is the fast magnetoacoustic wave of the MHD model. The c_p value is defined in terms of the parameter $\alpha_p = \Delta h\dfrac{c_h}{c_p^2}$, where $\alpha_p \in [0, 1]$, as described in [9]. We also have γ denoting the specific heat ratio. This system is well known as the Generalized Lagrange Multiplier (GLM) approach, and it reduces to the usual MHD system when $\psi = 0$.

In practical MHD cases, discontinuous shock and contact waves can develop, which requires the use of shock-capturing methods. In particular, the finite volumes method have been constructed to handle particularly this behaviour in a robust and oscillation free way [10]. Since in inviscid problems discontinuities and contact waves are usually very localised, a local increase of mesh resolution is beneficial to represent these jumps as accurately as possible.

In the computations presented here, we use a finite volume scheme based on a Cartesian mesh with HLLD numerical flux introduced by Miyoshi and Kusano [11] with monotonised central (MC) symmetric limiter discussed in [12] and an explicit second order Runge–Kutta scheme for time integration. We apply the divergence control parameter $\alpha_p = 0.4$ and $\psi \equiv 0$ in the initial conditions.

2.2. Block-structured AMR and AMROC framework

The structured adaptive mesh refinement method for finite volume methods introduced by Berger and Collela [13] follows a patch-oriented refinement approach, where non-overlapping rectangular sub-meshes $G_{\ell,m}$ define by $G_\ell := \bigcup_{m=1}^{M_\ell} G_{\ell,m}$ the domain of an entire level with index $\ell = 0, \cdots, L$.

As the construction of refinement proceeds recursively, a hierarchy of sub-meshes successively contained within the next coarser level domain is created. This method has become an important mesh adaptation approach for hyperbolic conservation laws.

The characteristic of the SAMR algorithm is that refinement patches overlay coarser mesh data structures, instead of being embedded, thereby avoiding data fragmentation. Values of cells covered by finer sub-meshes are subsequently overwritten by averaged fine mesh values, which, in general, would lead to a loss of conservation on the coarser mesh. A remedy to this problem is to replace the coarse mesh numerical fluxes at refinement boundaries with the sum of fine mesh fluxes along the corresponding coarse cell boundary, cf. [13,14].

The recursive nature of the algorithm allows only the addition of one new level in each refinement operation. The patch-based approach does not require special coarsening operations; sub-meshes are simply removed from the hierarchy. The coarsest possible resolution is thereby restricted to the level zero mesh. It is assumed that all mesh widths on level ℓ are r_ℓ-times finer than on the level $\ell - 1$, which ensures that a time-explicit finite volume scheme remains stable under a CFL-type condition on all levels of the hierarchy. The numerical update is applied on the level ℓ by calling a single-mesh routine implementing the uniform scheme in a loop over all the sub-meshes. The regularity of the input data allows a straightforward implementation of the scheme and further permits optimisation to take advantage of high-level caches, and pipelining, for instance. New refinement meshes are initialised by interpolating the vector of conservative quantities $\mathbf{Q}$ from the next coarser level. However, data in cells already refined is copied directly from the previous refinement patches. *Ghost cells* around each patch are used to decouple the sub-meshes computationally. Ghost cells outside of the root domain G_0 are used to implement physical boundary conditions. Ghost cells in G_ℓ have a unique interior cell analogue and are set by copying the data value from the patch where the interior cell is contained (synchronisation). For $\ell > 0$, internal boundaries can also be used. If recursive time step refinement is employed, ghost cells at the internal refinement boundaries on the level ℓ are set by time-space interpolation from the two previously calculated time steps of level $\ell - 1$. Otherwise, spatial interpolation from the level $\ell - 1$ is sufficient.

AMROC implements the SAMR method discretisation-independent in one to three space dimensions and is fully parallelised for distributed memory systems [14,15]. With its parallel distribution strategy the overlapping ghost cell regions of neighbouring patch blocks are synchronised over processor borders as boundary conditions are applied. The communication between processors is achieved through the MPI-library and a space filling curve algorithm is used for load-balanced data distribution as the refinement mesh is evolving during the course of a simulation. The parallel distribution is accomplished as a rigorous mesh decomposition in entities of the base level mesh, but note that the workload of all higher refinement levels including time step refinement is considered [16].

Adaptation along discontinuities can be achieved by evaluating gradients multiplied by the step size in all directions (*scaled gradient*). Cell(j, k) is flagged for refinement if at least one of the three relations

Table 1
Kelvin–Helmholtz instability: initial condition for u_x and u_y.

u_x	$5[\tanh(20y+10) - (\tanh(20y-10)+1)]$
u_y	$\frac{1}{4}\sin(2\pi x)\left(e^{-100(y+\frac{1}{2})^2} - e^{-100(y-\frac{1}{2})^2}\right)$

$$|w(\mathbf{Q}_{j+1,k}) - w(\mathbf{Q}_{j,k})| > \epsilon^w,\ |w(\mathbf{Q}_{j,k+1}) - w(\mathbf{Q}_{j,k})| > \epsilon^w,$$
$$|w(\mathbf{Q}_{j+1,k+1}) - w(\mathbf{Q}_{j,k})| > \epsilon^w$$

is satisfied for an arbitrary scalar quantity w, which is derived from the numerical vector of state $\mathbf{Q}^\ell(t)$ on level ℓ. The constant ϵ^w denotes the prescribed refinement threshold.

Central to the block-structured mesh refinement approach is the utilisation of a dedicated *cluster* algorithm to create blocks from individual cells tagged for refinement. We use a recursive algorithm proposed by Bell et al. [17] for this purpose. The algorithm starts with the steepest zero crossing and uses recursively weaker ones, until the ratio between flagged and all cells in every new mesh is above the prescribed value $0 < \eta \leq 1$. In practice, we use $\eta = 0.7$ on the present computations. A buffer zone of one cell is added around tagged cells to avoid degradation of results from interpolation. More details about these strategies can be found in [14].

3. Results

3.1. Kelvin–Helmholtz instability

In the context of space sciences, the Kelvin–Helmholtz (KH) instability appears in many phenomena such as the solar corona, the ionosphere and astrophysical objects. This instability is a phenomenon which occurs in single continuous fluids with a velocity shear layer or at the interface of two fluids with different velocities [18]. In this test, in order to study KH symmetry, two velocity shear layers in opposite directions are inserted at the lines $y = 0.5$ and $y = -0.5$. Along with these shear layers, a perturbation in the u_y component is inserted in order to create a vortex. The perturbations in u_y around $y = 0.5$ and $y = -0.5$ are in opposite direction, hence the vortices will be counter-rotating. The initial configuration is described in [8] by the values $\rho = 1,\ p = 50,\ B_x = 1,\ u_z = B_y = B_z = 0$. The settings for u_x and u_y are given in Table 1.

The computational domain is $[0, 1] \times [-1, 1]$ with periodic boundary conditions and the computations reach time $t_{\text{end}} = 0.5$. The simulation parameters are $\nu = 0.4,\ \gamma = 1.4$. As refinement criteria for the SAMR algorithm, we used the scaled gradient threshold $\epsilon^\rho = 0.01$, as applied to the density field only. Note that in these tests the finest level is always at a resolution equivalent to a uniform mesh with 2048^2 cells but the number of levels used is varied between one and three, using a refinement factor 2 on all levels.

The numerical solution at t_{end} obtained for the variable pressure in the case of three refinement levels running on eight processors is given in Fig. 1 (left). This picture uses ten contour curves, from minimum value to maximum pressure value. Super-imposed are the refinement levels of the mesh, where red is the most refined and blue is the coarsest level. We observe that the refinement zones agree with the isolines and the symmetry of the instability is almost preserved. Fig. 1 (right) shows the distribution of the cells updated by each processor related to this solution.

The $\mathbb{L}^1$ errors for the physical variables ρ, p, and the maximum of the errors among all the components of the magnetic field B_i and velocity field u_i are presented in Table 2. The errors double between the refinement levels for all variables, which is expected as the computations with higher refinement use consecutively coarser cells on the base mesh. The errors in the density are the smallest, whilst pressure has the largest ones, as expected.

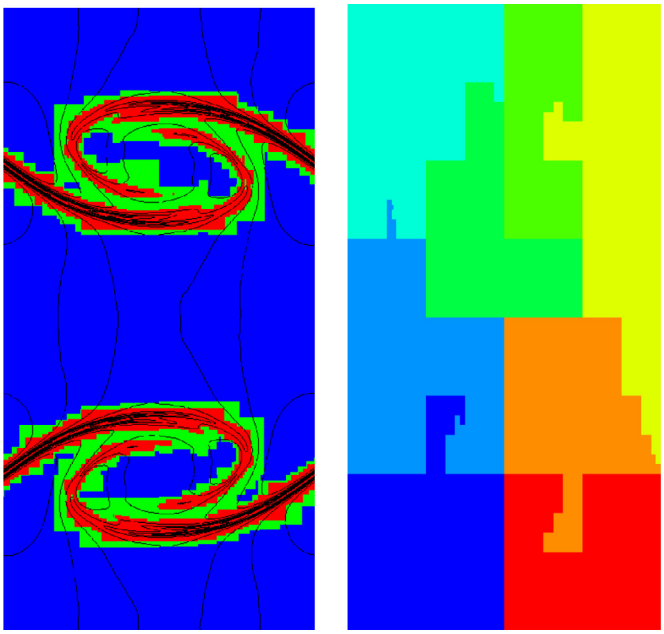


Fig. 1. Kelvin–Helmholtz instability: contour plot of pressure with background colour according to refinement level (left). Mesh distribution in each processor at t_{end} (right).

Table 2
Kelvin–Helmholtz instability: $\mathbb{L}^1$ errors considering uniform mesh of 2048^2 cells using two processors.

Levels	Variables			
	ρ	p	$\max(B_i)$	$\max(u_i)$
2	0.01	0.65	0.12	0.13
3	0.02	1.49	0.25	0.29

Table 3
Kelvin–Helmholtz instability: elapsed time, in minutes, of the computations using one to three refinement levels as a function of the number of processors. Simulations were performed using two nodes in a cluster with processors Intel Xeon 2.20 GHz with 20 cores each, dividing the processes equally among the two nodes.

		Processors				
		1	2	4	8	16
Levels	1	5790	3214	1597	825	573
	2	1187	627	300	165	113
	3	411	235	129	82	66

To verify the scalability of the parallel algorithm, we consider simulations with one to three refinement levels, where the most refined mesh is 2048^2 cells for every number of levels, using 2^n processors, with $n = 0$ to 4. The computational time in minutes of those experiments are given in Table 3. As expected, considering the uniform mesh there is a time decay of roughly $\frac{1}{2}$ between 2^n and 2^{n+1} processors. Similar results can be observed for varying refinement levels.

3.2. Orszag and Tang vortex

This test problem has been introduced by Orszag and Tang [19]. Since then, it has been extensively used in verification tests of ideal MHD simulations, for instance, in [20–22]. Due to its physical characteristics it is also a well-known model for 2D turbulence, as described in [23] using Fourier spectral methods. In detail, it verifies

Table 4
Orszag–Tang vortex: initial condition for velocity field.

	2D	3D
u_x	$-\sin(y)$	$-[1+\varepsilon_p \sin(z)]\sin(y)$
u_y	$\sin(x)$	$[1+\varepsilon_p \sin(z)]\sin(x)$
u_z	0	$\varepsilon_p \sin(z)$

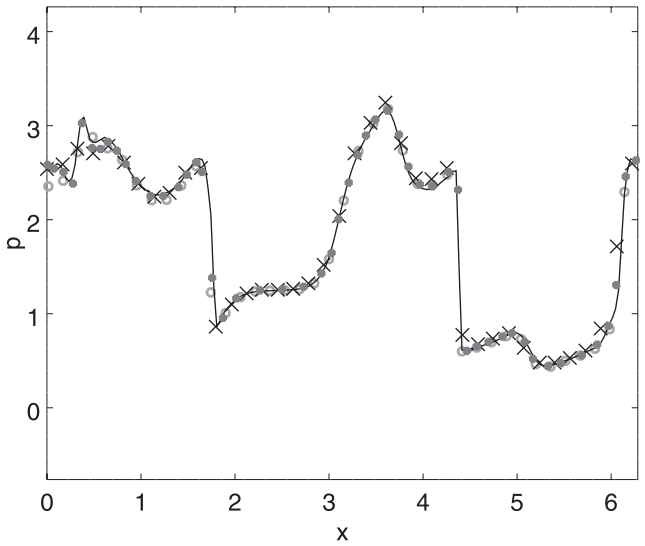


Fig. 2. 1D cut comparison of pressure solutions at $y = 0.64\pi$ at t_{end} obtained from our simulations (line) with other simulations: Londrilllo–Del Zanna (cross), Miyoshi–Kusano (dot), and FLASH eight-wave (circle).

the transitions in MHD structures, and consequently how robust the code is at handling the formation of MHD shocks, shock-shock interactions, and moreover, it helps in the identification of how significant magnetic monopoles affect the numerical solutions.

The initial conditions are for both cases $\rho = \gamma^2$, $p = \gamma$, $B_x = -\sin(y)$, $B_y = \sin(2x)$, and $B_z = 0$. The values for the velocity components are given in Table 4. Note that in the 3D case a perturbation parameter $\varepsilon_p = 0.2$ is included in addition into all components, following the example presented in [24].

The computational domain is $[0, 2\pi]$ in every direction with periodic boundary conditions until the final time $t_{\text{end}} = \pi$. These simulations are done using $\gamma = \frac{5}{3}$, and $\nu = 0.4$, and 0.3, with scaled gradient threshold applied to the density field of $\epsilon^{\rho} = 0.1$ and 0.5 for 2D, and 3D cases, respectively.

To visualize the local convergence of the Orszag–Tang vortex, we present a cut of the pressure field at $y = 0.64\pi$ and compare it to the results obtained in the literature (Fig. 2). All experiments were run with a 200^2 mesh, except the Miyoshi–Kusano one, which used a mesh size of 192^2. The FLASH solutions are computed using the HLLD flux and the eight-wave divergence cleaning scheme. The Londrilllo–Del Zanna result uses the third order LF–CENO scheme and staggered collocation for the magnetic field [22]. The Miyoshi–Kusano results used the HLLD flux and the mixed GLM divergence cleaning [11] as we do. Our results are obviously in good agreement with respect to the main structures of the solution.

For the parallel tests, our simulations are performed with one to three refinement levels, where the most refined mesh is 2048^2 (2D) and 128^3 (3D). In Fig. 3, we present similar graphs as in Fig. 1 for this 2D case. As desired, the refinement agrees with the structures present in the solutions, and there is a uniform distribution among the processors considering the refinement regions. The errors, using a $\mathbb{L}^1$ norm in relation with the solution in a uniform mesh are presented in Table 5. In this case, density and pressure present similar errors and the largest values are for two and three refinement levels. Again, for three decomposition levels the error roughly doubles in relation to the second refinement level. In Fig. 4, the pressure solution and the mesh distribution among the processors are presented at t_{end}. The solution symmetries are almost preserved, as desired, and the processor distributions follow well-balanced patterns observed already in the 2D cases.

The computational time in minutes for 2^n processors with $n = 0$ to 4 of those experiments are given in Table 6. In the 2D experiments after four processors the scalability is close to two. In 3D, all experiments present scalability near two, except for 16 processors, where probably the communication costs dominate due to the small problem size.

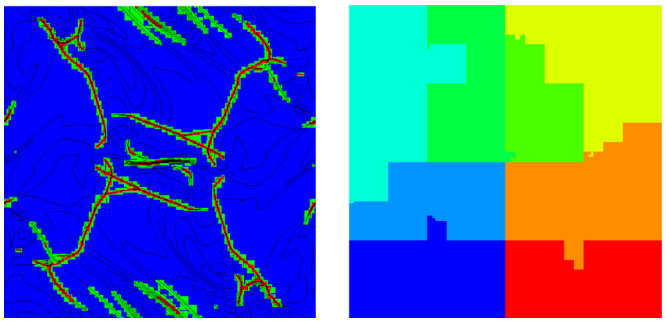


Fig. 3. 2D Orszag–Tang vortex: contour plot of pressure with background colour according to refinement level (left) and mesh distribution in each processor at the final iteration (right).

Table 5
Orszag–Tang vortex: $\mathbb{L}^1$ errors for 2D cases.

	Levels	Variables			
		ρ	p	$\max(B_i)$	$\max(u_i)$
2D	2	0.62	0.64	0.32	0.23
	3	1.24	1.39	0.68	0.48

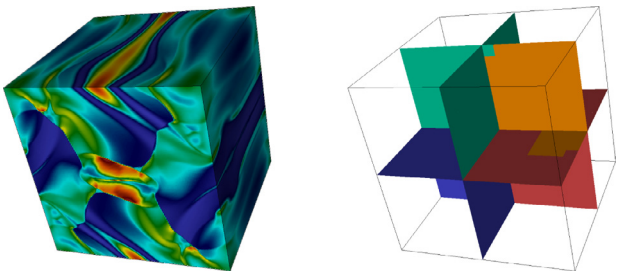


Fig. 4. 3D Orszag–Tang vortex: solution for pressure (left) and mesh distribution in each processor (right) at t_{end}.

Table 6
Orszag–Tang vortex: elapsed time of the computations using one to three refinement levels as a function of the number of processors. Simulations were performed using two nodes in a cluster with processors Intel Xeon 2.20 GHz with 20 cores each. Here, we divided the processes equally among the two nodes.

	Level	Processors				
		1	2	4	8	16
2D	1	1174	632	306	160	98
	2	409	240	136	77	41
	3	351	176	89	49	29
3D	1	99	69	28	17	11
	2	68	41	21	12	8
	3	69	50	20	12	8

Table 7
3D Shock-Cloud interaction: $\mathbb{L}^1$ errors.

Levels	Variables			
	ρ	p	$\max(B_i)$	$\max(u_i)$
2	0.04	1.47	0.07	0.08
3	0.13	6.24	0.20	0.40

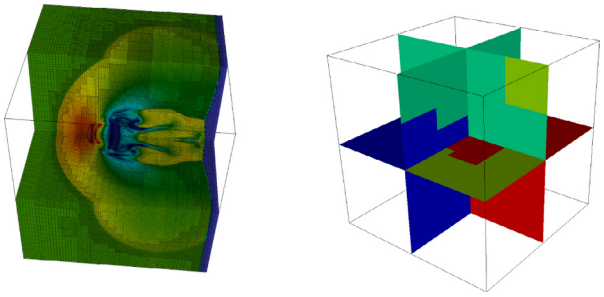


Fig. 5. 3D Shock-Cloud interaction: pressure plotted over the adapted mesh (left), and mesh distribution in each processor (right) at t_{end}.

3.3. Shock-Cloud iteration

To check the performance of the numerical scheme when dealing with super-fast flows, this problem presents a disruption of a high-density magnetic cloud by a strong shock wave, as described in [1]. The initial condition is constructed by considering two regions, the first one defines the advancing plasma – which causes the shock – and the other is a stationary state where the shock advances. These regions are limited by the domain boundaries and a plane parallel to the yz plane at $x = 0.05$. Inside the second region, we define the cloud as a high density region in hydrostatic equilibrium with the surrounding plasma.

We consider the cloud region as a sphere with centre at (0.25, 0.5, 0.5) and radius $r_0 = 0.15$. The advancing plasma initial condition is given by $\rho = 3.86859$, $p = 167.345$, $v_x = 11.2536$, $v_y = v_z = B_x = 0$, $B_y = 2.1826182$ and $B_z = -B_y$. The initial configuration of the stationary state is given by $p = 1$, $\mathbf{u} = 0$, $B_y = B_z = 0.56418958$. The density ρ is 10 inside the cloud and 1 otherwise. The computational domain is $[0, 1]^3$ with outlet boundaries until the time $t_{\text{end}} = 0.06$. These simulations use the parameters $\nu = 0.3$, $\gamma = \frac{5}{3}$, and the scaled gradient threshold $\epsilon^{\rho} = 8.0$.

As in the other two configurations, the simulations were performed with one to three refinement levels, where the most refined mesh is 128^3 cells, using 2^n processors with n from one to three. The pressure solution on the adaptive mesh at t_{end} is presented in Fig. 5 (left). We can observe that the symmetry is almost perfectly preserved and the adaptive mesh refines all relevant structures, particularly the bow shock. The mesh distribution is well balanced considering, again, the difference in computational costs required by the adaptive meshes. Similar to the KH instability, density has the lowest errors and pressure the largest values as usual for these cases. Similarly to the other configurations, the errors increase when more aggressive mesh adaptation, *i.e.*, a higher number of levels, is employed.

The computational times in minutes of those experiments are given in Table 8. We observe that in these experiments the maximum scalability is near 1.8 for one level and one to two processor, and it reduces to a factor of 1.2 for level two and comparing the one and two processor cases.

Table 8
3D Shock-Cloud interaction: elapsed time of the computations using one to three refinement levels as a function of the number of processors. Simulations were performed in a workstation with processors Intel Xeon 2.20 GHz with 12 cores each.

		Processors			
		1	2	4	8
Levels	1	79.8	43.1	24.0	17.0
	2	8.2	6.5	3.4	2.5
	3	3.1	2.3	1.5	1.0

4. Conclusions

In this paper we presented some widely used verification tests for compressible ideal MHD in our new MHD solver in the AMROC framework. For one test case with available results, our solutions are in good agreeement with predictions from similar codes. Moreover, the accuracy of the numerical solutions also exhibits the expected behaviour comparing the uniform mesh and the adaptive meshes. Scalability and accuracy of the new MHD solver were particularly investigated. In general, all test cases show good parallel performance, confirming the quality of the implementation.

Acknowledgments

The authors thank the FAPESP SPRINT – University of Southampton (Grant:16/ 50016-9), FAPESP (Grant: 2015/ 25624-2), CNPq (Grants: 306038/2015-3, 140626/ 2014-0, 141741/2013-9), and FINEP (Grant: 0112052700) for financial support of this research. ML thankfully acknowledges financial support from CAPES (Grant: 88881.132489/2016-01) for his doctorate sandwich stage at the University of Southampton. We also thank M. Banik and V. E. Menconi for their helpful computational assistance.

References

[1] Tóth G, van der Holst B, Sokolov IV, De Zeeuw DL, Gombosi TI, Fang F. Adaptive numerical algorithms in space weather modeling. J Comput Phys 2012;231(3):870–903.
[2] Bittencourt J. Fundamentals of plasma physics. Springer; 2004.
[3] Hopkins PF. A constrained-gradient method to control divergence errors in numerical MHD. Mon Notices R Astron Soc 2016;462(11):576–87.
[4] Gomes AKF, Domingues MO, Schneider K, Mendes O, Deiterding R. An adaptive multiresolution method for ideal magnetohydrodynamics using divergence cleaning with parabolic-hyperbolic correction. Appl Numer Math 2015;95:199–213.
[5] Domingues MO, Gomes AKF, Gomes SM, Mendes O, Di Pierro B, Schneider K. Extended generalized lagrangian multipliers for magnetohydrodynamics using adaptive multiresolution methods. ESAIM: Proc 2013;43:95–107.
[6] Majda A. Compressible fluid flow and systems of conservation laws in several space variables. Springer; 1984.
[7] Brackbill JU, Barnes DC. The effect of nonzero $\nabla \cdot \mathbf{B}$ on the numerical solution of the magnetohysdrodynamic equations. J Comput Phys 1980;35:426–30.
[8] Dedner A, Kemm F, Kröner D, Munz C-D, Schnitzer T, Wesenberg M. Hyperbolic divergence cleaning for the MHD equations. J Comput Phys 2002;175(2):645–73.
[9] Mignone A, Tzeferacos P. A second-order unsplit Godunov scheme for cell-centered MHD: the CTU-GLM scheme. J Comput Phys 2010;229(6):2117–38.
[10] Leveque RJ. Finite volume methods for hyperbolic systems. Cambridge University Press; 2002.
[11] Miyoshi T, Kusano KA. A multi-state HLL approximate Riemann solver for ideal magnetohydrodynamics. J Comput Phys 2005;208:315–44.
[12] Toro EF. Riemann solvers and numerical methods for fluid dynamics. Springer; 1999.
[13] Berger M, Colella P. Local adaptive mesh refinement for shock hydrodynamics. J Comput Phys 1989;82:64–84.
[14] Deiterding R. Block-structured adaptive mesh refinement – theory, implementation and application. ESAIM: Proc 2011;34:97–150.
[15] Deiterding R. Parallel adaptive simulation of multi-dimensional detonation structures. Brandenburgische Technische Universität Cottbus; 2003. Ph.D. thesis.
[16] Deiterding R. Construction and application of an AMR algorithm for distributed memory computers. In: Plewa T, Linde T, Weirs VG, editors. Adaptive mesh refinement – theory and applications. Springer; 2005. p. 361–72.

[17] Bell J, Berger M, Saltzmann J, Welcome M. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. SIAM J Sci Comput 1994;15:127–138.
[18] Frank A, Jones TW, Ryu D, Gaalaas JB. The magnetohydrodynamic Kelvin–Helmholtz instability: a two-dimensional numerical study. Astrophys J 1996;460:777–93.
[19] Orszag SA, Tang C-M. Small-scale structure of two-dimensional magnetohydrodynamic turbulence. J Fluid Mech 1979;90(1):129–43.
[20] Ryu D, Miniati F, Jones T, Frank A. A divergence-free upwind code for multidimensional magnetohydrodynamic flows. Astrophys J 1998;509(1):244–55.
[21] Dai W, Woodward PR. A simple finite difference scheme for multidimensional magnetohydrodynamical equations. J Comput Phys 1998;142(2):331–69.
[22] Londrillo P, Del Zanna L. High-order upwind schemes for multidimensional magnetohydrodynamics. Astrophys J 2000;530(1):508–24.
[23] Picone JM, Dahlburg RB. Evolution of the Orszag–Tang vortex system in a compressible medium. II. Supersonic flow. Phys Fluids B 1991;3(1):29–44.
[24] Helzel C, Rossmanith JA, Taetz B. An unstaggered constrained transport method for the 3D ideal magnetohydrodynamic equations. J Comput Phys 2011;230(10):3803–29.

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programas de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. Aceitam-se tanto programas fonte quanto os executáveis.