Universidad de La Laguna

Facultad de Ciencias
Sección de Física

TRABAJO DE FIN DE GRADO

Development of a Simple Fluid Dynamics Numerical Code and Application to an Idealized Solar Atmosphere Model

Isaac Alonso Asensio

Supervisor:

Dr. Fernando Moreno-Insertis

Contents

Abstract i

Resumen ii

1 Introduction 1

1.1 State-of-the-art . . . 1

1.2 The Fluid Mechanics background . . . 2

2 Objectives and Methodology 5

2.1 Objectives . . . 5

2.2 Methodology . . . 6

3 Code development 7

3.1 Python . . . 8

3.2 Code structure . . . 8

3.3 Numerical schemes . . . 12

3.4 Source terms . . . 14

4 Results 17

4.1 Test cases . . . 17

4.2 Application to a highly idealized *Solar Atmosphere* model. . . . 25

5 Conclusions 35

6 Bibliography 37

Abstract

Advanced numerical and computational methods are crucial in different branches of astrophysics, particularly so in Solar Physics, which studies the Sun, its structure and dynamical and radiative processes. An specially interesting aspect of Solar Physics is the thermal structure of its atmosphere, and, within it, the reason that leads to the sharp transition between chromosphere and corona. Two fundamental ingredients in this respect are thermal conduction and radiative cooling. On the other hand, to understand its stratification and dynamics, the fluid dynamics equations with gravity term have to be solved.

In this work, a simple, modular, one-dimensional code written in Python has been developed to study this region from an idealized point of view. The code solves the gas dynamic equations with realistic terms both for thermal conduction and radiative cooling. Furthermore, gravity is also included, as well as an ad-hoc momentum damping term to smooth the evolution if needed. Different testcases have been used to verify the implementation, each of them aimed at testing different parts of the code, namely: *(a)* its core, *(b)* the numerical schemes, *(c)* gravity and damping and *(d)* thermal conduction.

After the testing phase, where the physical and numerical aspects where tested separately, the code is used to obtain dynamical patterns of evolution toward hydrostatic and thermal equilibria that resemble the thermal structure of the solar atmosphere. Before obtaining these dynamic evolutions, a set of ordinary differential equations is obtained that fulfill the condition of hydrostatic and thermal equilibrium. They are solved analytical or numerically, depending on whether the radiative cooling is used or not. The study of the dynamical evolution, carried out through numerical simulations, has been split into: *(a)* cases where only gravity and thermal conduction is considered, and *(b)* cases where the radiative cooling is also taken into account. For the latter, the influence of the boundary condition on the evolutions and final equilibria has been found critical. Once the boundary conditions are carefully set, the obtained equilibrium after the dynamical evolution of the system is compared with the solutions for the ordinary differential equations.

The resulting equilibrium configuration shows a dense and cold zone in the lower part of the domain (similar to the chromosphere) caused by radiative cooling, where the thermal conduction is negligible. Above this region, a sharp gradient in both temperature and density appears, forming the so-called transition region. The hot upper part of the atmosphere (similar to the corona) is dominated by thermal conduction. With the preceding, the physical/astrophysical objective of this *Trabajo de Fin de Grado* is accomplished.

Resumen

Una de las ramas de la astrofísica donde los métodos numéricos y computacionales avanzados son más importantes es la Física Solar, que se encarga de estudiar el Sol, su estructura y procesos dinámicos y radiativos. Un aspecto particularmente interesante de la Física Solar es la estructura térmica de su atmósfera, y, dentro de ella, el motivo por el que hay un salto abrupto de temperatura entre la cromosfera y la corona. Para entender los aspectos puramente de estructura térmica, dos ingredientes fundamentales son la conducción térmica y el enfriamiento radiativo. Para entender su estratificación y dinámica hay que resolver las ecuaciones de los fluidos con término de gravedad.

En este trabajo, se desarrolla un código simple, modular y unidimensional escrito en Python para estudiar esta región de manera idealizada. Este código resuelve las ecuaciones de la dinámica de gases con términos realistas tanto para la conducción térmica como para el enfriamiento radiativo. También se incluye la gravedad y un amortiguamiento ad-hoc en caso de que fuera necesario para suavizar la evolución del sistema hacia el equilibrio. Se usan distintos casos de prueba para comprobar que el código está correctamente implementado cada uno de ellos centrado en diferentes partes del código, a saber: *(a)* el núcleo de la implementación, *(b)* los esquemas númericos, *(c)* gravedad y amortiguamiento, y *(d)* conducción térmica.

Después de haber comprobado sus distintos módulos y aspecto físicos por separado, usamos el código para obtener la evolución dinámica hacia configuraciones de equilibrios hidrostático y térmico que capturen la esencia de la estructura de la región de transición solar. Antes de obtener la evolución dinámica, se derivan las ecuaciones diferenciales ordinarias que describen una situación de equilibrio estático, y se solucionan analítica o numéricamente, dependiendo de si se considera enfriamiento radiativo o no. El estudio de la evolución dinámica, llevado a cabo mediante simulaciones numéricas, lo hemos separado en casos donde solo se usa la conducción térmica y la gravedad, y casos donde además se incluye el enfriamiento radiativo. En el último caso, las condiciones de contorno influyen de manera crítica en la evolución y equilibrio del sistema. Una vez las condiciones de contorno están correctamente fijadas, la configuración de equilibrio resultante de la evolución se compara con las soluciones de las ecuaciones diferenciales ordinarias.

La configuración obtenida para el equilibrio muestra en la zona inferior una zona densa y fría (similar a la cromosfera) causada por el enfriamiento radiativo, donde la conducción térmica no tiene efecto. En el límite superior de esta región se desarrollan altos gradientes de temperatura y densidad (región de transición), que conectan con una región caliente (corona), gobernada por la conducción térmica. Con ello alcanzamos el objetivo físico/astrofísico del Trabajo de Fin de Grado.

Chapter 1

Introduction

En este capítulo, se dará una justificación de por qué son necesarios los métodos numéricos para resolver problemas de interés físico. Se tratará en concreto la dinámica de los fluidos, describiendo sus ecuaciones, junto a los términos fuente de interés en este trabajo. A continuación, en una pequeña introducción sobre la Física Solar, en concreto de la atmósfera solar, se darán la forma de los términos de interés para este trabajo, a saber: la conducción térmica y el enfriamiento radiativo ópticamente delgado.

Throughout this chapter, a brief justification of why we need numerical codes to solve physical problems is presented (Sec 1.1). Afterwards, the theoretical background will be studied (Sec 1.2), divided into a general description of the fluid dynamical aspects, followed by a small introduction to the Solar Physics background of this work, more specifically to the link between *corona* and *chromosphere* via the *transition region*.

1.1 State-of-the-art

The dynamical phenomena in the stellar interiors and atmospheres are of a complex nature, and are governed by the Fluid Equations. In most cases there is no general analytical solution, so there is a need to solve numerically these equations to get insights into the dynamical behavior of this regions. A first contact with one-dimensional numerical codes in the Physics Degree is of great value, since they facilitate a better understanding of the physics underlying these phenomena, and of the numerical methods used to solve a system of partial differential equations.

The closest star to us is the Sun, and it has been extensively observed and studied for

a long time. From those observations, the thermal structure of the solar atmosphere has been described, although there are still many unknown aspects. At the bottom of the solar atmosphere one finds the *photosphere* and, above it, the *chromosphere*. Both are cold regions, where the temperature is between about $6000K$ and $20000K$. Above these, lies the *corona*, where the temperature is on the order of one million Kelvins. The latter is connected to the *chromosphere* through a region with a steep temperature gradient, namely the *transition region*.

The study of the solar atmosphere is carried out nowadays using large, parallel, multi-dimensional computer codes (like COOLFluiD [Lani et al., 2013] [Maneva et al., 2017], Bifrost [Gudiksen et al., 2011], MANCHA [Felipe et al., 2010], MURaM [Vögler et al., 2005], CO5BOLD [Freytag et al., 2012]). They are the result of an intensive team effort along many years. However, to understand basic aspects of the structure and time evolution of the atmosphere, one can often use simple one-dimensional codes. In particular, it is possible, thanks to 1D codes to obtain a first approximation to the thermal structure of the solar atmosphere. With these codes, the role of different physical phenomena and their contributions to the equilibrium can be well studied. For the upper levels of the atmosphere (transition region and corona), there are three important phenomena that determine the temperature, density and pressure profiles, namely: thermal conduction, radiative cooling and gravity [Priest, 2014]. Additionally, if the code contains not only thermal aspects but also hydrodynamics, a brief study of how the system evolves toward the equilibrium configuration, from a non-equilibrium initial condition, can be carried out.

1.2 The Fluid Mechanics background

In the following, the equations that govern the behavior of compressible gases will be described (Sec. 1.2.1), and, then, the particular form and values of a few of the non-ideal terms will be given (Sec. 1.2.2).

1.2.1 Fluid mechanics

The gas dynamics equations can be expressed in conservative form as:

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \boldsymbol{u}) = 0 \tag{1.1a}$$

$$\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \vec{\nabla} \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u} + p \bar{\bar{I}}) = -\mathscr{P} \tag{1.1b}$$

$$\frac{\partial E^{(t)}}{\partial t} + \vec{\nabla} \cdot \left((E^{(t)} + p) \boldsymbol{u} \right) = -\mathscr{L} \tag{1.1c}$$

The left-hand side of equations (1.1), when set equal to zero, corresponds to the classical conservation equations of ideal hydrodynamics without gravity, with $\bar{\bar{I}}$ the unit tensor, $\otimes$ the standard tensor product and $E^{(t)}$ the total energy (kinetic plus internal) per unit volume; the symbol ϵ is used for the thermal energy per unit mass. On the right-hand side, various momentum and energy sources, $\mathscr{P}$ and $\mathscr{L}$ respectively, are included.

$$\mathscr{P} = -\rho \boldsymbol{g} + D_m \tag{1.2a}$$

$$\mathscr{L} = \vec{\nabla} \cdot \boldsymbol{q} - \rho \boldsymbol{u} \cdot \boldsymbol{g} + L_r + D_e \tag{1.2b}$$

In the present work, we will be considering sources related to the action of gravity (with constant gravity acceleration $\boldsymbol{g}$), the heat flux $\boldsymbol{q}$ and a cooling term, L_r. Additionally, an ad-hoc damping term, D_m, to remove momentum ad-hoc as needed in the calculation (see Sec. 3.4.4)and the corresponding removal of energy, D_e, are also included.

In the equation (1.2b) it is important to note that the heat flux term, in contrast to the other terms, is in divergence form, so it must be subjected to discretization using finite differences, like the terms on the left-hand side of (1.1).

The equations (1.1) should be closed by the equation of state (*EOS*), $P = P(T, \rho)$. As we will be considering the simplest ideal gas, our EOS will be $P = \rho RT/\mu$ and its internal energy can be expressed as $\epsilon = c_v T = \frac{p}{\rho(\gamma - 1)}$

1.2.2 The thermal conduction and radiative cooling terms in the solar atmosphere

To explain the basic features of the thermal structure of the solar atmosphere and, in particular, the reason for the steep temperature gradient at the solar transition region, heat conduction and radiative cooling must be taken into account. The solar transition region is a few hundred kilometers thick, and is typically a corrugated surface at an average height of about 2 000 km above the photosphere, connecting the chromosphere and the corona. In this thin region, the temperature increases from about $2 \cdot 10^4 K$ up to $10^6 K$.

For this work, the heat flux $\boldsymbol{q} = -\kappa \nabla T$ is taken from [Spitzer, 1962], which gives, for a fully ionized hydrogen plasma in the direction of the magnetic field lines, or when the magnetic field is weak:

$$\kappa \equiv 1.8 \cdot 10^{-10} \frac{T^{5/2}}{\ln \Lambda} \approx 9 \cdot 10^{-12} T^{5/2} \quad W m^{-1} K^{-1} \tag{1.3}$$

For the second equality, the Coulomb logarithm ($\ln \Lambda$) has been taken equal to 20, a typical coronal value [Priest, 2014].

Regarding the *radiative losses* (L_r), they can be expressed for an optically thin atmosphere as:

$$L_r = n_e n_H \Lambda(T) = n^2 \Lambda(T) \tag{1.4}$$

The electron and hydrogen number densities (n_e and n_H respectively) can be set equal if the plasma is considered to be composed of fully ionized Hydrogen. $\Lambda(T)$ has been evaluated by various authors, using different elemental abundances. For our case, we will be using the values given in tabular form by [Dere et al., 2009].

Chapter 2

Objectives and Methodology

En este capítulo se planterán los objetivos de este Trabajo de Fin de Grado y cuál es la metodología seguida para cumplirlos. Los objetivos están divididos en tres grupos, dependiendo del área del conocimiento necesaria para alcanzarlos. La metodología es en muchos casos común: usar un código de desarrollo propio para entender una pequeña parte de la Física detrás de la atmósfera solar.

In the following, the objectives for this *Trabajo de Fin de Grado* will be enumerated (Sec. 2.1). Afterwards, the methodology used to achieve these objectives will be described in Section 2.2

2.1 Objectives

The objectives of this work can be separated into three groups, according to the knowledge needed to achieve them:

- To learn numerical methods to solve a system of partial differential equations, specifically the gas dynamics equations (1.1) with gravity and the non-ideal terms given by (1.2). To develop a modular code for solving one-dimensional problems and understand how to efficiently implement the numerical methods needed.
- To get acquainted with the most basics features of the thermal structure at the solar atmosphere, taking into account realistic terms for the thermal conduction and the optically thin radiative cooling. To obtain analytical solutions or easily integrable ordinary differential equations (ODEs), with and without radiative cooling.

- To have a first contact with the dynamics of the gases in the solar atmosphere, studying how the system reaches the equilibrium from an idealized initial condition. To understand how certain numerical features (like, e.g., the boundary conditions) can influence the physical solution, and analyze them. To compare the obtained equilibrium configuration with the solution of the ODE.

2.2 Methodology

To solve the gas dynamics equations (1.1) along with the source terms (1.2), the way to proceed is to develop from scratch a numerical code, using the Finite Difference Method. This code will be extensively tested through a series of validation numerical experiments that will confirm the correctness of different parts of the implementation. So this code, which is a essential part of this *Trabajo de Fin de Grado*, will be both a result and a tool throughout this work.

From the gas dynamics equations, a static equilibrium configuration can be obtained solving analytically the differential equations, following standard procedures. In the case where an exact analytical solution is not available, then the ODE will be solved numerically.

To better understand the gas behavior in the solar atmosphere, the code will be used to obtain a static equilibrium that correctly matches the analytical/ODE solution. Firstly, a simple case without radiative cooling will be run, and different initial conditions will be given to the code to study how the system evolves toward the equilibrium configuration. Afterwards the radiative cooling will be added and again the evolution and final equilibrium will be studied. Knowing the solution for the equilibrium obtained before (analytically or integrating an ODE), it can be compared with the final equilibrium configuration obtained with the code.

Chapter 3

Code development

A lo largo de este capítulo se discutirá la estructura que presenta el código para que sea modular y eficiente, así como la justificación de Python como el lenguaje de programación escogido. Se explicarán los esquemas numéricos actualmente implementados en el código, así como los términos no ideales que se usarán durante el resto del trabajo.

The first important objective, as stated in Sec. 2.1, is to develop a numerical code able to solve the gas dynamic equations in one dimension, with gravity, thermal conduction and radiative cooling. The code to develop must be a useful, flexible tool that allows an in-depth acquaintance with the basic ideas behind the numerical solution of fluid dynamics problems.

The core of the code is similar to the one developed by the students of the course *Técnicas de Simulación Numérica* of the Master of Astrophysics, but we have created it here in Python from scratch as part of the present *Trabajo de Fin de Grado*. It can be further improved and extended due to its high modularity, which allows the user to integrate new algorithms without actually modifying the whole code and its structure. Also, the structure itself can be further improved in future works using OOP (Object-Oriented Paradigm).

This code is designed for execution in a single CPU. However, it is written using Python's *numpy* mathematical program library, which admits of parallelization using the MPI protocol.

The project, which has more than 1200 non-empty lines of code, is open-source and can be found on *GitHub* [1], and downloaded using the most common version control programs, as `git` or `svn`. Most of the changes are logged in `changes.log`, as well as in the commit history of the project in *GitHub*.

[1] `https://github.com/Isaac-a95/TFG`

In the following section, the justification for the use of Python to build the code will be given (Sec. 3.1), and then the code structure will be discussed (Sec. 3.2). Afterwards the implemented numerical schemes are presented and explained in Sec. 3.3. Finally, the implementation of the *Source Terms* is presented throughout Sec. 3.4.

3.1 Python

Almost any programming language is able to solve this kind of problems, but obviously there are advantages and disadvantages depending upon which one is used. We can classify the programming languages into two big groups, interpreted and compiled ones.

The first group, which includes *Python*, *IDL*, *MATLAB*, executes each command in a sequential way, reading directly from the *script* file. Also, simple commands can be called through the terminal. Usually, these languages are object oriented and can easily tackle arrays, use complex algorithms (which are actually written in a compiled language), and most importantly, they are really easy to understand and learn from scratch.

The second group includes *C*, *C++* or *Fortran*, and instead of running each command from the *script* file, they need to be *compiled*, and then an executable is created and run. Usually they are faster, and can control precisely memory allocation, making them the most suitable for computationally extensive applications (such as Computational Fluid Dynamics)

However, in this work an interpreted language is used (*Python*). This will make the code more readable and easier to use, but slower. On the other hand, *Python* has plenty of plugins and modules which can tackle cost expensive computations efficiently (*numpy*, *scipy*), because their backbone is written in a compiled language. Another important *pro* of Python is the ease to make plots with the results and save the results, and save them both to images or data files.

3.2 Code structure

In line with the state-of-the-art in all modern computer codes, big or small, in particular those to solve the equations of fluid dynamics, we have given our code a modular structure. Each module is in charge of a specific task within the flow of the code. The core of the code is the main program contained in the file `Main.py`. Inside this file, all the other modules are imported and the main loop of the program is defined; this is, in fact, the *only* explicit Python loop. Hereafter, we will refer to modules (e.g. `Main`) as the variables and functions inside a file with the same name (e.g. *Main.py*).

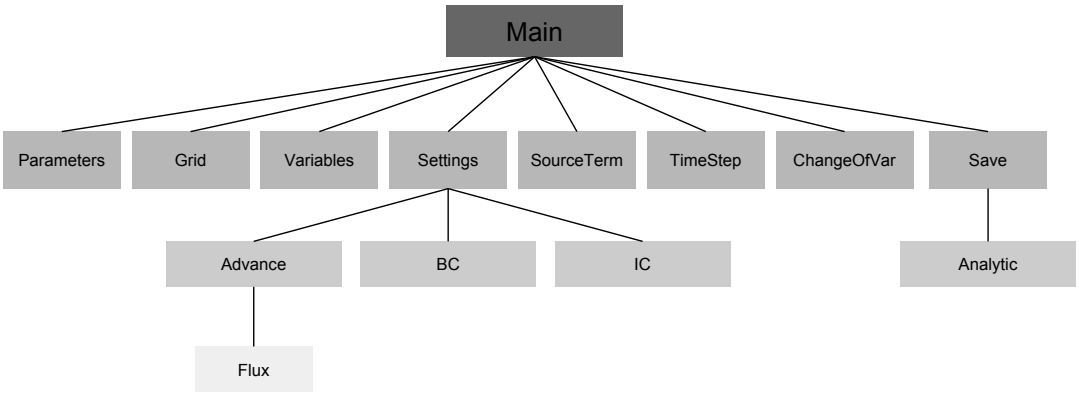


Figure 3.1: Module structure of the code

The code has been designed such that, to use it as an external tool, only the modules `Main`, `Parameters` and `Settings` must be modified. The first one, as stated before, includes all the imports and the main loop. The second module stores all the user-defined parameters needed to run the simulations, all of them being either boolean variables (like `IsThereGravity`), or float/integers ones (e.g. `gamma`). `Settings` is used to select from the different implemented numerical schemes, boundary/initial conditions which ones should be used. For example, to select the numerical scheme, the variable `Scheme` is set equal to the function inside the module `Advance` that we want to use. This way, each time that in another part of the code `Settings.Scheme()` is called, then the numerical scheme inside `Advance` will be run. As an example, the files for running the case of sinusoidal sound waves can be found in Listing 3.1 and 3.2

```
#------------------------------------------
#  Parameters.py
# In this module all the different user
# defined parameters are set.
#
#------------------------------------------

import numpy as np

print "Loading Variables.."

# ------------------- MESH ---------------------
N = 100
z0 = 0.
zf = 1.

# ------------------- TIME SETUP ---------------

dt = dt_max
tt = 0.
tf = 1.
it = 0
max_it = np.inf
cfl_set = 0.9
cfl = cfl_set

# ------------------- PLOT SETUP ---------------

FolderName = 'RESULTS_TESTCASES/SoundWaves_FirstGen'
save_rate = 1
```

```
SaveToFile = True
SaveToFileRatio = 1

PlotFile = False
FileToPlot = ''

PlotCharacteristics = False

# Soundwaves
SoundSpeedLine = False
SoundSpeedAnalytic = True
rhoAxis = [0.995,1.005]
vAxis = [-.005, .005]
PAxis = [0.995, 1.005]
TAxis = [1.495, 1.505]

# ------------------- SOURCE TERMS ---------------

IsComputingSource = False

# No need for more Source terms parameters

# ------------------- EQUATION OF STATE ----------

gamma = 5./3.
R = 1.
Na = 1
molarMass = 1.
mu = 1.
cv = 1.

def Computecv():
    global cv
    cv = R/(mu*(gamma-1.)*molarMass*Na)
```

Listing 3.1: `Parameters.py` for the Sound Waves testcase

```
#-------------------------------------------
#  Settings.py
# This module defines the different functions
# and numerical schemes used for the simulation
#
#-------------------------------------------

import numpy as np
import Advance
import BoundaryConditions
import InitialConditions
import Parameters as par

print 'Loading Settings..'

InitialCondition = InitialConditions.SoundWaves
argsIC = [1.0, 0.001, 1.0, 4.]

BoundaryConditionL = BoundaryConditions.Periodic
argsL = []
BoundaryConditionR = BoundaryConditions.Periodic
argsR = []

Scheme = Advance.LaxFriedichs
```

Listing 3.2: `Settings.py` for the Sound Waves testcase

The actual flow of the code is shown in Figure 3.2. This follows a standard loop, which eventually will stop either if the maximum number of iterations is reached, or the time advances beyond a preset value (both stored at `Parameters`).

The first step after importing the parameters consists in creating the numerical mesh or grid and declare the variables to be used, i.e., allocate them the correct size and variable

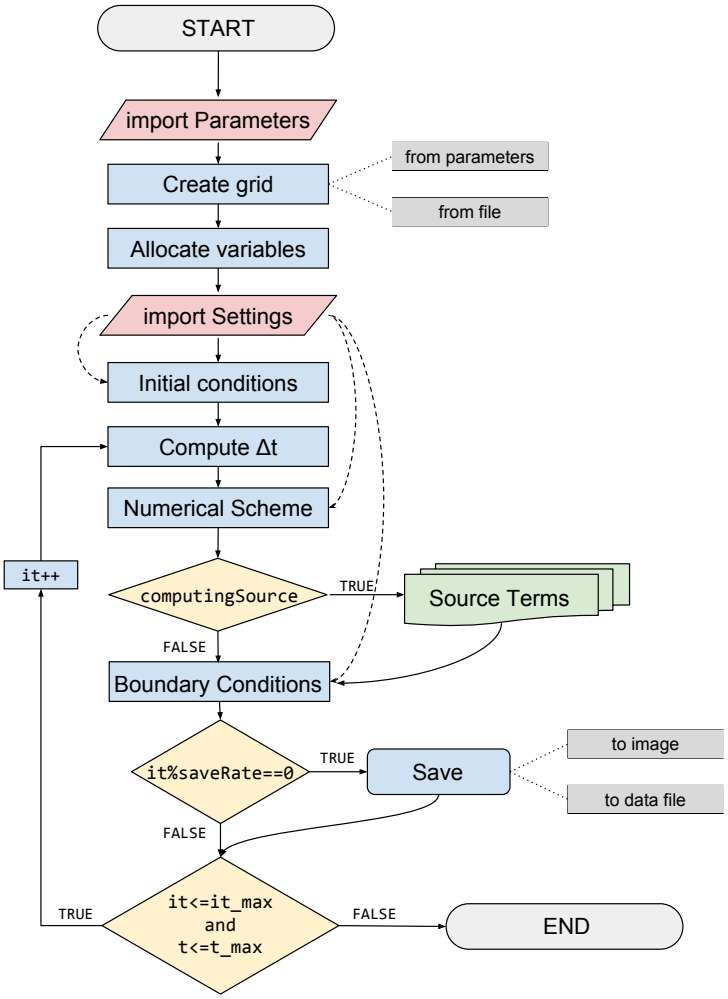


Figure 3.2: Flow chart of the code

type. The variables can be loaded via an analytical expression implemented in the module `InitialConditions`, or directly read from a file, written in different optional formats (typically containing the result of previous experiments). Once the *ICs* are set, then the main loop of the program starts. An important first stage in the calculation is to determine the timestep to be used, following the classical Courant-Friedrichs-Lewy (CFL) criterion. After that, the time advance is carried out, in two parts. Firstly, the variables are updated using the left-hand side of equations (1.1) only, i.e., the classical ideal hydrodynamics terms, and applying the numerical scheme selected at `Settings`. Secondly, the non-ideal terms are incorporated using a standard Operator Splitting technique (Sec. 3.4). Once the time advance is done, the *BCs* are set.

The results can be output either to plots (saved as `.png`) or to data files (`.dat`), and the

frequency at which the code creates those is user-defined.

3.3 Numerical schemes

The main numerical scheme is in charge of making an update of the conservative variables (i.e. obtain ρ^{n+1}, $(\rho \boldsymbol{u})^{n+1}$ and $\left(E^{(t)}\right)^{n+1}$) according to the ideal hydrodynamic equations, i.e, (1.1) with $\mathscr{P} = \mathscr{L} = 0$.

As of today, two different numerical schemes have been implemented in the code, namely the Lax-Friedrichs scheme and the Richtmyer two-step Lax-Wendroff method. Both of them are explicit, in the sense that they calculate the variables at the advance time requiring only the values of the variables at the current or previous times.

But before going into the numerical schemes themselves, the notation used in this work will be explained:

- When referring to spatial points $(..., i-1, i, i+1, ...)$ the index will be written as a subindex, whereas when referring to time points $(..., n-1, n, n+1, ...)$ it will be written as a super-index. For example, a function $f(z,t)$ evaluated at z_i at the timestep n will be written as f_i^n.
- It is important to note that, for a one-dimensional mesh of N points, there are actually $N+2$ nodes. The two extra nodes are called *ghost nodes* and are not part of the physical domain. They are used to set the boundary conditions, and, following the standard notation in Python or IDL, are expressed as $i = 0$ and $i = -1$ for the left and right boundaries, respectively. In the same fashion, a point located left to the $i = -1$ node, will be noted as $i = -2$ and so on.

3.3.1 Lax-Friedrichs

The *Lax-Friedrichs scheme* [LeVeque, 1992] [Laney, 1998], due to its simplicity, was the first one implemented. It is second order in space, and first order in time. It is derived as a simple discretization of the spatial and times derivatives, yielding, for a general set of variables $\boldsymbol{\psi}$ governed by a conservative law $\frac{\partial \boldsymbol{\psi}}{\partial t} + \frac{\partial}{\partial z}\left(f(\boldsymbol{\psi})\right) = 0$:

$$\boldsymbol{\psi}_i^{n+1} = \frac{1}{2}\left(\boldsymbol{\psi}_{i+1}^n - \boldsymbol{\psi}_{i-1}^n\right) - \frac{\lambda}{2}\left(f(\boldsymbol{\psi}_{i+1}^n) - f(\boldsymbol{\psi}_{i-1}^n)\right) \tag{3.1}$$

Where $\lambda = \frac{\Delta t}{\Delta z}$. This scheme is CSFT (*Centered in Space, Forward in Time*). For the case of the ideal hydrodynamics equations (1.1) it can be shown to be numerically stable if a simple

Courant-Lewy-Friedrichs (CFL) condition is fulfilled, namely:

$$\Delta t < \frac{\Delta z}{\max_{i=0,..,N}\left(|(c_s)_i + u_i|, |(c_s)_i - u_i|\right)} \tag{3.2}$$

However, it turns out to have too much numerical diffusivity, so, for instance, simple wave solutions tend to decrease their amplitude in a short physical time.

3.3.2 Richtmyer two-step Lax-Wendroff method

The main scheme we will be using for the ideal hydrodynamics part of the time update belongs to the family of the so-called first-generation schemes, specifically from the Lax-Wendroff family [Lax, 1959]. These schemes are second order in both space and time, but in a general formulation, they require to compute the jacobian matrix, which can be computationally intensive.

To avoid this, there are matrix-free methods, such as the Richtmyer or MacCormack methods [LeVeque, 1992]. In particular, we will use the first one, which make two time steps to advance from time n to $n+1$. To do so, the first step computes the values for an intermediate $n+\frac{1}{2}$ time step at points displaced by half a grid cell (i.e. by $\Delta z/2$). With these, the final solution is computed at time $n+1$. This can be sketched as shown in Figure 3.3. The final formula for this scheme can be expressed as:

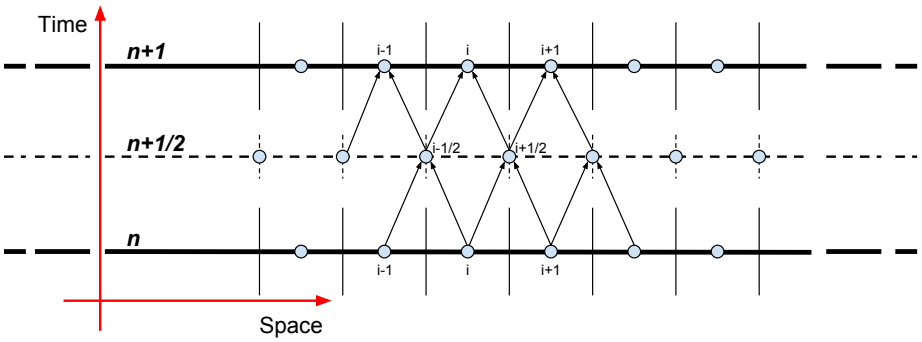


Figure 3.3: Richtmyer two-step Lax-Wendroff method visualization

$$\text{First step:} \qquad \bar{\boldsymbol{\psi}}^{n+1/2}_{i+1/2} = \frac{1}{2}\left(\boldsymbol{\psi}^n_i + \boldsymbol{\psi}^n_{i+1}\right) - \frac{\lambda}{2}\left[\mathbf{f}(\boldsymbol{\psi}^n_{i+1}) - \mathbf{f}(\boldsymbol{\psi}^n_i)\right] \tag{3.3a}$$

$$\text{Second step:} \qquad \boldsymbol{\psi}^{n+1}_i = \boldsymbol{\psi}^n_i - \lambda\left[\mathbf{f}(\bar{\boldsymbol{\psi}}^{n+1/2}_{i+1/2}) - \mathbf{f}(\bar{\boldsymbol{\psi}}^{n+1/2}_{i-1/2})\right] \tag{3.3b}$$

3.4 Source terms

To include the effects of the source terms (1.2), different approaches can be taken. In this work, the *Operator Splitting* technique was used. In this technique, one first advances the variables as indicated in (3.1) or (3.3) for the ideal hydrodynamics part. Thereafter, the resulting variables are updated a second time adding the effect of the gravity and non-ideal terms.

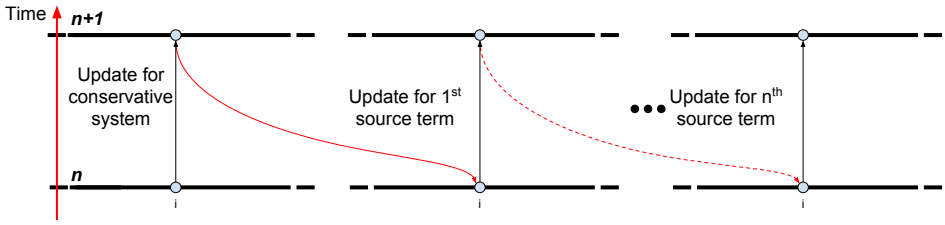


Figure 3.4: Operator Splitting technique

The advantage of this approach is that one can use completely different schemes for the ideal and non-ideal terms, according to their physical nature and numerical requirements. Additionally, this perfectly matches the idea of a modular code.

3.4.1 Gravity term

In order to successfully simulate the solar atmosphere, it is mandatory to compute the gravitational forces acting on a fluid element. The solar gravity is taken constant and equal to $27360 \frac{cm}{s^2}$. Even though the scales that we are working with are of mega-meters ($10^8 cm$), for simplicity the solar gravity is taken constant. To justify this simplification, note that in a range of $10Mm$ at the surface of the Sun, the acceleration of gravity decreases by a factor $\approx 1\%$, so its change can be neglected. The gravity terms are used to update the momentum and energy variables as follows:

$$(\rho u)_i^{n+1} = (\rho u)_i^n + \Delta t \rho_i^n g \tag{3.4}$$

$$(E^{(t)})_i^{n+1} = (E^{(t)})_i^n + \Delta t (\rho u)_i^n g \tag{3.5}$$

3.4.2 Thermal conduction

The net heat flux ($\boldsymbol{q}$) for a general system is $-\bar{\bar{\kappa}} \nabla T$, where the conduction coefficient κ can be anisotropic (tensor form), depend on different variables and so on. However, in this case

only the isotropic case will be studied (κ is a rank-zero tensor). The dependence upon the thermodynamical variables can vary for different physical systems. In this work, κ will be set constant for testing purposes, and for the actual calculations, $\kappa \propto T^{\frac{5}{2}}$ will be used (see Sec. 1.2.2). The total contribution of the heat conduction in the energy equation is $\nabla\kappa\cdot\nabla T+\kappa\nabla^2 T$, which can be computed following an explicit or implicit scheme.

3.4.2.1 Explicit scheme

For this scheme, the spatial derivatives are discretized in such a way that there is no dependence on the $n+1$ step. A centered scheme is used for both first and second derivatives.The resulting formula for one dimension yields:

$$(\rho\epsilon)_i^{n+1} = (\rho\epsilon)_i^n + \Delta t \frac{\left[\frac{1}{4}\left(\kappa_{i+1}^n - \kappa_{i-1}^n\right) + \kappa_i^n\right] T_{i+1}^n - 2\kappa_i^n T_i^n + \left[\kappa_i^n - \frac{1}{4}\left(\kappa_{i+1}^n - \kappa_{i-1}^n\right)\right] T_{i-1}^n}{(\Delta z)^2} \tag{3.6}$$

Although this is a simple scheme, and easy to implement and compute, it has an important drawback. In order to be numerically stable the time step should be smaller than $f_{clf}\frac{c_v\rho(\Delta z)^2}{\kappa}$, where f_{cfl}, the CFL factor, is 1/2 for the scheme presented at (3.6). This condition is a huge problem when trying to increase the spatial resolution of the simulation, as there is a quadratic dependence on the grid spacing. For fine grids, it can become orders of magnitude below the timestep given by the *CFL* condition for the ideal hydrodynamics update, thus it takes thousands or even millions of iterations to reach the dynamic characteristic times.

3.4.2.2 Implicit scheme

A much better option implemented in the code is an implicit scheme, where the temperature derivatives are computed at $n+1$. This yields a system of equations for the internal energy.

$$(\rho\epsilon)_i^{n+1} = (\rho\epsilon)_i^n + \Delta t \frac{\left[\frac{1}{4}\left(\kappa_{i+1}^n - \kappa_{i-1}^n\right) + \kappa_i^n\right] T_{i+1}^{n+1} - 2\kappa_i^n T_i^{n+1} + \left[\kappa_i^n - \frac{1}{4}\left(\kappa_{i+1}^n - \kappa_{i-1}^n\right)\right] T_{i-1}^{n+1}}{(\Delta z)^2} \tag{3.7}$$

Defining $\lambda = \frac{\Delta t}{\Delta z^2}$ and rearranging to make apparent the matrix structure.

$$\frac{\lambda}{c_v \rho_{i+1}^n}\left[\frac{1}{4}\left(\kappa_{i+1}^n - \kappa_{i-1}^n\right) + \kappa_i^n\right](\rho\epsilon)_{i+1}^{n+1} - \left[\frac{\lambda\kappa_i^n}{c_v\rho_i^n} - 1\right](\rho\epsilon)_i^{n+1} + \frac{\lambda}{c_v\rho_{i-1}^n}\left[\kappa_i^n - \frac{1}{4}\left(\kappa_{i+1}^n - \kappa_{i-1}^n\right)\right](\rho\epsilon)_{i-1}^{n+1} = -(\rho\epsilon)_i^n \tag{3.8}$$

This matrix, if the boundary conditions are non-periodic, is tridiagonal and can be easily solved thanks to multiple algorithms already implemented within *scipy*, as `solve_banded`.

This scheme is unconditionally stable, i.e., does not require timestep limitations to guarantee numerical stability, and thus the time step can be taken longer, without compromising the stability of the simulation.

3.4.3 Radiative losses

The L_r term corresponds the energy losses due to optically thin radiative cooling of the elements in the corona. In agreement with the general theory, L_r can be written as a loss function $\Lambda(T)$ times the product of the hydrogen and electron number density: $L_r = \Lambda(T) n_H n_e$. There are still various possibilities for $\Lambda(T)$ depending on the abundances and number of spectral lines chosen to calculate it [Priest, 2014]. Throughout this work we have used the results given by [Dere et al., 2009], with a smooth cutoff for temperatures lower than $3 \cdot 10^4 K$ to make the cooling negligible in the chromosphere.

3.4.4 Momentum damping

To obtain a dynamical (and thermal) equilibrium, a damping term can be needed if the system does not tend to the equilibrium configuration by itself. This term will be in charge of eliminating the excess momentum and energy associated with the transition from the initial condition to the final configuration. The damping must be of the right size so that the fluids moves smoothly toward the equilibrium without developing shocks nor strong bouncing waves, but without choking the system. The main idea is to subtract a fraction of the momentum ($D_{\%}$), and the corresponding kinetic energy at each iteration. The user can define this ratio, which is highly case-dependent, and can also set a maximum velocity. In this case, the damping will only be applied at nodes whose velocity is higher than this threshold.

Chapter 4

Results

En este capítulo se presentan los resultados de las diferentes simulaciones numéricas que se han realizado en el marco de este trabajo. En primer lugar, se discuten los resultados de distintos casos de prueba, cuya función es asegurar que el código desarrollado funciona correctamente. Posteriormente, se estudian las simulaciones que tratan la estructura térmica de la atmósfera solar, primero sin enfriamiento radiativo, y luego con él. Los resultados se comparan con la solución de la ODE derivada para el equilibrio estático.

In this section the different physical and numerical results will be discussed. Firstly, a set of cases whose solution can be independently obtained via solution of an ordinary differential equation will be studied. This is done to test the correctness of the code, and to compare the different numerical schemes in the code (Sec. 4.1). Afterwards, the main numerical results of this work concerning the thermal structure of the solar atmosphere will be obtained and discussed (Sec. 4.2). This will be done in two phases: *(a)* equilibria that do not involve radiative cooling (Sec. 4.2.1) and *(b)* equilibria with radiative cooling (Sec. 4.2.2). In both cases, the dynamical processes that take the initial condition towards the final equilibrium is calculated and the nature of the final equilibrium studied.

4.1 Test cases

For the following cases, a dimensionless EOS for an ideal gas will be used, i.e. $p = \rho T$. Throughout this work we adopt $\gamma = 5/3$ for the specific heat ratio.

4.1.1 Constant Flow

The zero-test for the code is aimed at checking the core of the code, and that all the important modules are correctly working together. The ideal hydrodynamic equations have a simple solution in the form of a spatially uniform distribution of density, pressure and (generally non-zero) velocity which is constant in time. Given uniform distributions of ρ, p and u as initial condition, the test consists in checking if the code maintains those values with no variation for a sufficiently long time. The Lax-Friedrichs scheme will be used, as it was the first implemented in the code.

Using as initial conditions the uniform values of pressure, density and velocity $p_0 = 1$, $\rho_0 = 1$ and $v_0 = 1$, a uniform and constant solution is indeed obtained. The solution is shown in the accompanying animation `4-1-1_ConstantFlow.mp4`. An image for the last snapshot is shown in Figure 4.1. This solution is maintained for a physical time of at least $t \sim 25\,000$, approximately 32 500 sound crossing times[1], which is sufficient for the test.

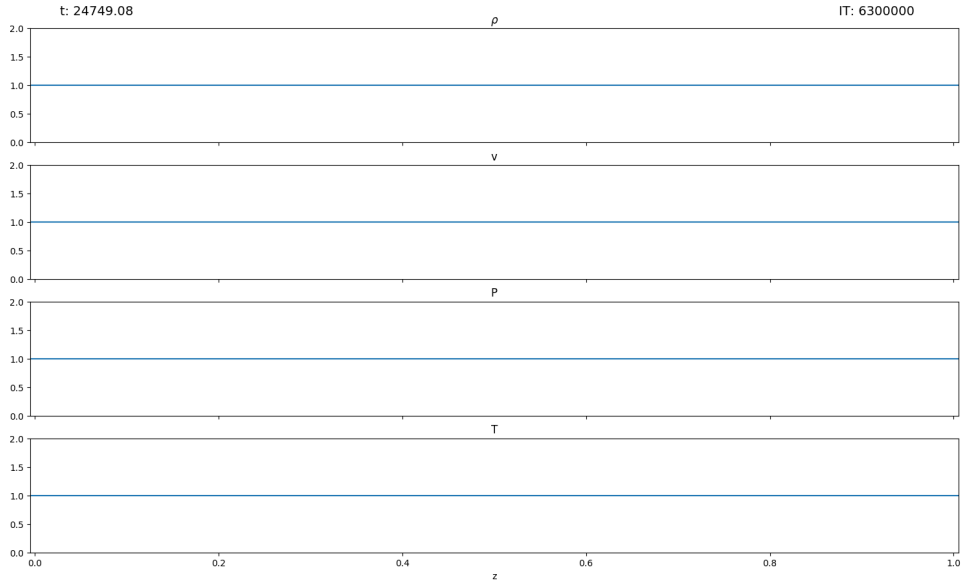


Figure 4.1: Equilibrium for a constant flow.

4.1.2 Sound waves

For the second test, both implemented numerical schemes will be tested with the same conditions, to briefly study the numerical diffusion of both. This comparison will be done

[1] Time taken for a sound wave to travel through the domain. In the present case, $c_s = 1.29$, as explained in the following subsection 4.1.2.

with simple harmonic sound waves propagating through a medium with constant density and temperature. For this case, there is an analytical solution given by:

$$\varphi = 2N_{wave}\pi\frac{z}{z_f - z_0} \tag{4.1a}$$

$$\rho = \rho_0\left(1 + A\cos(\varphi)\right) \tag{4.1b}$$

$$v = c_s A\cos(\varphi) \tag{4.1c}$$

$$p = p_0\left(1 + A\cos(\varphi)\right) \tag{4.1d}$$

Where A is the perturbation amplitude, N_{wave} the number of complete wavelengths in the domain (which has been set equal to 4), and φ the phase of the wave. For the analytical solution to be valid, the condition of small amplitude $A \ll 1$ must be fulfilled. The boundary conditions are taken periodic, this means that the solution at a point x should be equal to the solution at $x + D$, where D is the given period. In this case, D is our domain length. Numerically, for every variable ψ, the condition is written in the from: $\psi_0 = \psi_{-2}$ and $\psi_{-1} = \psi_1$, where the subscripts follow the notation given in Sec. 3.3

According to the elementary theory, the sound wave solution is obtained substituting φ with $\varphi - \omega t$ in equations (4.1), with $\omega = kc_s$, $c_s = \sqrt{\gamma p_0/\rho_0}$ and $k = 2N_{wave}\pi/(z - z0)$. Using $p_0 = 1$, $\rho_0 = 1$, we obtain $c_s = 1.29$ and therefore a sound crossing time of 0.775

The Lax-Friedrichs scheme has been found highly diffusive (Figure 4.2). At time $t = 1$, the wave has only traveled $\approx 5.16\lambda$, where $\lambda = \frac{2\pi}{k}$ is the wavelength of the perturbation, but its amplitude has dropped significantly, and there is a big discrepancy between the analytical and numerical solution.

For the Lax-Wendroff Ritchmyer scheme, the results for the same time can be seen in Figure 4.3. This numerical scheme, for the same time, yields a more accurate solution due to its much lower numerical diffusion.

To have a more quantitative appreciation of the results, the variation in time of the perturbation amplitude for both schemes is shown in Figure 4.4. As time increases, it can be easily seen that the Lax-Friedrichs scheme reduces the amplitude of the perturbation almost by a factor 0.4. For the Lax-Wendroff Ritchmyer scheme, however, the decrease is very small, only 2% compared to the original.

As a result of this test we conclude that both schemes are correctly implemented and work properly, except for the high numerical diffusion of the Lax-Friedrichs scheme. Consequently,

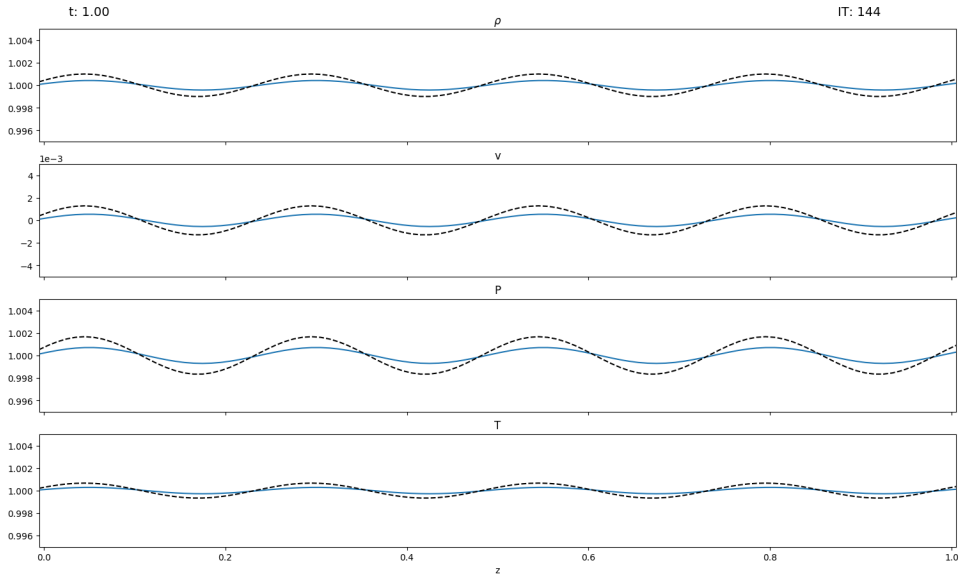


Figure 4.2: Density, velocity, pressure and temperature at $t = 1$ with the Lax-Friedrichs scheme. Blue line, numerical solution. Dashed line, analytical solution. Movie: `4-1-2_SoundWaves_Lax.mp4`

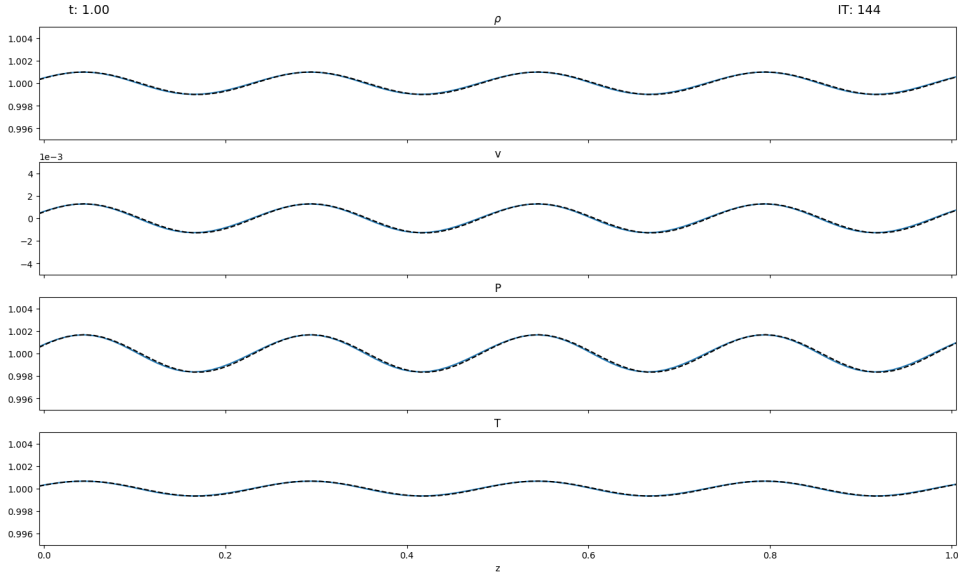


Figure 4.3: Density, velocity, pressure and temperature at $t = 1$ with the Lax-Wendroff Ritchmyer scheme. Blue line, numerical solution. Dashed line, analytical solution. Movie: `4-1-2_SoundWaves_FirstGen.mp4`

the Lax-Wendroff scheme will be used hereafter.

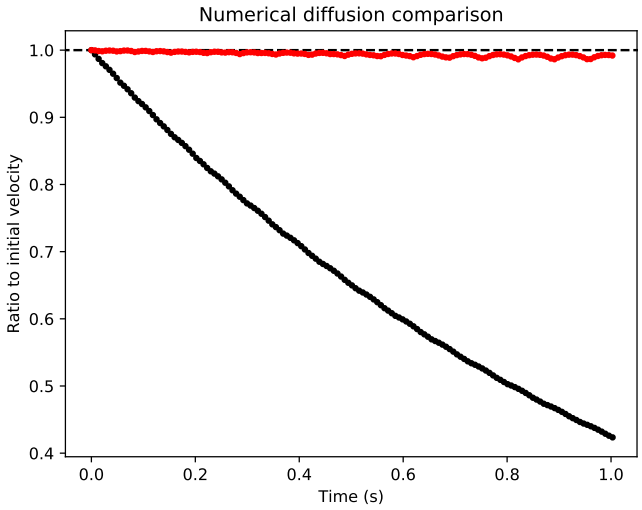


Figure 4.4: Value of the peak velocity of the wave, normalized to the initial condition for both schemes (black, Lax-Friedrichs; red, Lax-Wendroff Ritchmyer)

4.1.3 Isothermal atmosphere

For the next test a stratified ($|g| = 1$) case will be used to check the implementation of the gravity terms (Sec. 3.4.1) and the effectiveness of the momentum damping (Sec. 3.4.4). For simplicity, an isothermal atmosphere will be studied, where the analytical solution can be written as:

$$p = p_0 \exp\left(-z/H\right) \tag{4.2a}$$

$$T = T_0 \tag{4.2b}$$

In (4.2a), $H = \frac{p}{\rho g} = \frac{\mathscr{R} T_0}{\mu g}$ is the pressure scale-height. This solution is implemented as initial condition and its stability is checked. The boundary conditions are indeed crucial, because if they do not fulfill (4.2) they will create non-physical fluid motions. In this case, we choose as boundary conditions the following: *(a)* the temperature is fixed at both extremes, to be consequent with (4.2b); *(b)* for the momentum, a symmetry condition is applied (i.e., the values in the points immediately inside and outside the physical domain are the same). This condition does not force the velocity to be zero at the boundary, thus allowing inflow/outflow of mass. *(c)* For the density, its first derivative is set constant across both boundaries (e.g. numerically for the lower boundary: $\rho_0 = 2\rho_1 - \rho_2$). However, from the analytical solution,

the first derivative of the density ($\frac{\partial \rho}{\partial z} \propto \exp(-z/H)$) is not strictly constant at the boundaries. So the BC for the density that we are imposing is not consistent with the stratified profile, and thus can lead to spurious motions that eventually can evolve into instabilities.

The evolution for a mesh with 1000 nodes can be seen through the movie `4-1-3_Isothermal Atmosphere.mp4`. The instability that causes the simulation to be destroyed can be seen evolving from $t \approx 40$ onwards, when there is an upward movement of the fluid, which leads to an 'ejection' or 'explosion' of the of the medium out of the domain. However this is not a physical effect, but caused by the numerical boundary conditions.

If we wish to keep the same boundary conditions, but to avoid these fluid motions, a damping can be set, to 'control' the behavior of the fluid. For this case, a simple and constant damping has been set that subtracts 0.01% of the velocity at each time step. The evolution is shown in the accompanying movie `4-1-3_IsothermalAtmosphere_Damping.mp4`. Its last frame is shown in Figure 4.5. The damping in this case compensates the tendency of the system to move upwards, and it forces an equilibrium configuration, even though the boundary conditions do not fulfill a hydrostatic condition.

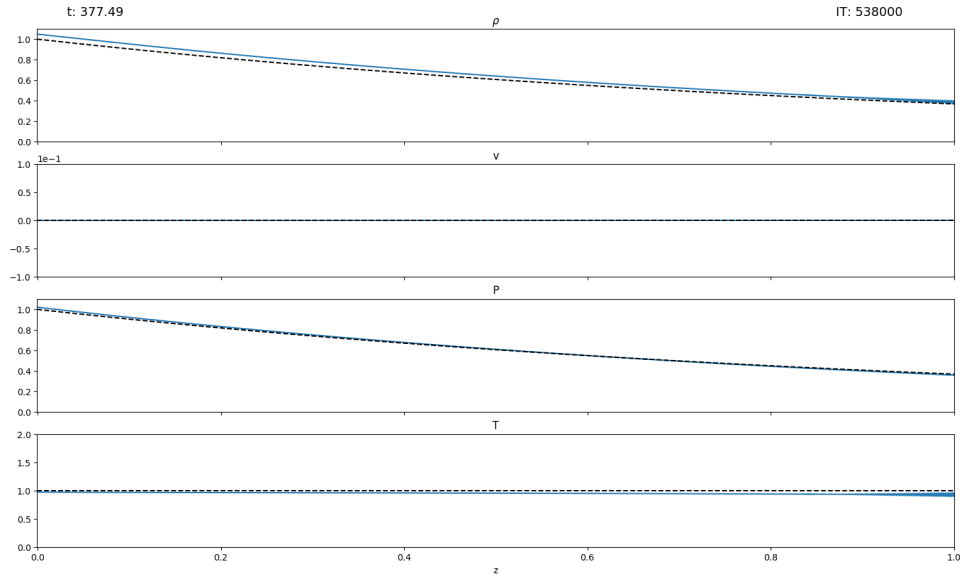


Figure 4.5: Final equilibrium configuration with non-ideal boundary conditions and damping.

4.1.4 Thermal conduction

The last test case will check the influence of the thermal conduction (Sec. 3.4.2) without stratification. Both implicit and explicit schemes for the thermal conduction will be tested and compared.

In order to have the simplest analytical solution, the thermal conductivity, κ, will be set uniform and constant in time. However, the code allows for non-homogeneous conduction, like the Spitzer conduction (1.3). For constant κ, the solution for an initial Gaussian function for the temperature of the form $T(t = t_0) = T_0 + T_1 \exp\left[-(z - z_c)^2/\sigma\right]$, is given by:

$$T(t) = T_0 + T_1 \sqrt{\frac{\sigma}{\kappa t + \sigma}} e^{-\frac{(z-z_c)^2}{4\kappa t + \sigma}} \tag{4.3}$$

For these simulations a uniform grid with 100 nodes is used. The Gaussian is centered at $z_c = 0.5$ and its width is set by $\sigma = 0.0005$. T_0 is set equal to unity and $T_1 = 2$. For the temperature, a zero derivative across the boundaries is set. It is important to notice that for this case, the variables update is done only for the energy (i.e. there are no fluid motions).

As expected, both explicit and implicit schemes correctly match with high accuracy the analytical solution at all times (Figure 4.6), but for the implicit scheme much longer timesteps can be used without losing accuracy. Hence to reach the same physical time, the implicit scheme requires much fewer timesteps. For this case, for the implicit scheme they are approximately 1/3 longer, however, this difference can increase substantially when using a finer mesh (see Sec. 3.4.2).

Even though we will not discuss any analytical solutions for inhomogeneous κ, there are some cases in which a numerical solution of the equation can be found and its validity discussed based on general physical principles. We include here one of those cases: the thermal conduction will be set as a Gaussian function, centered at the center of the domain, and the same width than the initial temperature configuration. For two given times, the solution is shown, with κ, at Figure 4.7. Beforehand, one would expect that, as the thermal conduction is virtually zero on both sides, the temperature must not change there. On the other hand, in the center of the domain, where κ is maximum, a plateau is expected to appear. This will lead to an equilibrium in the temperature profile (but significant changes would appear given a large amount of time).

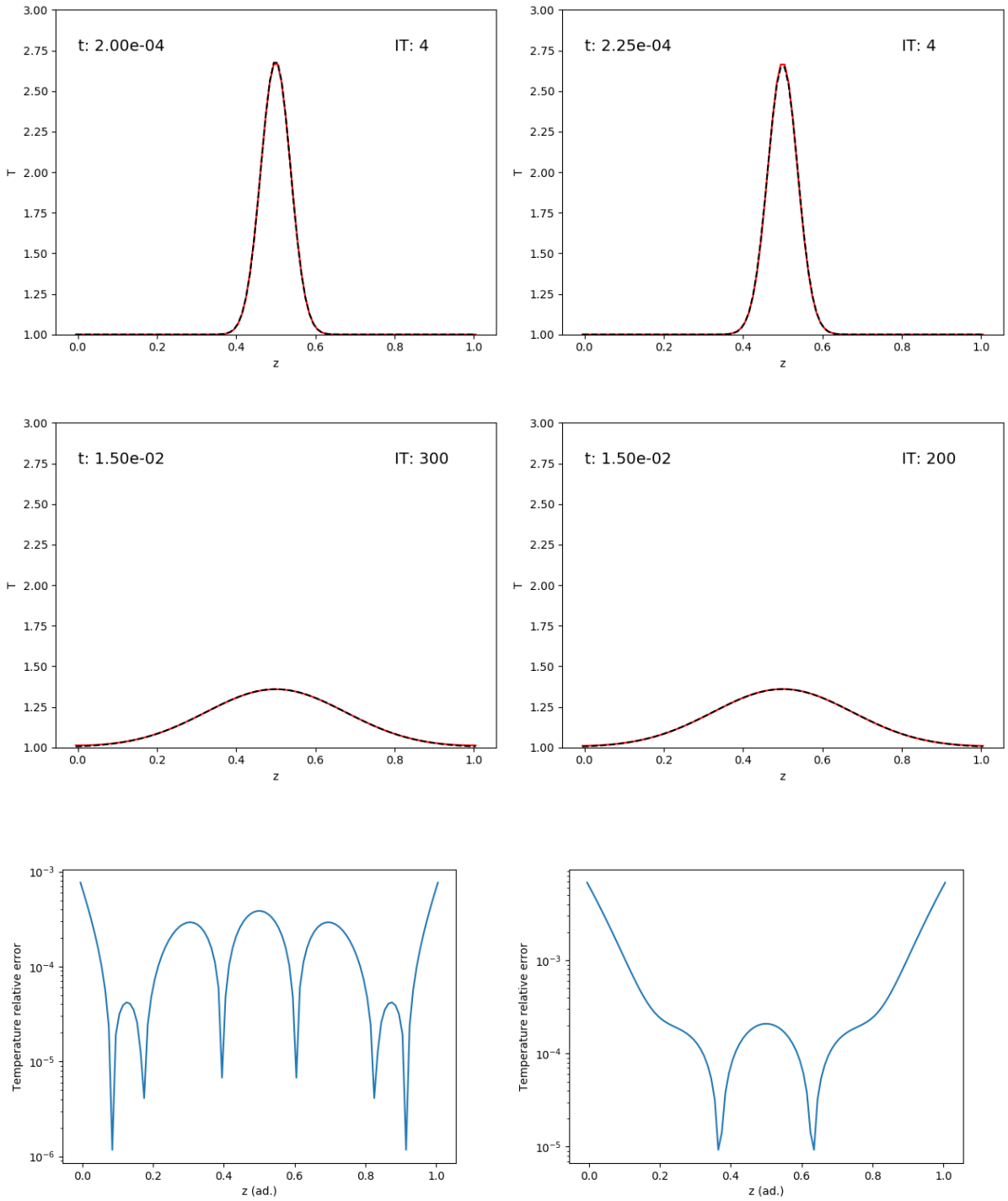


Figure 4.6: Solution for the temperature with the same initial conditions and constant thermal conductivity κ. Using the explicit (left - `4-1-4_ConstantK_Explicit.mp4`) and the implicit (right - `4-1-4_ConstantK_Implicit.mp4`) scheme, for two different times. In solid red, the numerical solution. In black dashed lines, the analytical solution. In the lowest row, the relative errors at time $t = 1.5 \cdot 10^{-2}$ for both schemes.

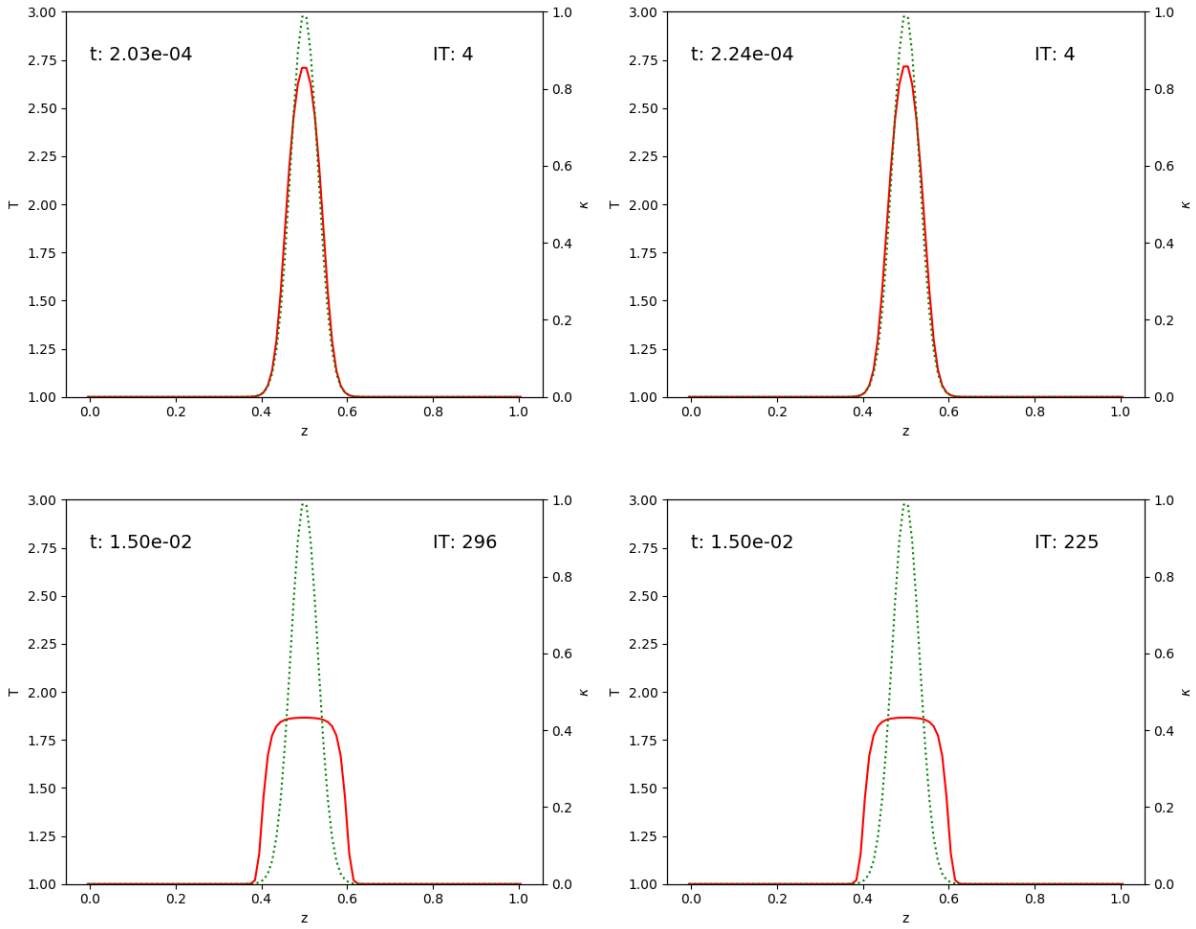


Figure 4.7: Solution for the temperature with the same initial conditions and non-uniform κ. Using the explicit (left - `4-1-4_GaussianK_Explicit.mp4`) and the implicit (right - `4-1-4_GaussianK_Implicit.mp4`) scheme, for two different times. In solid red, the numerical solution. In green dotted lines, κ.

Hereafter, the implicit scheme will be used due to its ability to use a bigger time step, hence speeding up the simulation process.

4.2 Application to a highly idealized *Solar Atmosphere* model.

After introducing and checking the different methods and algorithms in the previous sections, in this one a number of physical cases regarding the solar atmosphere will be studied. In contrast with the previous numerical simulations, now the dimensional EOS is taken, i.e. $p = \rho \mathscr{R} T / \mu$, where $\mathscr{R} = 8.3144598 \cdot 10^7$ erg K^{-1} mol^{-1} and $\mu = 1.1132$. Hereafter the gravity is taken as $|g| = 2.736 \cdot 10^4$ cm s^{-2}.

Firstly (Sec. 4.2.1), an equilibrium without radiative cooling will be studied and compared to an analytical solution. Afterwards (Sec. 4.2.2), we will study how the system evolves toward that equilibrium when the radiative cooling is added. For both simulations, a mesh with 5000 nodes will be used.

4.2.1 Equilibrium without radiative cooling

For this case, we can obtain a simple analytical solution for the equilibrium that can serve to test the validity of the solution calculated with the full code. To that end, we note that, for an equilibrium configuration, the partial derivatives in the fluid equations (1.1), vanish ($\frac{\partial}{\partial t} = 0$). If this equilibrium is also static (ie. $\boldsymbol{u} = 0$) then:

$$\nabla p = -\rho \boldsymbol{g} \tag{4.4a}$$

$$0 = -\vec{\nabla}\left(-\kappa \vec{\nabla} T\right) - L_r \tag{4.4b}$$

For the equations (4.4), if the thermal conductivity κ only depends on the temperature as described in (1.3):

$$\kappa = \mathscr{A} T^{5/2} \tag{4.5}$$

Then the energy equation (4.4b) becomes a ordinary differential equation:

$$\frac{d}{dz}\left(-\mathscr{A} T^{5/2} \frac{dT}{dz}\right) = -L_r \tag{4.6}$$

That can be rewritten as:

$$\frac{d^2 T^{7/2}}{dz^2} = \frac{7 L_r}{2\mathscr{A}} \tag{4.7}$$

For the calculation here we are assuming no radiative losses, i.e., $L_r = 0$, so the right hand side of (4.7) is zero and the ordinary differential equation can be solved as:

$$T(z) = \left[\alpha\left(z + \beta\right)\right]^{2/7} \tag{4.8}$$

With integration constants α and β that can be obtained by specifying the boundary conditions. For a simple problem, we fix the temperature to a constant value at the boundaries

of the domain ($T(z_0) = T_0$ and $T(z_f) = T_f$). Thus the integration constants are:

$$\alpha = \frac{T_0^{7/2}}{z_0 + \beta} \tag{4.9a}$$

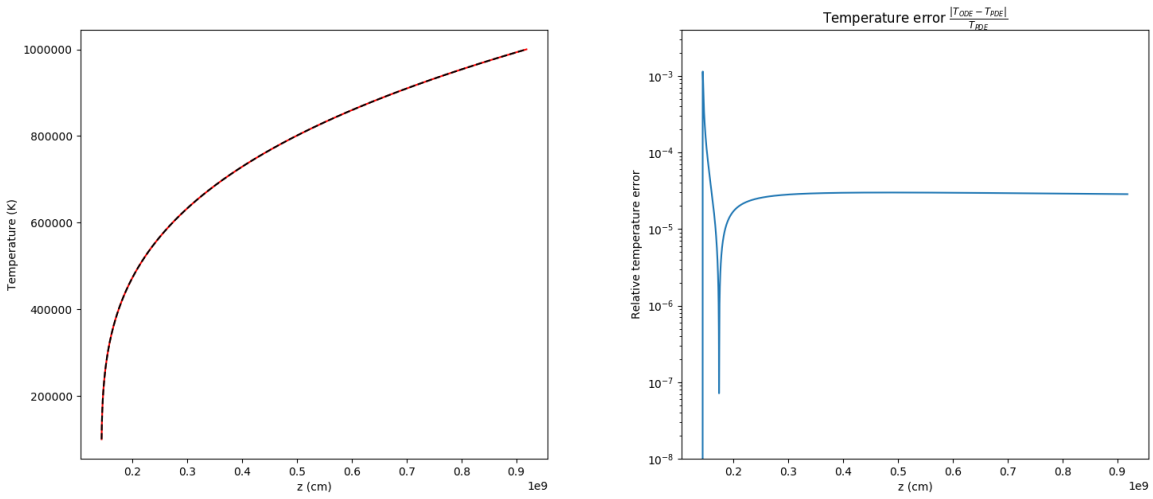
$$\beta = \left[\left(\frac{T_f}{T_0}\right)^{7/2} - 1\right]^{-1} \left[z_f - z_0 \left(\frac{T_f}{T_0}\right)^{7/2}\right] \tag{4.9b}$$

Following the same procedure, and using the obtained temperature profile, the pressure profile can be computed from the hydrostatic equilibrium condition (4.4a). This leads to:

$$\log \frac{p}{p_0} = -\frac{7}{5} \frac{\mu g}{\mathscr{R}} \alpha^{2/7} \left[(z+\beta)^{5/7} - (z_0+\beta)^{5/7}\right] \tag{4.10}$$

We now proceed to obtain the full solution of the problem using the full hydrodynamical code. We first use as initial condition the analytical solution we have just obtained, and then calculate the evolution starting from a different initial condition. If we then first start with (4.8) - (4.10), the profile should be maintained in time, assuming that the BC also satisfy the equilibrium. For this calculation, additionally to fixing the temperatures, we are choosing the following boundary conditions: antisymmetry is imposed for the momentum at both ends of the domain, and the density is set equal to a constant in the lower boundary, and the its derivative across the upper boundary is set equal to zero. The evolution is shown at `4-2-1_AnalyticIC.mp4`. Since the boundary conditions do not exactly satisfy the equilibrium, the fluid moves to another configuration. In this new configuration, the density is lower than the given IC, and thus, also the pressure is lower to maintain the temperature profile (at the lower boundary, the pressure is $\sim 90\%$ of the initial). This reconfiguration happens because the BC for the density at the bottom, does not exactly match the initial density profile. However, this transition is smooth and there is no need to use the momentum damping. The final solution for the temperature can be seen in Figure 4.8, and the relative error in the temperature profile in Figure 4.8b

As we said, now we calculate a more interesting case in which the initial condition is not an equilibrium and let the system evolve toward a full equilibrium including the conduction. For example, a linear logarithm temperature profile can be set, and using the hydrostatic condition (4.4a) to obtain the pressure, the initial conditions can be computed. The code will then tend to modify this profile and drive it towards a full (dynamical and thermal) equilibrium configuration. For this numerical experiment, at both boundaries the momentum is set equal to zero using an antisymmetry condition to avoid a high inflow/outflow of mass at the



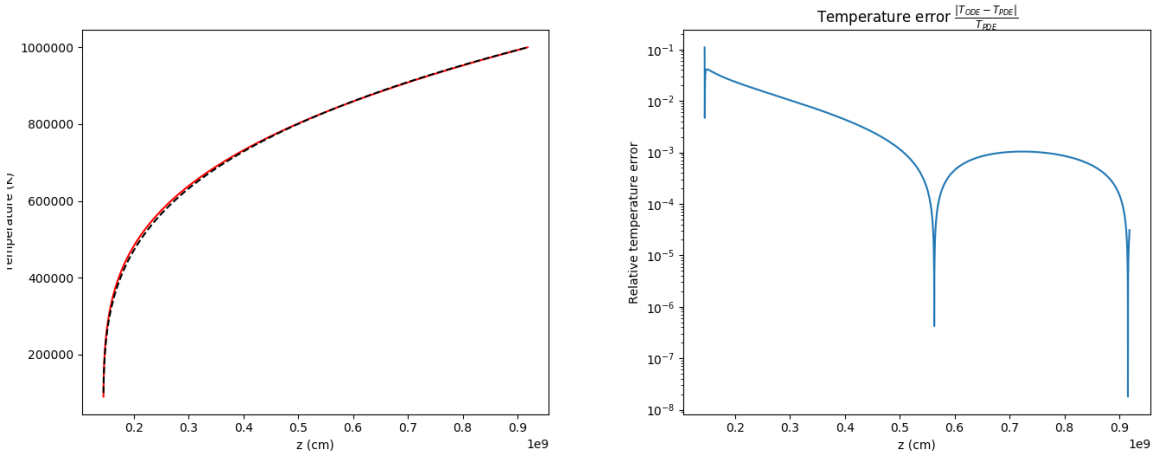
(a) **(b)**

Figure 4.8: Temperature at the equilibrium configuration and its the relative error to the numerical solution of the ODE, with the analytical solution as initial condition. Numerical solution, red line. Analytical solution, dashed black line

boundaries. For the densities, the second derivatives are set constant across the boundaries to allow the system to readjust freely to the temperature profile. Finally, the temperature is fixed at $10^5 K$ and $10^6 K$ at the bottom and top boundaries, respectively.

The evolution can be seen in the accompanying animation `4-2-1_LogTProfileIC.mp4`, and the final equilibrium for the temperature, compared with the analytical solution is showed in Figure 4.9. The steepening of the T profile toward the lower boundary could be expected. The thermal conductivity κ is essential to understand this profile. Due to its dependence with the temperature ($\propto T^{5/2}$), κ changes considerably across the domain. On the other hand, the condition of stationarity ((4.4b) with $L_r = 0$) forces the heat flow to be uniform throughout the domain.Therefore, where the temperature is high, the conductivity is also high and the profile tends to become less inclined; where the temperature is low, the conductivity is low as well and, to maintain a uniform heat flow, the temperature profile must become steep. For this slope, the lower the temperature at the bottom, steeper the profile will be.

As a final note, we would like to mention that for the latter numerical simulation: *(a)* the damping was not needed as the system tended smoothly to the equilibrium configuration, and *(b)*, the total mass inside the physical domain has been found to decrease by a factor 0.9 in the first 1500 seconds (i.e. $\approx$ 11.5 sound crossing times). As the numerical scheme is



(a) **(b)**

Figure 4.9: Temperature at the equilibrium configuration at $t = 2992.9s$ and its relative error to the numerical solution of the ODE, using an hydrostatic equilibrium with linear logarithm temperature as initial condition. Numerical solution, red line. Analytical solution, dashed black line

conservative by itself, this mass decrease is caused by the boundary conditions (although there is a zero momentum boundary condition). This will be discussed in detail in the following section.

4.2.2 Equilibrium with radiative cooling - First approach to the thermal structure of the *Solar Transition Region*

In this section, we finally reach the final objective of this work, namely obtaining a proper equilibrium configuration that captures part of the essence of the temperature profile across the *solar transition region*. This will be done in two steps, i.e., via two successive simulations where we use the final state of the first one as initial condition for the second one, while using different boundary conditions for them.

Similarly to the preceding section, we would like to have an analytical expression to compare with the configuration obtained with the code, in the case when the radiative losses are non-zero. In a general case where L_r does not obey a particular power law, there is no analytical solution to equation (4.7). However, this ODE can be integrated along the domain using elementary numerical methods. Using a centered scheme for the second derivative, leads to:

$$T_{i+1}^{7/2} = (\Delta z)^2 \frac{7L_r}{2\mathscr{A}} - T_{i-1}^{7/2} + 2T_i^{7/2} \tag{4.11}$$

It it important to notice that for the temperature equation (4.7) two boundary conditions should be known to solve the system. For the analytical solution these were the temperature at both sides of the domain. However, for the ODE integration expressed at equation (4.11) these conditions are the temperature at z_0 as well as its derivative at the same point (i.e. T_0 and T_1 known). Thus with this numerical integration we can not fix the temperature at z_f. Although, we can modify the slope to get as close as T_f as we desire. Also, this numerical integration can starts at z_f, and be solved downward.

4.2.2.1 First phase - Non-conservative boundary condition

To begin with, the initial conditions will be set equal to the analytical solution without radiative cooling, i.e. (4.8) and (4.10). The boundary conditions for this case are as follows: like in the previous case, we set an antisymmetry condition for the momentum. Also, constant first derivative for the density is prescribed across the boundary. Here, though, instead of imposing fixed values for he temperature, as adequate for the previous case, a condition of hydrostatic equilibrium is imposed on the pressure:

$$(\log p)_0 = (\log p)_1 - \Delta z \frac{\rho_0 + \rho_1}{2} g \tag{4.12}$$

This condition will allow the temperature at the lower boundary to change, because the radiative cooling will tend to decrease the temperature at the bottom of the domain. If this is not used, then non-physical steep slopes will tend to appear in the lower part of the domain.

The numerical results for this simulation between $t = 0$ and $t = 162371.7s \approx 45h$ can be found in the movie `4-2-2-1_FirstPhase.mp4`. As can be seen, a high density zone develops in the lower part of the domain, and a sharp transition is formed right above it. Physically, the radiative cooling is decreasing the temperature of the initial configuration, and thus the pressure decreases. As the boundary conditions in the lower part do not restrict the temperature, it can decrease. This cooling causes a downward motion of matter to the bottom of the domain because now the lower part of the atmosphere can not longer sustain the rest. A denser region is developed fast, until a high enough density is reached to compensate the drastic decrease of temperature, in an attempt of the system to reach hydrostatic equilibrium. In this cold and dense region, the thermal conduction is negligible due to the low temperatures (its characteristic time is $\sim 10^8 s \sim 3$ years).

As time advances, this denser zone continuously extends upwards, but without reaching

an actual equilibrium configuration. This happens because, in spite of the condition of antisymmetry for the velocity, mass can leak into the domain through the lower boundary. The reason for this is as follows: we can obtain the integrated mass in our physical domain (i.e., the *column density*, in physical or astrophysical terms) at timestep n through the following numerical formula that uses the trapezoidal rule:

$$\xi^n \equiv \Delta z \sum_{i=1}^{N-1} \left(\rho_i^n\right) \tag{4.13}$$

The first and last nodes are not included in the column mass calculation since they are ghost nodes, i.e., they are outside of the physical domain. (see Sec. 3.3). For the Lax-Wendroff Ritchmyer scheme, an intermediate step is carried out at $n+1/2$. This step is given by (3.3). So the column density changes following the law :

$$\xi^{n+1} = \xi^n + \Delta t \left[(\rho u)_{-1/2}^{n+1/2} - (\rho u)_{1/2}^{n+1/2}\right] \tag{4.14}$$

Where the momentum at the $n+1/2$ step is given by:

$$(\rho u)_{i+1/2}^{n+1/2} = \frac{1}{2}\left((\rho u)_{i+1}^n + (\rho u)_i^n\right) - \frac{\lambda}{2}\left[\rho_{i+1}u_{i+1}u_{i+1} + p_{i+1} - \rho_i u_i u_i - p_i\right] \tag{4.15}$$

For a system with global mass conservation, ξ must be a constant in time, i.e., the term in square brackets in (4.14) should vanish. We can simplify (4.15) taking into account the typical values of our problem. If we assume that $\rho \sim 10^{-13}\frac{g}{cm^3}$, $u \sim 10^3\frac{cm}{s}$ and $p \sim 10^{-1}\frac{dyn}{cm^2}$, then the advection term is much smaller than the pressure, i.e. $\rho u u \ll p$. Also, for this problem, it can be seen that the order of magnitude of the pressure difference (derivative) at both boundaries is notably different. We can then disregard the pressure difference at the upper boundary and (4.14) can finally be expressed as:

$$\xi^{n+1} \approx \xi^n + \frac{\Delta t}{2}\left[(\rho u)_{-1}^n + (\rho u)_{-2}^n\right] - \frac{\Delta t}{2}\left[(\rho u)_0^n + (\rho u)_1^n\right] + \frac{\lambda \Delta t}{2}\left[p_1 - p_0\right] \tag{4.16}$$

If we use an antisymmetric condition for the momentum, then only the pressure term remains. That term being proportional to $(\Delta t)^2$ should be comparatively small. However, the non-small value of $\left[p_1 - p_0\right]$ and the large number of timesteps leads to an important increase of the column density as time advances.

In the following section, new boundary conditions will be used continuing the simulation to obtain a proper equilibrium configuration.

4.2.2.2 Second phase - Conservative boundary conditions

In the previous phase the system reached an interesting solution with a very dense stretch of chromospheric nature ($T \sim 2 \cdot 10^4 K$, $\rho \sim 2 \cdot 10^{-13} g\, cm^{-3}$) at the bottom of the domain surmounted by a steep, transition-region like profile in the temperature and a corona (meaning a region with temperature of order $10^6 K$ and low gradient of temperature). However, the system was not stationary: there was a slow but constant inflow of mass into the domain. To reach a final equilibrium we start from the final configuration of that simulation, but we change the lower boundary conditions as follows: we fix the value of the density and switch the antisymmetry condition on the velocity into a condition of symmetry. The condition of hydrostatic equilibrium on the pressure is maintained. The results of this simulation can be seen at `4-2-2-2_SecondPhase.mp4`, the final snapshot at Figure 4.12, and the column density for both simulation phases is shown at Figure 4.10.

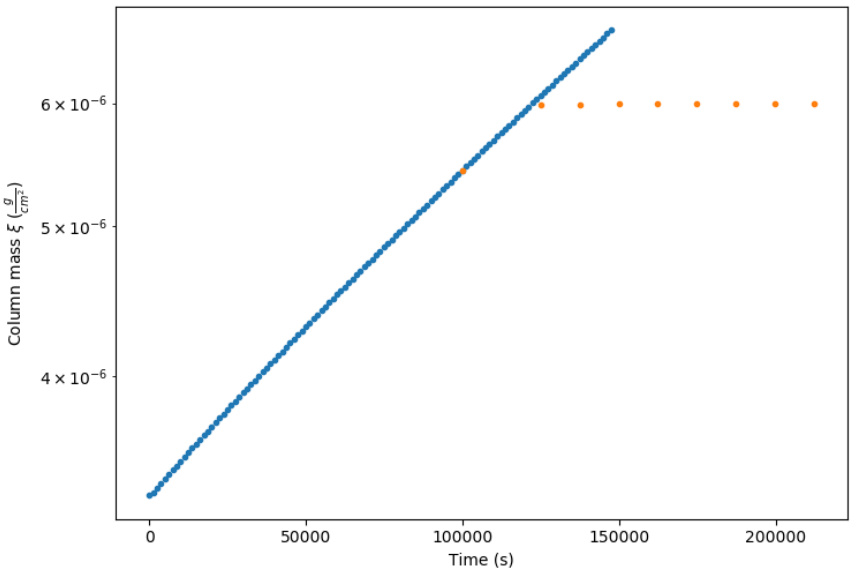


Figure 4.10: Column density for the two phases of the simulation. First phase: blue dots. Second phase: orange dots

Once the equilibrium is reached, the solution can be compared to the integrated ODE given in (4.11), where the integration is carried out downward to avoid big derivatives at the beginning of the numerical integration. This comparison can be seen in Figures 4.11.

The solution obtained with the code matches almost perfectly the equilibrium solution, with relative errors of 10^{-4} for both temperature and pressure. The lower part of the domain

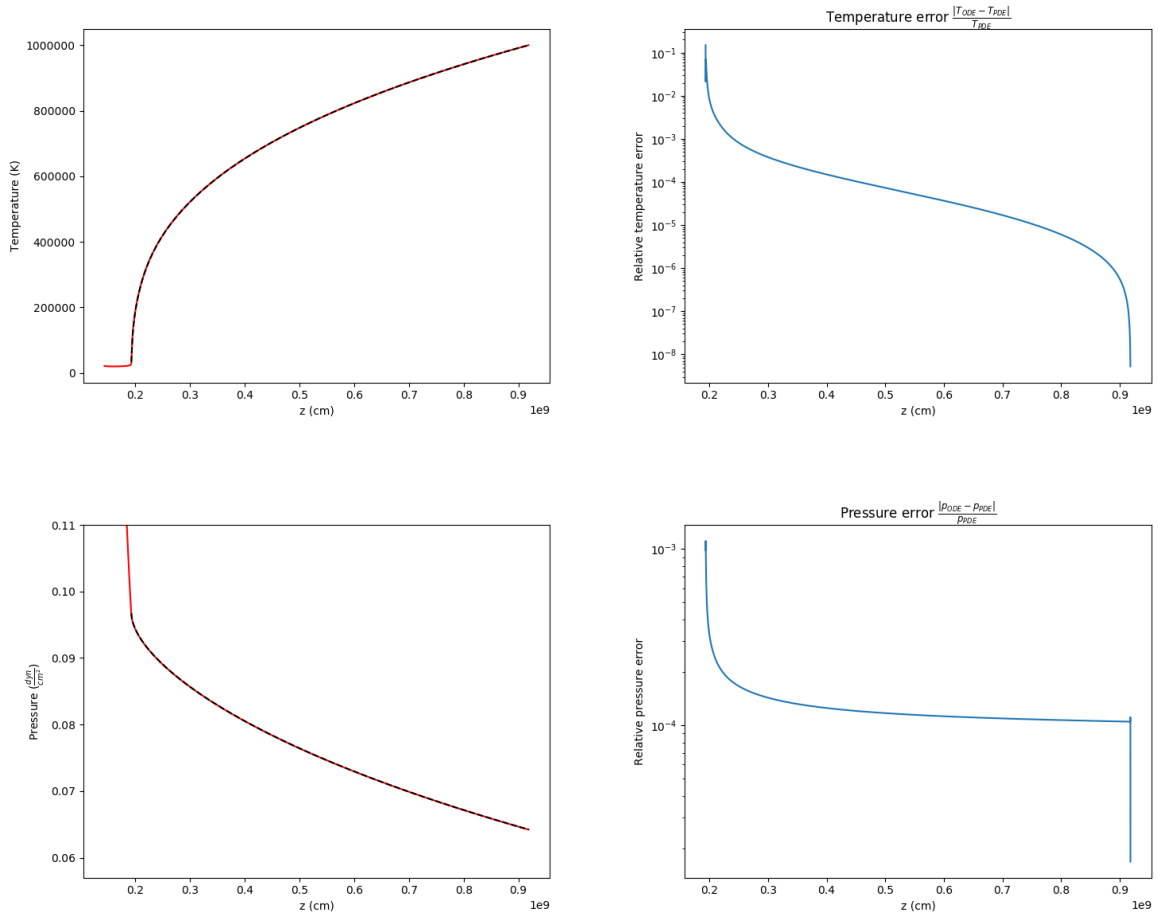


Figure 4.11: Temperature and pressure at the equilibrium configuration and their relative error to the numerical solution of the ODE. Numerical solution, red line. Analytical solution, dashed black line.

is not well represented in the analytical solution due to the high slope in the temperature profile, and therefore is not represented in the Figures 4.11.

Physically, in the lower region, as it is in hydrostatic equilibrium, both density and pressure decrease exponentially with height. This configuration takes up the lowest 1900 km, until the density is low enough to reduce substantially the effectiveness of the radiative cooling. Meanwhile, in the upper part, the radiative cooling is almost negligible, but the thermal conduction is the dominant factor. Due to the high temperatures, κ is also large ($\kappa \propto T^{5/2}$) and this causes that in the upper part the temperature varies only slowly and any perturbation is quickly smoothed due to conduction (the dynamic time is much higher than the conduction time).

Going downwards of $z = 3000$ km, we see that the temperature decreases, and, with it, κ.

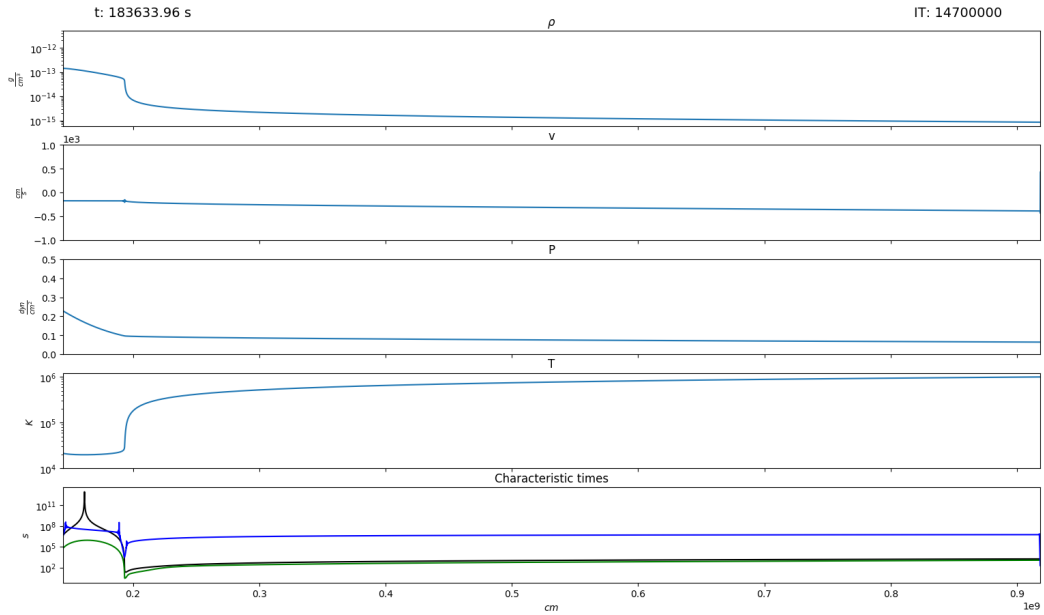


Figure 4.12: Final equilibrium configuration for the second phase of the simulation. For the characteristic times: blue, dynamic time; green, cooling time; black, conduction time.

On the other hand, the radiative cooling becomes more important as the density increases. Following from the thermal equilibrium condition (4.7), the larger the cooling term, the steeper will be the temperature profile, as now the heat flux should compensate for cooling term.

In this region with steep temperature profile, both the density and the temperature change drastically (almost an order of magnitude in just $100km \approx 60\Delta z$). This transition can be a simple description of the thermal structure that connects the *chromosphere* and the *corona* (i.e. the *solar transition region*). Although this is a simple, one-dimensional example, it provides the necessary insight to understand why there is such a steep temperature gradient in the *solar atmosphere*, and why the radiative cooling is a key element to this equilibrium, as well as the sensitive dependence of the thermal conduction coefficient on the temperature.

Chapter 5

Conclusions

En este trabajo, se ha desarrollado un código modular capaz de resolver las ecuaciones de los fluidos. Además ha sido probado con diferentes casos de prueba y ha mostrado resultados exitosos para resolver un equilibrio estático, similar a la estructura térmica de la atmósfera solar, usando distintas condiciones de contorno. Además, ha servido de breve introducción a la Física Solar, en concreto al estudio de la atmósfera del Sol, aunque sea con modelos hidrodinámicos sencillos.

We summarize in the following different conclusions and achievements obtained in the framework of this project:

- We have developed a simple numerical code to solve the gas dynamic equations (1.1) with gravity and non-ideal terms (1.2), namely: thermal conduction, optically thin radiative losses and ad-hoc momentum damping. The code has a modular structure to make it readable and easy to work with. The update for the ideal hydrodynamics equations is done through an explicit scheme, achieving up to second-order in time and space. Two such schemes have been implemented in the code and can be chosen depending on the calculation to be carried out. A variety of possible initial and boundary conditions has been implemented. The operator-splitting technique has been used to deal with the gravity and the non-ideal terms. For the thermal conduction, in particular, both an explicit and an implicit scheme have been developed for the code and installed in it as possible choices.

- The code has been tested using simple cases where there is an analytical solution available to compare. These tests checked different parts of the code to make sure

that they work correctly. It has shown to behave very well for ideal hydrodynamics cases with and without gravity, for cases with thermal conduction (implicit and explicit schemes) as well as for more complex cases that additionally involve radiative losses.

- Application of the code to understand the thermal structure of the solar atmosphere has been carried out. For the numerical simulations, the complicated physics of the solar atmosphere has been strongly simplified and an idealized model has been used, where the dominant terms in the evolution are gravity, thermal conduction (where $\kappa \propto T^{5/2}$) and radiative losses. The final equilibrium can be calculated through solution of a simple ODE when adequate boundary conditions for the temperature and density are given: these simple equilibria have been calculated in advance to check the solutions obtained through dynamical evolution using the code. The numerical simulations allow the the system to evolve and reach (or not) the final equilibrium, depending on the boundary conditions used. Three possibilities have been explored:

  1. Simulation starting both from an initial condition that already fulfills the equilibrium and another one which is far from it. In both cases the equilibrium is reached and/or maintained after the dynamical evolution.
  2. Numerical simulation with radiative cooling, that showed how the temperature profile changes due to radiative losses. The mass inflow through the lower boundary leads to the formation of a chromospheric-like dense and cool region at the bottom of the domain. No static equilibrium is reached.
  3. Finally, a successful static equilibrium was obtained continuing the preceding simulation with different boundary conditions, and both the temperature and pressure profiles match very well the equilibrium profile calculated in advance via solution of an ODE.

  Further work can be done to extend the code carried out here, as well as to better understand the physics of the system under study. There are many extensions that can be attempted, some of which coincide with tools developed in the framework of various courses of the *Masters of Astrophysics* studies (like the *Numerical Simulation Techniques* or *Advanced Programming Techniques* lecture courses). Examples are:

  – Development of a Lagrange tracing algorithm.
  – Extension to two dimensions.
  – Code parallelization using the MPI protocol.

Chapter 6

Bibliography

[Dere et al., 2009] Dere, K., Landi, E., Young, P., Del Zanna, G., Landini, M., and Mason, H. (2009). Chianti - an atomic database for emission lines. ix. ionization rates, recombination rates, ionization equilibria for the elements hydrogen through zinc and updated atomic data. *Astronomy & Astrophysics*, 498:915–929.

[Felipe et al., 2010] Felipe, T., Khomenko, E., and Collados, M. (2010). Magneto-acoustic waves in sunspots: first results from a new three-dimensional nonlinear magnetohydrodynamic code. *The Astrophysical Journal*, 719(1):357.

[Freytag et al., 2012] Freytag, B., Steffen, M., Ludwig, H.-G., Wedemeyer-Böhm, S., Schaffenberger, W., and Steiner, O. (2012). Simulations of stellar convection with co5bold. *Journal of Computational Physics*, 231(3):919–959.

[Gudiksen et al., 2011] Gudiksen, B. V., Carlsson, M., Hansteen, V. H., Hayek, W., Leenaarts, J., and Martínez-Sykora, J. (2011). The stellar atmosphere simulation code Bifrost - code description and validation. *Astronomy & Astrophysics*, 531:A154.

[Laney, 1998] Laney, C. B. (1998). *Computational gasdynamics*. Cambridge university press.

[Lani et al., 2013] Lani, A., Villedieu, N., Bensassi, K., Kapa, L., Vymazal, M., Yalim, M. S., and Panesi, M. (2013). COOLFluiD: an open computational platform for multi-physics simulation and research. In *AIAA 2013-2589*, San Diego (CA). 21th AIAA CFD Conference.

[Lax, 1959] Lax, P. D. (1959). *Systems of conservation laws*. Los Alamos Scientific Laboratory of the University of California,.

[LeVeque, 1992] LeVeque, R. J. (1992). *Numerical methods for conservation laws*. Springer Science & Business Media.

[Maneva et al., 2017] Maneva, Y. G., Laguna, A. A., Lani, A., and Poedts, S. (2017). Multi-fluid modeling of magnetosonic wave propagation in the solar chromosphere: Effects of impact ionization and radiative recombination. *The Astrophysical Journal*, 836(2):197.

[Priest, 2014] Priest, E. (2014). *Magnetohydrodynamics of the Sun*. Cambridge University Press.

[Spitzer, 1962] Spitzer, L. (1962). Physics of fully ionized plasmas. *J. Wiley and Sons, New York*.

[Vögler et al., 2005] Vögler, A., Shelyag, S., Schüssler, M., Cattaneo, F., Emonet, T., and Linde, T. (2005). Simulations of magneto-convection in the solar photosphere. equations, methods, and results of the muram code. *Astronomy & Astrophysics*, 429:335–351.