UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
Grau en Enginyeria Física

Treball Final del Grau en Enginyeria Física

Three-dimensional hydrodynamic simulations of dynamos in white dwarfs

Joan Marco Rimmek

Co-director Dr. Pablo Lorén Aguilar
Department of Physics and Astronomy
University of Exeter

Co-director Dr. Jordi José Pont
Departament de Física
Universitat Politècnica de Catalunya

January 29, 2018

Joan Marco Rimmek
Three-dimensional hydrodynamic simulations of dynamos in white dwarfs
Treball Final del Grau en Enginyeria Física
January 29, 2018
Thesis Co-directors: Dr. Pablo Lorén Aguilar and Dr. Jordi José Pont

Universitat Politècnica de Catalunya
Grau en Enginyeria Física
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
Campus Diagonal Nord, C/Jordi Girona, 1
08034 Barcelona

Abstract

The origin of magnetic white dwarfs is still a controversial issue. When white dwarfs cool enough, they crystallize. This causes a phase separation of its main constituents in the nucleus, leaving a less dense liquid behind which is redistributed by Rayleigh-Taylor instabilities. In a recent paper (Isern et al. 2017) it was shown that this configuration can produce magnetic fields of strengths up to $0.1MG$, and thus, could explain magnetism in single white dwarfs. However, realistic simulations of this scenario are still lacking. The main goal of this work consists of performing these computations. For that purpose, the project makes use of the freely distributed magneto-hydrodynamical code Athena to simulate the convective region of a white dwarf.

Acknowledgement

First of all, I would want to dedicate this work to the memory of Enrique García Berro. He was a great teacher, passionate about his work. He moved heaven and earth to find the right place for me to carry out this project.

Secondly, I would like to thank Pablo Lorén Aquilar and Jordi José Pont, the directors of my work, for their patience, help and, detailed corrections.

Last, but not least, I would like to express my thanks to Patsy Lang, who welcomed me into her home and allowed me to have a comfortable stay during my months in Exeter.

Contents

1 Theory on White Dwarfs

White dwarfs are the most common end-point of stellar evolution...

— **Every paper on white dwarfs**
Ever.

White dwarfs are celestial objects with sizes comparable to Earth but masses between $0.17\ \mathrm{M}_\odot$ (Kilic et al. 2007) and $1.33\ \mathrm{M}_\odot$ (Kepler et al. 2007), i.e. of the same order of magnitude as the mass of the Sun. With such a huge amount of mass in so little volume, densities are extremely high, and interesting physical phenomena arise.

1.1 Formation

White dwarfs are believed to be the final evolutionary stage of 97% of all the main sequence stars (e.g. Althaus et al. 2010). That is, all the stars that don't end up as neutron stars or black holes.

Clouds of hydrogen, helium and dust in the interstellar medium collapse by gravitational attraction. High pressures and temperatures allow for the fusion of hydrogen at the nucleus and therefore the formation of stars. The fate of newly formed stars depends on their masses. Low-mass stars (less than $0.5\ \mathrm{M}_\odot$) are fully convective and therefore have access to all of the hydrogen of the star. However, once all the hydrogen is consumed, the star is unable to fuse helium. With no nuclear fusion in its core the star shrinks due to the gravitational pull and becomes a helium white dwarf.

Intermediate-mass stars (between $0.5\ \mathrm{M}_\odot$ and $8\ \mathrm{M}_\odot$) are not fully convective. Therefore, when hydrogen at the core is depleted the nucleus contracts due to the gravitational pull and as the pressure increases, so does the temperature. With high enough temperatures, the region surrounding the nucleus will start to fuse hydrogen. Moreover, with these higher temperatures, the star expands (as would any gas when heated up) and becomes a red giant. As it grows in size the outer layers feel less gravitational pull and are blown away by radiation pressure.

In the meantime, the nucleus keeps contracting until it starts to fuse helium into carbon and oxygen. Because of its mass, the star will not be able to fuse carbon into neon and therefore, once all the helium at the core is used up, the star will contract due to its gravitational pull, becoming a carbon-oxygen white dwarf.

Stars of masses between $8\ M_\odot$ and $10\ M_\odot$ follow the same path as intermediate-mass stars, but don't stop at carbon and oxygen. They repeat the process and fuse carbon into neon and become oxygen-neon white dwarfs.

High-mass stars (larger than $10\ M_\odot$) are able to fuse neon into iron and end their lives in a supernova Type II explosion leaving either a neutron star or a black hole (again, depending on the initial mass).

1.2 Structure

Since there is not enough heat in white dwarfs for nuclear fusion to occur, there has to be another process through which hydrostatic equilibrium is maintained. This physical process was explained in 1926 using the then new field of quantum mechanics (Fowler 1926). In a nutshell, since electrons are fermions, they follow the Pauli exclusion principle, which states that there can only be one fermion for each quantum state. Therefore, even at zero temperature, there will be electrons occupying states of higher energy than the ground state because, unlike bosons, they can not all occupy the lowest energy states only. That is why white dwarfs are called degenerate.
As a consequence of this, when a white dwarf compresses, the amount of electrons increases per unit volume increasing in turn their kinetic energy which yields a higher pressure. This is called electron degeneracy pressure and that is what supports the white dwarf against gravitational collapse.

Another consequence of electron degeneracy is that white dwarfs are almost isothermal. Heat is transferred very efficiently by electrons and therefore only a small gradient in temperature is required to transport the energy flux. An interesting fact about white dwarfs is that as their mass increases, their radius decreases.

The chemical composition of a white dwarf depends on its mass. If its mass is lower than $0.45\ M_\odot$ it will be completely made of helium. If its mass is between $0.45\ M_\odot$ and $1.1\ M_\odot$ the chemical composition will be about 40% carbon and 60% oxygen. White dwarfs more massive than $1.1\ M_\odot$ are made of a mixture of oxygen and neon (Ritossa et al. 1996).

1.3 Evolution

White dwarfs have energy in the form of stored heat, therefore their evolution can be described as a simple gravothermal process (Althaus et al. 2010). In 1960 it was predicted that in the late stage of cooling, white dwarfs would crystallize (Abrikosov 1960; Kirzhnits 1960; Salpeter 1961). That is, that at the centre of the star, the plasma solidifies to an oxygen rich, body-centered cubic lattice. (Barrat et al. 1988). When white dwarfs undergo this solidification, latent heat is released, providing a source of thermal energy which delays the cooling of the star (Van Horn 1968; Lamb & Van Horn 1975).

1.4 Magnetism

Some white dwarfs have magnetic fields ranging from 10^3 to 10^9 G (e.g. Ferrario et al. 2015). However, there is a scarcity of white dwarfs with magnetic fields between 10^5 and 10^6 G, which suggests that magnetic white dwarfs exhibit a bimodal distribution: a high field population (of fields ranging from 10^6 to 10^9 G) and a low field one (lower than 10^5 G).

It has not been possible to completely explain the origin of such magnetic fields yet. There are currently 3 explanations for the source of magnetic fields in white dwarfs. The first explanation, is that the magnetic field comes from their progenitors in two different ways: Either from magnetic main-sequence stars whose field is conserved due to magnetic flux conservation (Wickramasinghe & Ferrario 2005).
Or from field generation during the helium-burning phase through dynamo action (Levy & Rose 1974).

The second explanation considers binary systems and their evolution. In this scenario the field is generated by a dynamo which can be generated either in the common envelope phase (Tout et al. 2008; Nordhaus et al. 2011) or in the hot corona produced during the merger of the two white dwarfs (García-Berro et al. 2012).

The third explanation is that the field is created in the outer convective envelope of single white dwarfs, formed during their evolution. These fields however, can only reach magnitudes of up to $0.01 MG$ (Fontaine et al. 1973), much lower than the ones observed and thus this mechanism is not efficient enough.

Since the last one is not valid and the first and second are unable to reproduce the amount of magnetic white dwarfs observationally found (Ferrario et al. 2015), another mechanism is needed to explain the creation of magnetic fields in white dwarfs.

In order to explain this fourth mechanism, we have to consider the physical processes that modify the internal chemical profiles of white dwarfs during their evolution. There are two: At high luminosities, gravitational settling of neutron-rich species in the liquid phase (Bravo et al. 1992; Bildsten & Hall 2001; Garcia-Berro et al. 2010; Camisassa et al. 2016). And, at lower luminosities, phase separation upon crystallization (Isern et al. 1997, 2000; Garcia-Berro et al. 2010).
In both cases, the energy involved is large, namely $2 \cdot 10^{46} erg$.

As mentioned in the previous section, the chemical composition of white stars between $0.45 M_{\odot}$ and $1.1 M_{\odot}$ will be approximately 40% carbon and 60% oxygen. The phase diagram of the carbon-oxygen mixture is of the azeotrope form (Horowitz et al. 2010). Therefore, when white dwarfs crystallize, oxygen is more abundant in the solid phase and a less dense carbon-rich liquid is left behind which is then redistributed by Rayleigh-Taylor instabilities (Mochkovitch 1983; Isern et al. 1997, 2000).

A solid core surrounded by a convective mantle driven by compositional buoyancy is a similar configuration to that found in Earth's core, where the light element release due to the growth of the inner core is a primary driver of the dynamo (Lister & Buffet 1995). The scaling law relating the magnetic fields of Earth, Jupiter, T Tauri and M dwarf stars predicts that the maximum field that can be generated at the top of the dynamo ranges from $5 \cdot 10^4$ to $2.5 \cdot 10^5$ G and thus this mechanism could explain white dwarfs with magnetic fields of magnitudes of less than 10^5 G (Isern et al. 2017).

2 Objectives

The real historical plural of 'dwarf' is dwarrows anyway: rather a nice word, but a bit too archaic. Still I rather wish I had used the word dwarrow.

— **J. R. R. Tolkien**
15-10-1937 in a letter to Sir Stanley Unwin

In this work, we have mainly two objectives: On one hand we want to see the effect on the strength of convection for different shapes of gravitational field. On the other hand, we want to study the strength of convection due to phase separation through crystallization in white dwarfs of different masses. In both cases we will start the simulations in an unstable equilibrium and observe how it evolves. This unstable equilibrium is created by introducing a sudden jump in density at $r = 0.75R_{wd}$.

2.1 Gravity

As mentioned before, for the gravity we will try two different approximations. In a polytrope, the density is given by the Lane-Emden equation:

$$\frac{1}{\xi^2}\frac{d}{d\xi}\left(\xi^2\frac{d\theta}{d\xi}\right) + \theta^n = 0 \tag{2.1}$$

where the density is given by $\rho = \rho_c\theta^n$ and ρ_c is the central density. And ξ is related to the radius by $r = \alpha\xi$ where $\alpha^2 = (n+1)K\rho_c^{\frac{1}{n}-1}/4\pi G$. This equation however, has only exact solutions for $n = 0, 1, 5$. For the case we are considering, namely $n = 3$, it can only be approximated. An analytic approximation for the solution can be found in (Liu 1996). Using it, we can obtain the density profile and with it, the gravitational field. In many papers, though, for the sake of simplicity, a constant field approximation is done.

In figure 2.1, we have made a plot of the realistic shape of the gravitational field found by the approximate solution to the Lane-Emden equation, the gravitational field generated by a point source at $r = 0$, and a constant gravitational field with a value given by the field felt at $r = 0.75R_{wd}$.

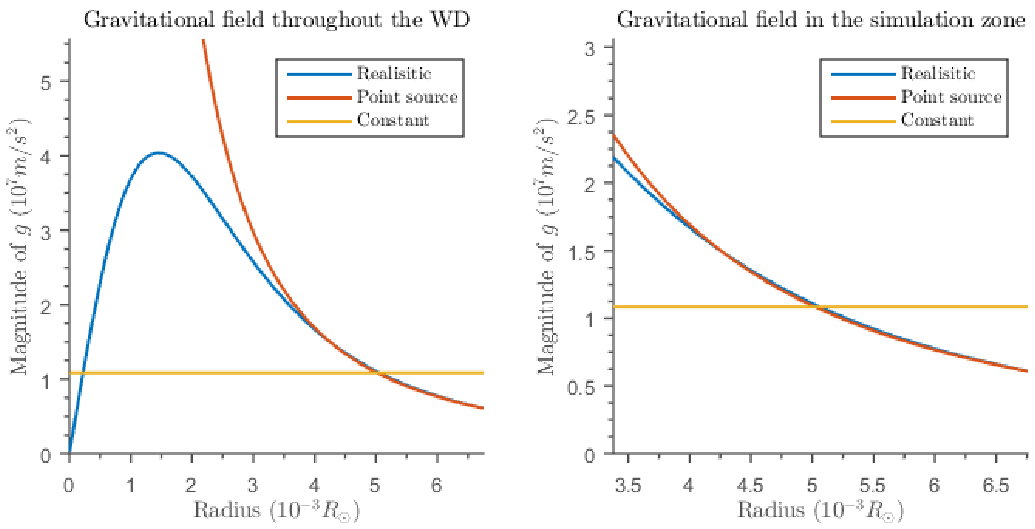

Fig. 2.1: Gravitational field approximations: through an analytic approximation of the Lane-Emden equation (blue), assuming all the mass is accumulated at $r = 0$ (red) and assuming a constant field equal to the field strength at $r = 0.75R_{wd}$ (yellow). Throughout the star (left) and from $0.5R_{wd}$ to R_{wd} (right)

Although these fields are very different near the centre of the star, in the region where we are carrying out the simulations we can see that the point source gravitational field is reasonably similar to the realistic one. Since the point source gravitational field is much easier to implement, we will use it as an approximation to the realistic field in the convection zone. However, we will run some simulations with both a constant and a realistic gravitational field, in order to check the validity of the approximation.

2.2 Mass and composition

In order to prove/disprove the theoretical results obtained in Isern et at. 2017, that is, to see whether the intensity of the magnetic field follows the scaling law that relates the magnetic fields of Earth, Jupiter, T Tauri and M dwarf stars, we will create multiple setups varying the mass of the white dwarf and the height of the jump in density that creates the instability. In this work however, only the results for a single value for the jump in density ($\Delta\rho/\rho = 0.1$) are shown since at the time of delivery of this document simulations for $\Delta\rho/\rho = 0.01$ are still running.

If certain conditions are met (the dynamo is saturated, the magnetic Reynolds number is large enough, and convection is described by the mixing length formalism), the magnetic field of a star will be related to the energy of the convective mantle by the following equation, found in Christensen 2010.

$$\frac{B^2}{2\mu_0} = c f_\Omega \frac{1}{V} \int_{r_i}^{r_o} \left[\frac{q_c(r)\lambda(r)}{H(r)} \right]^{2/3} \rho(r)^{1/3} 4\pi r^2 dr \qquad (2.2)$$

where μ_0 is the permeability of vacuum, c is an adjustable constant, f_Ω is the ratio of the Ohmic dissipation to the total dissipation, V is the volume encompassed by the convective region, and the integral is the aforementioned energy of the convective mantle.

Since we are looking for a qualitative result, we will measure the strength of the convection as a qualitative measure of the magnetic field, and asses whether a change in the mass of the star yields a concurrent change in its magnetic field.

Setup 3

In the different setups we will run, we will change the mass of the star and the shape of gravity. Here we show the software and hardware we used, and the calculations needed to deduce the initial conditions. The simulations are carried out with the open source code Athena (Stone et al. 2008) which is programmed in C.

3.1 Software

As mentioned before, we will be using the Athena Code in C in order to perform the simulations. The simulations will be run in parallel using OpenMPI, which is an open source set of libraries that allows us to run the simulations in multiple cores at the same time, reducing considerably the time needed to carry them out.

In order to visualize the results I used VisIt, an open source visualization tool, the appearance of the visualizations can be seen in figure 3.1. VisIt allows us to see the data, but is not very good for quantitative comparisons. Therefore, we used gnuplot to quickly plot the values we wanted to compare and MATLAB to make the more elaborate plots shown in this document.

All the previously mentioned software can be found at:

Athena:	`https://princetonuniversity.github.io/Athena-Cversion/`
OpenMPI:	`https://www.open-mpi.org/`
VisIt:	`https://wci.llnl.gov/simulation/computer-codes/visit`
gnuplot:	`http://www.gnuplot.info/`
MATLAB:	`https://www.mathworks.com/products/matlab.html`

3.2 Hardware

Low resolution simulations were run in my laptop using a single core. Intermediate resolution simulations where done using eight computation nodes of the Atria UPC computer cluster. Specifically three computers with Intel Core i7 CPUs at $3.07 GHz$ and five computers with Intel Xeon CPUs at $3.10 GHz$. Finally, the more

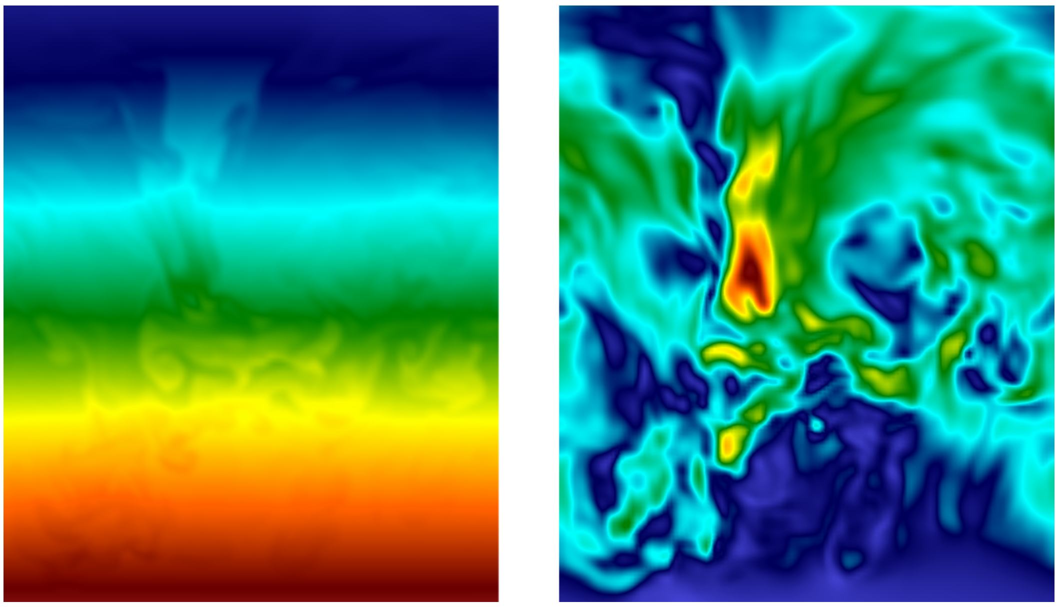
Fig. 3.1: Visualization of a simulation using VisIt. Density (left) and momentum magnitude (right).

computationally expensive simulations with very high resolutions were run in a supercomputer at the University of Exeter.

3.3 Configuration of the solver

The default configuration of athena solves the equations of compressible, adiabatic, inviscid, ideal magnetohydrodynamics (Stone et al. 2008):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{3.1}$$

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v} - \mathbf{B}\mathbf{B} + \mathrm{P}^*) = 0 \tag{3.2}$$

$$\frac{\partial \mathrm{E}}{\partial t} + \nabla \cdot [(\mathrm{E} + \mathrm{P}^*)\mathbf{v} - \mathbf{B}(\mathbf{B} \cdot \mathbf{v})] = 0 \tag{3.3}$$

$$\frac{\partial \mathbf{B}}{\partial t} - \nabla \times (\mathbf{v} \times \mathbf{B}) = 0 \tag{3.4}$$

$$\mathrm{P}^* = \mathrm{P} + \frac{\mathbf{B} \cdot \mathbf{B}}{2} \tag{3.5}$$

$$\mathrm{E} = \frac{\mathrm{P}}{\gamma - 1} + \frac{\rho(\mathbf{v} \cdot \mathbf{v})}{2} + \frac{\mathbf{B} \cdot \mathbf{B}}{2} \tag{3.6}$$

These equations are written in units such that the magnetic permeability $\mu = 1$. Eq. 3.1 is the continuity equation (conservation of mass), eq. 3.2 is the momentum conservation equation, eq. 3.3 is the energy conservation equation, eq. 3.4 is the induction equation, eq. 3.5 yields the total pressure (gas plus magnetic) and eq. 3.6 is the total energy density (for an ideal gas).

In our simulations, we have not calculated the magnetic field $\mathbf{B}$ and therefore the equations become much simpler: the induction equation and all the terms that depend on the magnetic field are dropped.

Adding a term for the effect of gravity in equation 3.2 we arrive at the final set of equations that the modified solver will apply:

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

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v} + \mathrm{P}) = \rho \mathbf{g} \tag{3.8}$$

$$\frac{\partial \mathrm{E}}{\partial t} + \nabla \cdot [(\mathrm{E} + \mathrm{P})\mathbf{v}] = 0 \tag{3.9}$$

$$\mathrm{E} = \frac{\mathrm{P}}{\gamma - 1} + \frac{\rho(\mathbf{v} \cdot \mathbf{v})}{2} \tag{3.10}$$

3.4 Geometry

We have carried out the simulations on a cartesian box geometry. The coordinates are set in such a way that the point $(x, y, z) = (0, 0, 0)$ is the centre of the box. The Z component represents the height, whilst the X and Y components represent depth and width respectively. The point $(x, y, z) = (0, 0, 0)$ is positioned at $r = 0.75R_{wd}$ and the length of the box in the Z direction is $0.5R_{wd}$ so that the box starts at $0.5R_{wd}$ and ends at R_{wd}. The box will have a width of one half the height, i.e. $0.25R_{wd}$ and a depth of one tenth of the height, i.e. $0.05R_{wd}$. The geometry is discretised in tiny cubes, and the equations are solved in each of them.

The number of cells (cubes) defines the processing time needed to run a simulation. Therefore we used lower resolutions in the Atria computer cluster than in Exeters supercomputer. In the computer cluster we run a single simulation in each node with a resolution of $(\#_x, \#_y, \#_z) = (26, 128, 256)$, yielding a total of $851\,968$ cells. In the supercomputer we run simulations with a much higher resolution of $(\#_x, \#_y, \#_z) = (102, 512, 1024)$, resulting in a total of $53\,477\,376$ cells.

3.5 Units and constants

We made the following choice regarding the units to be used in the code:

Magnitude			Unit in code		
Mass	$[M]$	$\rightarrow$	$1M_\odot$	$=$	$1.989 \cdot 10^{30}\, kg$
Distance	$[L]$	$\rightarrow$	$0.1R_\odot$	$=$	$6.957 \cdot 10^{7}\, m$
Time	$[T]$	$\rightarrow$			$50\, s$
$\frac{\text{Energy}}{\text{Volume}}$	$[M][L]^{-1}[T]^{-2}$	$\rightarrow$			$1.144 \cdot 10^{19}\, J \cdot m^{-3}$

We also have calculated the unit in code for the energy by unit of volume because those are the units in which the code outputs the total energy.

Using this set of units, some things simplify. For example the gravitational constant:

$$G = 6.674 \cdot 10^{-11} \frac{m^3}{kg \cdot s^2} \approx 1 \; \frac{(0.1R_\odot)^3}{M_\odot \cdot (50s)^2}$$

The sound speed inside a white dwarf is around $3 \cdot 10^8 cm/s$ which becomes about:

$$C_s = 3 \cdot 10^8 cm/s \approx 2 \; \frac{0.1R_\odot}{50s}$$

We have chosen the radius for all the stars in our simulations to be $4.7 \cdot 10^8 cm$ as in (Isern et al. 2017) which is $0.0676(0.1R_\odot)$ in the code. The interface of our simulation is situated at $0.75R_{wd}$ and therefore at $0.05(0.1R_\odot)$.

3.6 Initial conditions

3.6.1 Constant gravitational field

The field intensity for the constant field will be given by $g = \frac{G\mathrm{M}_{wd}}{z_0^2}$. Where z_0 is the distance between the centre of the star and the centre of the box. In code units $G = 1$ and the expression simplifies to $g = \mathrm{M}_{wd}/z_0^2$. We use a gravitational potential of the form $\Phi = gz$ to have a gravitational field of:

$$\mathbf{g} = -\nabla\Phi = (0, 0, -g) \tag{3.11}$$

In order to simulate the conditions inside a white dwarf we set the initial pressure to be given by a polytrope of polytropic index $n = 3$ and constant of proportionality A, i.e. by:

$$P = A\rho^{(n+1)/n} = A\rho^{4/3} \tag{3.12}$$

We set the value of γ to be $4/3$. As an initial approach, we will take A as a step function which in turn, will create a sudden change in density. From now on, we will call the upper region a and the lower region b and the respective values of A as A_a and A_b. We will obtain the initial density profile by imposing hydrostatic equilibrium, the speed of sound at the interface for the fluid on top and, pressure continuity.

First of all, hydrostatic equilibrium has to be satisfied:

$$\nabla \mathrm{P} = \rho \mathbf{g} \tag{3.13}$$

Which, with a gravitational field given by eq. 3.11, yields:

$$\frac{d\mathrm{P}}{dz} = -g\rho \tag{3.14}$$

Using the expression for the pressure given by eq. 3.12, and a constant value for A in each region, we obtain:

$$\frac{d\rho}{\rho^{2/3}} = -\frac{3}{4}\frac{g}{A}dz \tag{3.15}$$

Which can be integrated in region b:

$$\int_{\rho_b(z)}^{\rho_b(z_i)} \frac{d\rho}{\rho^{2/3}} = \int_{z}^{z_i} -\frac{3}{4}\frac{g}{A_b}dz \tag{3.16}$$

Which, when choosing a system of reference where $z_i = 0$, yields a density of:

$$\rho_b(z) = \left(\rho_b(0)^{1/3} - \frac{gz}{4A_b}\right)^3 \tag{3.17}$$

In an analogous manner, the density in region a can be calculated and results:

$$\rho_a(z) = \left(\rho_a(0)^{1/3} - \frac{gz}{4A_a}\right)^3 \tag{3.18}$$

Therefore, if we calculate the values of $\rho_a(0)$ and $\rho_b(0)$ the density profile will be uniquely determined. In order to do so, we impose the sound speed in the higher fluid at the interface. The speed of sound is given by $C_s = (\mathbf{g}H_p)^{1/2}$ where H_p is the pressure scale height:

$$H_p = \left(\frac{1}{P}\frac{dP}{dr}\right)^{-1} \tag{3.19}$$

Using the equation of pressure and the condition of hydrostatic equilibrium, we find that the pressure scale height at the interface and for the upper liquid is:

$$H_{p_a}(0) = \frac{A_a \rho_a(0)^{1/3}}{-g} \tag{3.20}$$

Finally we have:

$$C_s^2 = -gH_{p_a} = A_a \rho_a(0)^{1/3} \tag{3.21}$$

And thus:

$$\rho_a(0) = \left(\frac{C_s^2}{A_a}\right)^3 = C_s^6 A_a^{-3} \tag{3.22}$$

Lastly, to obtain the value of $\rho_b(0)$ we impose continuity of pressure at the interface:

$$P_a(0) = P_b(0) \tag{3.23}$$

$$A_a \rho_a(0)^{4/3} = A_b \rho_b(0)^{4/3} \tag{3.24}$$

$$\rho_b(0) = \rho_a(0)\left(\frac{A_a}{A_b}\right)^{3/4} = C_s^6 \left(A_a^3 A_b\right)^{-3/4} \tag{3.25}$$

Summarizing, the density in the whole domain can be expressed as:

$$\rho(z) = \begin{cases} \left(C_s^2 A_a^{-1} - \dfrac{gz}{4A_a}\right)^3 & \text{for} \quad z > 0 \\[2ex] \left(C_s^2 \left(A_a^3 A_b\right)^{-1/4} - \dfrac{gz}{4A_b}\right)^3 & \text{for} \quad z < 0 \end{cases} \tag{3.26}$$

3.6.2 Point source gravitational field

The point source gravitational field initial conditions are very similar to the ones in the constant case, except a few minor differences due to the change in the aforementioned field. Again z_0 will be the distance between the centre of the star and the centre of the box. The gravitational potential is of the form

$$\Phi = -\frac{GM_{wd}}{z+z_0} = -\frac{1}{z+z_0}\mathrm{M}_{wd} \tag{3.27}$$

which yields a gravitational field of:

$$\mathbf{g} = -\nabla\Phi = (0, 0, -\frac{1}{(z+z_0)^2}\mathrm{M}_{wd}) \tag{3.28}$$

Using the same geometry and boundary conditions at the interface as we used in the previous case, we impose again hydrostatic equilibrium as in eq. 3.13 with the new gravitational field, to obtain:

$$\frac{d\mathrm{P}}{dz} = -\frac{1}{(z+z_0)^2}\mathrm{M}_{wd}\rho \tag{3.29}$$

Using the expression for the pressure given by eq. 3.12, a constant value for A in each region, deriving and rearranging, this time we obtain:

$$\frac{d\rho}{\rho^{2/3}} = -\frac{3}{4A}\frac{\mathrm{M}_{wd}}{(z+z_0)^2}dz \tag{3.30}$$

Integrating in region b:

$$\int_{\rho_b(z)}^{\rho_b(z_i)} \frac{d\rho}{\rho^{2/3}} = \int_z^{z_i} -\frac{3}{4A_b}\frac{\mathrm{M}_{wd}}{(z+z_0)^2}dz \tag{3.31}$$

Setting again $z_i = 0$, we obtain densities of:

$$\rho_b(z) = \left(\rho_b(0)^{1/3} - \frac{\mathrm{M}_{wd}}{4A_b z_0}\frac{z}{z+z_0}\right)^3 \tag{3.32}$$

And:

$$\rho_a(z) = \left(\rho_a(0)^{1/3} - \frac{\mathrm{M}_{wd}}{4A_a z_0}\frac{z}{z+z_0}\right)^3 \tag{3.33}$$

The calculations of $\rho_a(0)$ and $\rho_b(0)$ do not depend on the form of the gravitational field, so the expressions we calculated before are still valid. Therefore, the final density profile in the whole region for the point source gravity is:

$$\rho(z) = \begin{cases} \left(C_s^2 A_a^{-1} - \dfrac{\mathrm{M}_{wd}}{4A_a z_0} \dfrac{z}{z+z_0} \right)^3 & \text{for} \quad z > 0 \\ \\ \left(C_s^2 \left(A_a^3 A_b\right)^{-1/4} - \dfrac{\mathrm{M}_{wd}}{4A_b z_0} \dfrac{z}{z+z_0} \right)^3 & \text{for} \quad z < 0 \end{cases} \tag{3.34}$$

The expressions that we have found for the density in both the constant gravitational field and the point source one can be plotted and compared. In figure 3.2 we can see how these density profiles look like. We have taken values of A_a and A_b so that the density jump at $z = 0$ is of $\frac{\Delta\rho}{\rho} = 0.1$, that is, a 10% change in density.

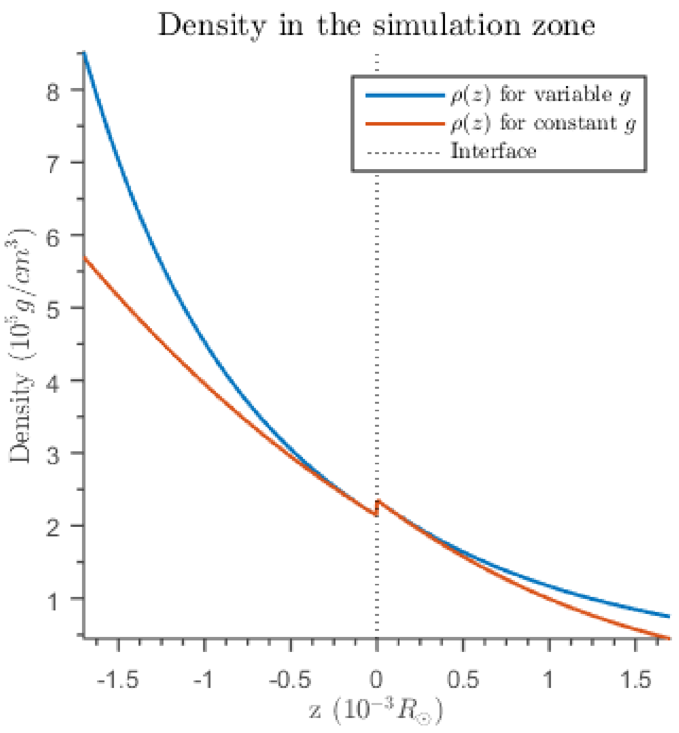

Fig. 3.2: Density profile in the simulation zone as in expressions 3.26 and 3.34 when A_a and A_b are chosen so that $\Delta\rho/\rho = 0.1$ at the interface $z = 0$.

3.6.3 Obtaining the polytropic constants

In the last two sections we have shown that knowing the mass of the white dwarf M_{wd}, the speed of sound C_s, the height at which the jump in density happens z_0 and, the polytropic constants of proportionality A_a and A_b, the initial density profile in the simulation zone is fully determined.

From those constants, the ones that are still lacking are A_a and A_b. In order to obtain them, we use the approximation of the solution for the Lane-Emden equation (Liu 1996) in the case where $n = 3$ and impose a $\mathrm{M}_{wd} = 1\mathrm{M}_\odot$ and a $R_{wd} = 4.7 \cdot 10^8 cm$. Imposing those values, we can calculate the density at $r = 0.75 R_{wd}$ or equivalently $z = 0$. With that density and equation 3.22 we obtain the value for A_a. Finally, dividing the density by $1 + \Delta\rho/\rho$ we get the density at the lower side of the interface. Using equation 3.25 we obtain the value for A_b.

3.6.4 Initial perturbation

In order to break the initial equilibrium, we add a small perturbation in the third component of the momentum. This perturbation is given by:

```
if(x3 < lz/2.0 && x3 > -lz/2.0)
  pGrid->U[k][j][i].M3 =
      amp*0.5*(cos(2.0*PI*x2/ly))*(1.0+cos(2.0*PI*x3/lz))
         *0.5*(1.0+cos(2.0*PI*x1/lx))*(1.0+0.1*(ran2(&iseed)-0.5));
```

Where `ran2(&iseed)` generates a random number between 0 and 1, `amp` is a user-defined constant and `lx`, `ly` and `lz` are lengths that change the shape of the perturbation. The `if` statement ensures that we take only one oscillation of the cosine that depends on the Z axis. Written in a more intelligible way the amplitude of the initial perturbation is given by:

$$p_z(x, y, z) = \text{amp}\frac{1 + cos(2\pi x/l_x)}{2} cos(2\pi y/l_y) \frac{1 + cos(2\pi z/l_z)}{2} K_{rand}(x, y, z) \tag{3.35}$$

where $K_{rand}(x, y, z)$ is a random number between 0.95 and 1.05. This random modulation exists to help break the symmetry of the simulation. A qualitative image of the module of the perturbation can be seen in figure 3.3.

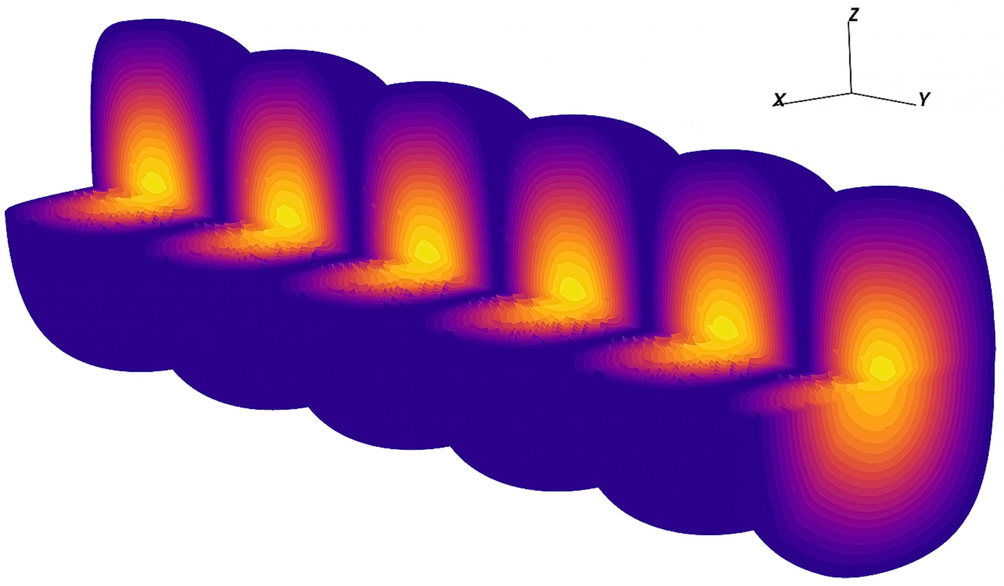

Fig. 3.3: Module of the perturbation on the momentum in the Z component. A legend is not shown since its magnitude is dependant on the variable amp and its shape on l_x, l_y and l_z.

3.7 Boundary conditions

In directions X and Y we choose periodic boundary conditions. In direction Z we impose hydrostatic equilibrium (as in the initial conditions) moduled by an exponential such that its value is 1 at the boundary and 0 at one quarter of the length of the box in the Z direction away from the boundary. We also kill the momentum in the Z direction, with an identical modulation.

3.8 Main files

Here we describe the main files of the Athena code that were modified. The source code of these modifications of the files can be found in the codes annex.

3.8.1 Input File

The input file defines the geometry of the domain: the length and the amount of cells in each spatial component. It defines the outputs (history, VTKs and resets) and the time between each of them.

- VTKs allow us to visualize the simulation with programs such as VisIt or ParaView.

- The history file contains multiple volume averaged magnitudes, such as total energy, kinetc energy, mass, etc. A line with the new values for those magnitudes is appended to the history file when the time specified in the input file passes by. Other values apart from the default ones can be defined by the user.

- The reset files allow us to pick up the simulation at the time the last reset file was generated. Without them, you can not stop the simulation and continue from the last point.

It also sets the maximum amount of time for the simulation and the maximum amount of time steps. Finally it sets some problem specific constants. In our case gamma. The input file is the same for all the simulations. The full file can be found in the annex.

3.8.2 Problem Generator

The problem generator defines the initial conditions, the standard periodic boundary conditions, and the shape and strength of the gravitational field. It is also in the problem generator where we specify extra volume averaged magnitudes to be displayed in the history file. In our particular case we request the code to output the magnitude of the square velocity, averaged over the whole volume, as a measure of the intensity of convection. Since both the shape of gravity and the initial density profiles change if we simulate the constant or the point source gravitational field, we have two problem generators. The relevant parts of both files can be found in the annex.

3.8.3 CTU integrator

We use the Corner Transport Upwind (CTU) integrator. Details on the method can be found at (Gardiner & Stone 2007).

In this part of the code, we implemented our own boundary conditions in order to mimic the crystallization conditions inside the white dwarf. These can too be found in the annex.

Results 4

We have run 8 simulations, with different gravity shapes and intensities as can be seen in table 4.1.

Simulation no.	1	2	3	4	5	6	7	8
$M_{wd}[M_\odot]$	0.8	0.9	1.0	1.1	0.8	0.9	1.0	1.1
Gravity shape	C	C	C	C	V	V	V	V

Tab. 4.1: Setups of the simulations carried out, where C stands for constant gravity and V for (variable) point source gravity.

These different setups account for different density profiles which can be seen in figure 4.1, we have set the profiles in such a way that the density at $z = 0$ is the same for all the simulations.

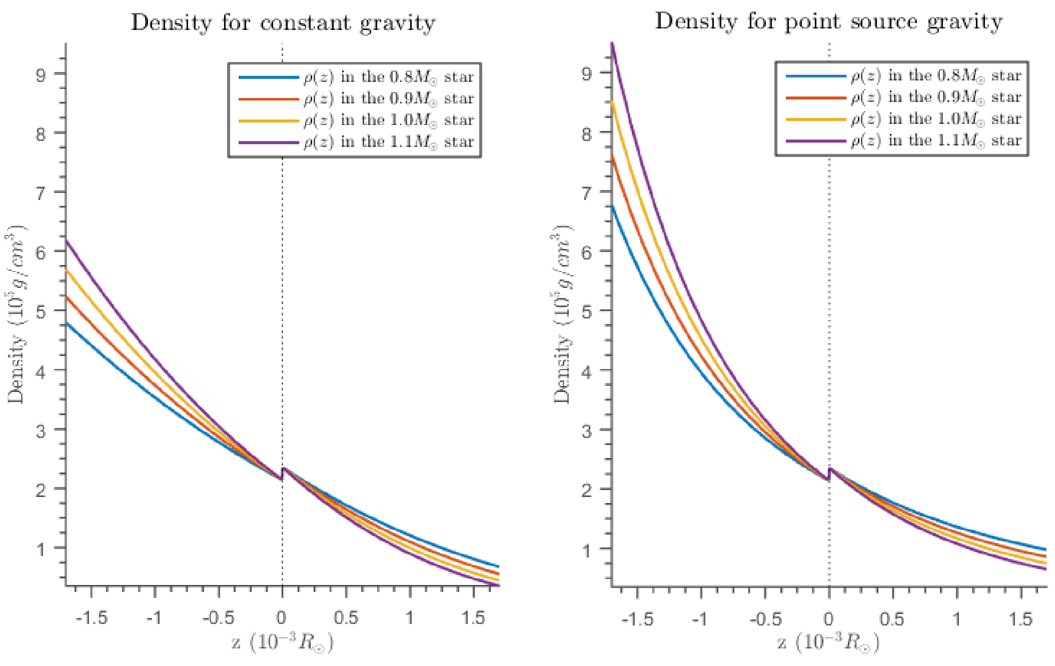

Fig. 4.1: Initial density in the simulation zone for eight simulations depending on their mass and shape of gravity. With a jump in density of $\Delta\rho/\rho = 0.1$.

Since the density is different in each setup, in order to compare the intensity of convection in each simulation we needed a unit that we could measure that did not depend on the density. Therefore we could neither use the kinetic energy nor the momentum. We settled in measuring a mean of the square of the velocities in each cell. Writing a few lines of code in the problem generator file, we get the code to output:

$$\langle \mathbf{v}^2 \rangle = \frac{1}{N} \sum_{cells} \frac{p_x^2 + p_y^2 + p_z^2}{\rho^2} \tag{4.1}$$

Where N is the number of cells. Since $\boldsymbol{p}$ is defined as $\boldsymbol{p} = \rho \mathbf{v}$ in the code (Stone et al. 2008), equation 4.1 is equivalent to

$$\langle \mathbf{v}^2 \rangle = \frac{1}{N} \sum_{cells} \left(\mathrm{v}_x^2 + \mathrm{v}_y^2 + \mathrm{v}_z^2 \right) = \frac{1}{N} \sum_{cells} |\mathbf{v}|^2 \tag{4.2}$$

Qualitative comparisons can be done with this quantity. We plotted its value for the different simulations in the following figures.

As could be expected such a quantity doesn't attain a fixed value, and rather oscillates as time goes by. In order to be able to compare the simulations, we computed an accumulated average. We waited an arbitrary time to start to calculate this average, since the simulation starts at a unrealistic static equilibrium. We ran the simulations for 15 computational seconds, which is equivalent to 12 minutes and 30 seconds of real simulated time. Each one of the eight nodes of the Atria computer cluster ran a simulation for ten days long in order to obtain these results.

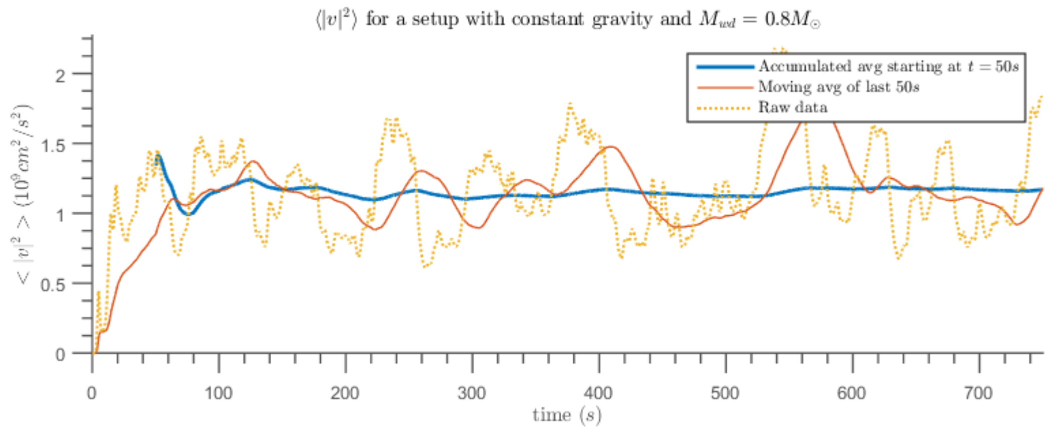

Fig. 4.2: $\langle \mathbf{v}^2 \rangle$ versus time in a white dwarf with constant gravity and a mass of $0.8 M_\odot$.

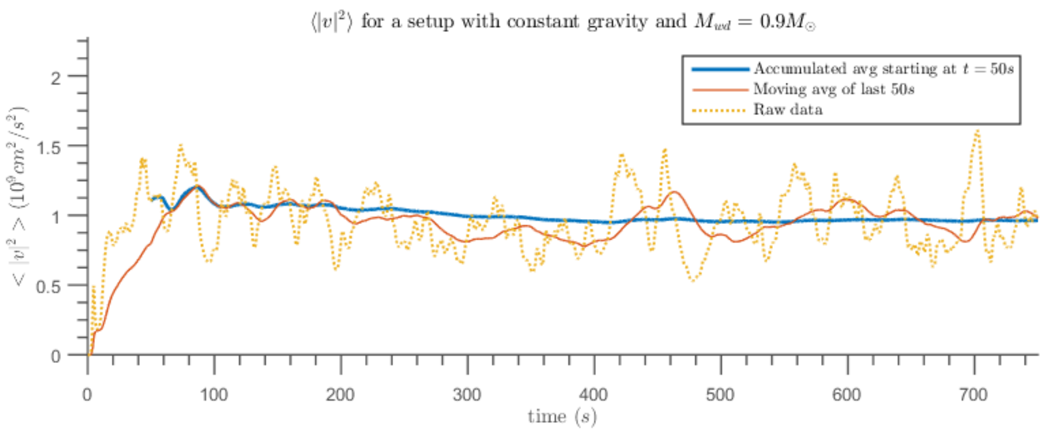

Fig. 4.3: $\langle \mathbf{v}^2 \rangle$ versus time in a white dwarf with constant gravity and a mass of $0.9M_\odot$.

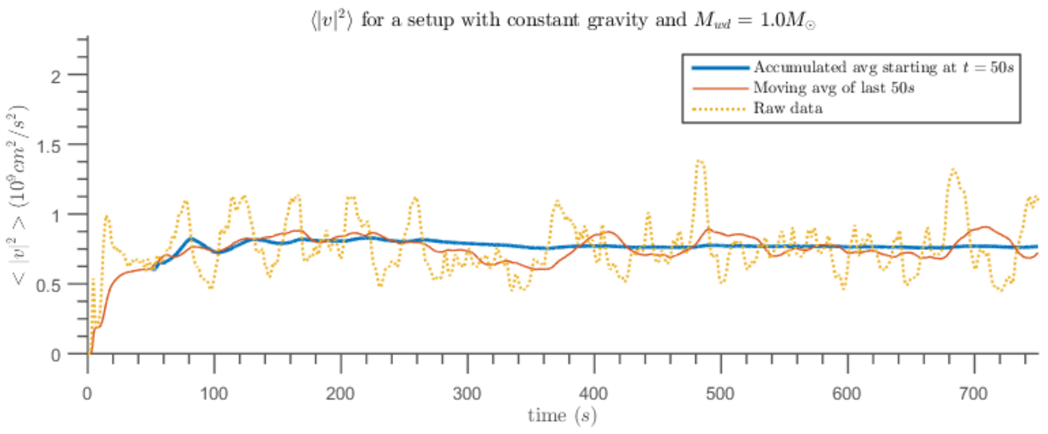

Fig. 4.4: $\langle \mathbf{v}^2 \rangle$ versus time in a white dwarf with constant gravity and a mass of $1.0M_\odot$.

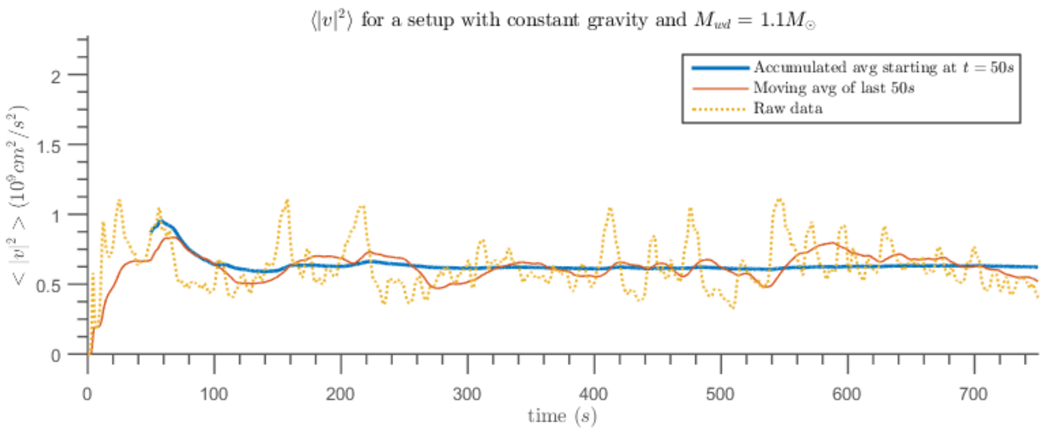

Fig. 4.5: $\langle \mathbf{v}^2 \rangle$ versus time in a white dwarf with constant gravity and a mass of $1.1M_\odot$.

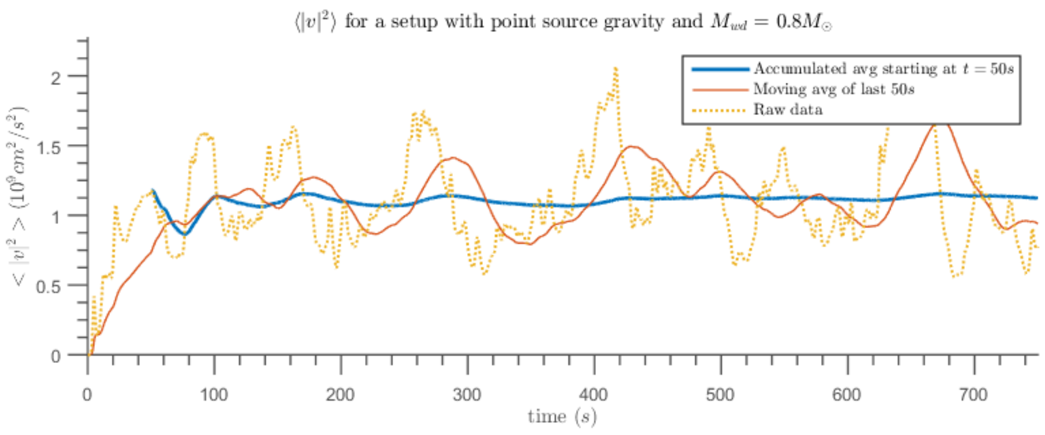

Fig. 4.6: $\langle \mathbf{v}^2 \rangle$ versus time in a white dwarf with point source gravity and a mass of $0.8M_\odot$.

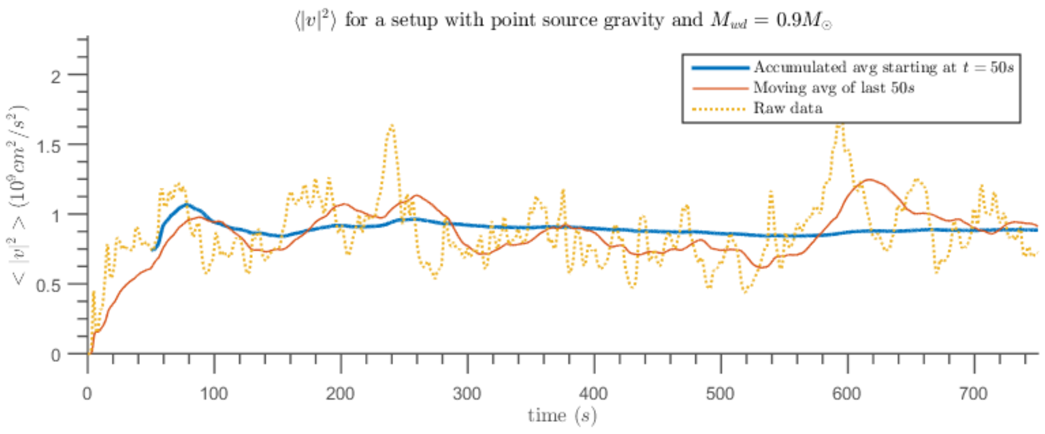

Fig. 4.7: $\langle \mathbf{v}^2 \rangle$ versus time in a white dwarf with point source gravity and a mass of $0.9M_\odot$.

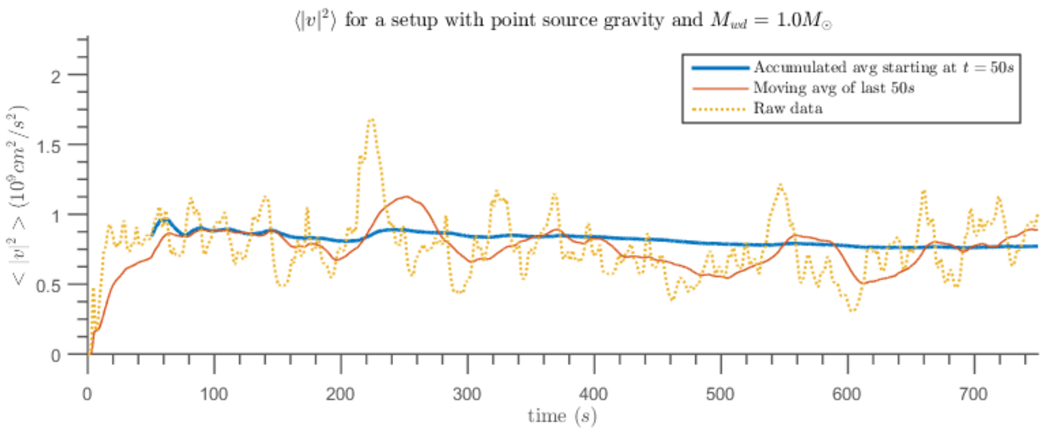

Fig. 4.8: $\langle \mathbf{v}^2 \rangle$ versus time in a white dwarf with point source gravity and a mass of $1.0M_\odot$.

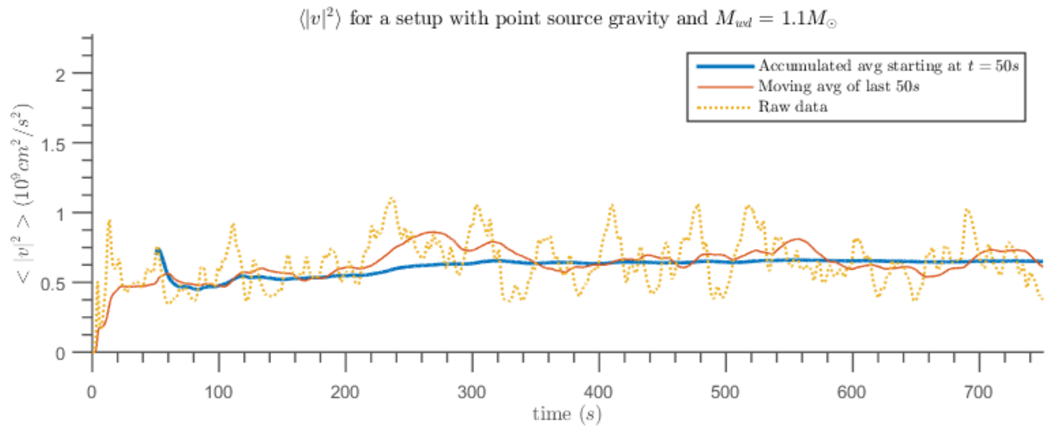

Fig. 4.9: $\langle \mathbf{v}^2 \rangle$ versus time in a white dwarf with point source gravity and a mass of $1.1M_\odot$.

All the simulations show a similar behaviour. At first the mean in velocity is zero since they all start at a static equilibrium. Then there is an initial spike as the first bubbles of heavier fluid reach the bottom and the less dense ones reach the top (it is barely visible due to the long duration of the simulations). From then on, all of them follow an oscillating behaviour with an amplitude and a mean that depend on the white dwarfs mass.

We can compare the simulations with each other with a single metric: the final value of the accumulated average of $\langle \mathbf{v}^2 \rangle$. These values can be seen in table 4.2.

Simulation no.	1	2	3	4	5	6	7	8
$M_{wd}[M_\odot]$	0.8	0.9	1.0	1.1	0.8	0.9	1.0	1.1
Gravity shape	C	C	C	C	V	V	V	V
$\langle \mathbf{v}^2 \rangle (10^9 cm^2/s^2)$	1.170	0.963	0.767	0.623	1.124	0.885	0.772	0.649

Tab. 4.2: Results for the different setups, where C stands for constant gravity and V for (variable) point source gravity.

A plot of the time average of the volume average of $\mathbf{v}^2$ against the mass in each simulation can be seen in figure 4.10. While the difference in the intensity of convection is evident for a change in mass, the differences between the simulations with constant gravity compared with the ones with point source gravity are not as significant. This similarity in convection intensity for both gravitational field shapes shows that assuming a constant gravitational field throughout the convection zone is an acceptable approximation for back of the envelope calculations.

The change in the measured parameter $\langle \mathbf{v}^2 \rangle$ induces a change in the overall kinetic energy (as can be seen in table 4.11) and thus, through equation 2.2 a change in the total magnetic field. With no other factors taken into account, this would indicate that as the mass of the white dwarf increases the magnetic field decreases. This behavior is opposite to the one predicted by Isern et al. 2017, which was that as the mass of the white dwarf increases, so would its magnetic field.

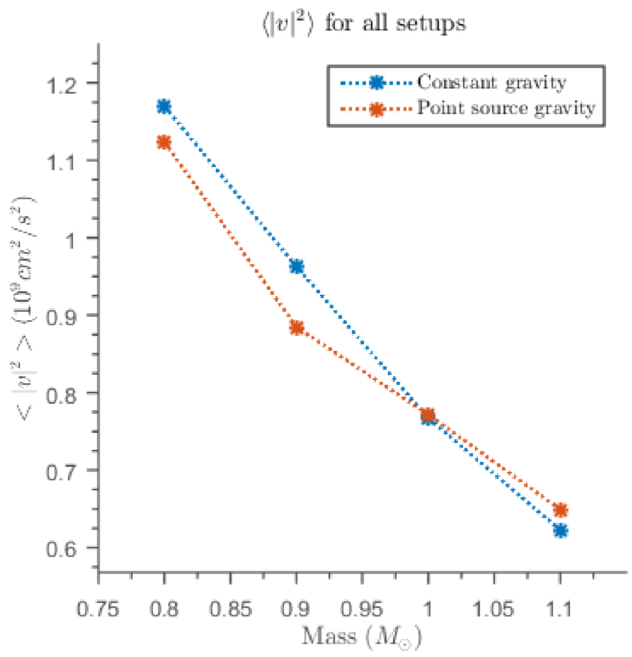

Fig. 4.10: Value obtained in the simulations of $\langle \mathbf{v}^2 \rangle$ as a function of mass for all setups.

It should be noted that the values for the overall kinetic energy that can be seen in figure 4.11 are quite high. In Isern et al. 2017 the predicted values for $\log E/V$ when expressed in ergs per cubic centimetre is between 5 and 6, while the ones we obtain are between 19 and 20 (when expressed in the same units). This may be caused by different factors. The most probable one is the fact that we used a very big density jump ($\Delta\rho/\rho = 0.1$). Other factors could be the absence of Coriolis forces which limit convection (Sakurai 1972) and, the absence of magnetic fields which limit convection too (Valyavin et al. 2014).

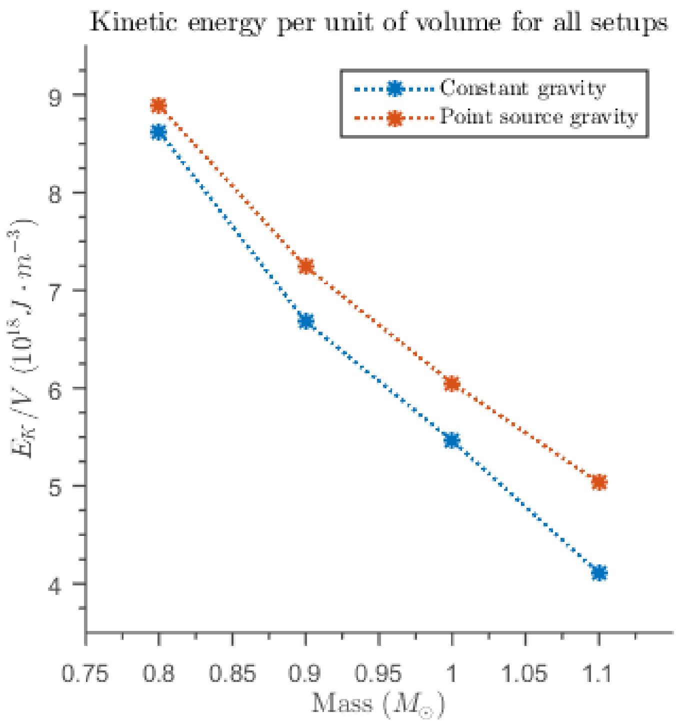

Fig. 4.11: Value obtained in the simulations of the kinetic energy as a function of mass for all setups.

While all those plots and numeric results are fine and indeed necessary in order to make comparisons and draw conclusions. There is also a certain beauty in watching different magnitudes evolve as time goes by. In figure 4.12 we can see snapshots of various magnitudes in different moments in time. The leftmost image shows the density throughout the simulation zone, we use a color scheme that allows us to see small variations in the density even though there is a big gradient from bottom to top. The image in the centre shows the magnitude of the momentum, light color indicating a bigger momentum. The third and rightmost image shows the derivative of the density, with blue indicating a decrease in density and red indicating an increase. Lighter colors show a higher absolute value of the derivative (light blue and yellow respectively).

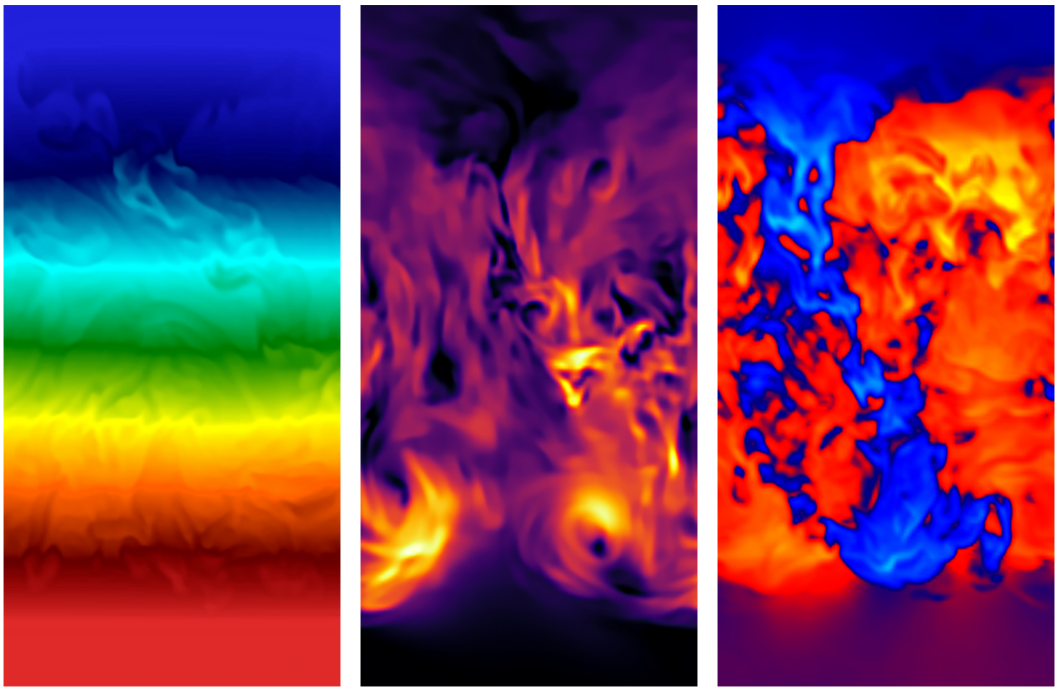
Fig. 4.12: Snapshots of different magnitudes. Density (left), momentum magnitude (centre) and, density derivative (right).

5 Conclusion

5.1 Summary and conclusion

We have developed a setup using the Athena Code that simulates the convective region of a white dwarf. We have run multiple simulations modifying the shape and intensity of the gravitational field (through a change in the total mass of the star). Although improvable, these simulations stand as a proof of concept on how to create a setup for white dwarfs capable of simulating Rayleigh-Taylor induced convection, and maintaining this convection stable. These simulations yield two conclusions:

The **first one** is that, as the mass of the star increases, the intensity of convection decreases; there is a strong relationship between convection and gravitational field intensity at the interface.

The **second one** is that whilst the relation between convection and gravitational field intensity is solid, the same can not be said about its relation with the shape of the gravitational field.

These conclusions can be related with previous work: the first conclusion seems to contradict the prediction in Isern et al. 2017. This could be due to a variety of factors like the absence of rotation in our setup, the way in which we model the change in density due to crystallization or to the simplified evolution of chemical composition. Or it even could be due to a mistake in the aforementioned paper. This calls for both, a revision of the assumptions made in that paper and, the development of a more rigorous computational model that takes into account a broader set of variables. In any case, this work serves as a foundation on which further improvements can be made, some are mentioned in the next section.

The second conclusion seems to prove true that a constant field approximation is a reasonable one.

5.2 Future work

The results of this work should be taken with a grain of salt. They will be useful indeed as an initial approach to the study of convection due to phase separation through crystallization in white dwarfs, but a lot of work has to be carried out still in order to asseverate with higher confidence the results of this document. Some of the future work that has to be done is:

- Migrate the project to a spherical geometry and add the rotation one would expect in a white dwarf. This rotation would then introduce Coriolis forces, which have been shown to limit convection (Sakurai 1972). Rotation will besides, depending on the changes in density, induce Kelvin-Helmholtz instabilities. Simulating in spherical geometry was actually the initial intention in this project, but due to the limited time available and the inherent complexity due to the factors stated above, we ended up using a Cartesian geometry without rotation. The introduction of the more complicated setup was left as future work to be carried out once a stable setup in Cartesian coordinates was attained.

- Use a more realistic gravity and add self gravity.

- Add the magnetic field to the simulation (as it is known that in the case of Earth, the magnetic field limits itself (Buffett 2000) and it does so too for white dwarfs (Valyavin et al. 2014)), and directly extract the magnetic field, instead of using formula 2.2 to approximate it.

- Create a more realistic setup for the crystallization process.

- Add an equation of state that evolves the chemistry of the mixture and then calculate the pressure as given by equation 15 in Mochkovitch 1983:

$$P = P_e - 0.3\Gamma\rho RT\frac{\overline{Z^{5/3}}}{\mu} \tag{5.1}$$

- Running the simulation with higher resolution and during longer times is always a good idea.

Bibliography

6

[1] Abrikosov, A. A. 1960, Sov. Phys. JETP, **12**, 1254

[2] Althaus, L. G., Córsico, A. H., Isern, J., & García-Berro, E. 2010, A&A Rev., **18**, 471

[3] Barrat, J. L., Hansen, J. P., & Mochkovitch, R. 1988, A&A, **199**, L15

[4] Bildsten, L. & Hall, D. M. 2001, ApJ, **549**, L219

[5] Bravo, E., Isern, J., Canal, R., & Labay, J. 1992, A&A, **257**, 534

[6] Buffett, B. A. 2000, Science, **288**, 2007

[7] Camisassa, M. E., Althaus, L. G., Córsico, A. H., Vinyoles, N., Serenelli, A. M., Isern, J., Miller-Bertolami, M. M., & García-Berro, E. 2016, ApJ, **823**, 158

[8] Christensen, U. R. 2010, Space Sci. Rev., **152**, 565

[9] Ferrario, L., de Martino, D., & Gänsicke, B. T. 2015, Space Sci. Rev., **191**, 111

[10] Fowler, R. H. 1926, MNRAS, **87**, 114

[11] García-Berro, E., Torres, S., Althaus, L. G., Renedo, I., Lorén-Aguilar, P., Córsico, A. H., Rohrmann, R. D., Salaris, M., & Isern, J. 2010, Nature, **465**, 194

[12] García-Berro, Lorén-Aguilar., L., Aznar-Siguán, G., Torres, S., Camacho, J., Althaus, L. G., Córsico, A. H., Külebi, B., & Isern, J. 2012, ApJ, **749**, 25

[13] Gardiner, T. A. & Stone, J. M. 2008, J. Comput. Phys., **227**, 4123

[14] Horowitz, C. J., Schneider, A.S., & Berry, D. K. 2010, Phys. Rev. Lett., **104**, 231101

[15] Isern, J., Mochkovitch, R., García-Berro, E., & Hernanz, M. 1997, ApJ, **485**, 308

[16] Isern, J., García-Berro, E., Hernanz, M., & Chabrier, G. 2000, ApJ, **528**, 397

[17] Isern, J., García-Berro, E., Külebi, B., & Lorén-Aguilar, P. 2017, ApJ, **836**, L28

[18] Kepler, S. O., Kleinman, S. J., Nitta, A., Koester, D., Castanheira, B. G., Giovannini, O., Costa, A. F. M., & Althaus, L. 2007, MNRAS, **375**, 1315

[19] Kilic, M., Allende Prieto, C., Brown, W. R., & Koester, D. 2007, ApJ, **660**, 1451

[20] Kirzhnits, D. A. 1960, Sov. Phys. JETP, **11**, 365

[21] Lamb, D. Q. & Van Horn, H. M. 1975, ApJ, **200**, 306

[22] Levy, E.H. & Rose, W.K. 1974, ApJ, **193**, 419

[23] Lister, J. R. & Buffet, B. A. 1995, PEPI, **91**, 17

[24] Liu, F. K. 1996, MNRAS, **281**, 1197

[25] Mochkovitch, R. 1983, A&A, **122**, 212

[26] Nordhus, J., Wellons, S., Spiegel, D. S., Metzger, B. D., & Blackman, E. G. 2011, PNAS, **108**, 3135

[27] Ritossa C., García-Berro E., & Iben I. 1996, ApJ, **469**, 489

[28] Sakurai, K. 1972, ApJ, **177**, 423

[29] Salpeter, E. E. 1961, ApJ, **134**, 669

[30] Stone, J. M., Gardiner, T. A., Teuben, P., Hawley, J. F., & Simon, J. B. 2008, ApJS, **178**, 137

[31] Tout, C. A., Wickramasinghe, D. T., Liebert, J., Ferrario, L., & Pringle, J.E. 2008, MNRAS, **387**, 897

[32] Valyavin, G., Shulyak, D., Wade, G. A., Antonyuk, K., Zharikov, S. V., Galazutdinov, G. A., Plachinda, S., Bagnulo, S., Machado, L. F., Alvarez, M., Clark, D. M., Lopez, J. M., Hiriart, D., Han, I., Jeon, Y. B., Zurita, C., Mujica, R., Burlakova, T., Szeifert, T., & Burenkov, A. 2014, Nature, **515**, 88

[33] Van Horn, H. M. 1968, ApJ, **151**, 227

[34] Wickramasinghe, D. T. & Ferrario, L. 2005, MNRAS, **356**, 1576

List of Figures

ANNEX: Main code files

7

7.1 Problem generation

7.1.1 Constant gravity

src/prob/wd_Gconst.c

Lines with [...] denote unimportant code that is not specific for our problem and therefore it is not shown in this document.

```
[...]
#define z0 0.05              // Variable definitions
#define M_WD 1.0
#define Aa 1.1717
#define Ab 1.3303
#define C_sonido 2.0
[...]
void problem(DomainS *pDomain){
  [...]
 /* 3D PROBLEM -----------------------------------------------------------*/
 /* Initialize problem with interface at z=0.0 */
 double g_const = M_WD/SQR(z0);
 double rho_a = pow(C_sonido, 6.0)*pow(Aa, -3.0);
 double rho_b = pow(C_sonido, 6.0)*pow((pow(Aa,3.0)*Ab), -3.0/4.0);

 for (k=ks; k<=ke; k++)
   for (j=js; j<=je; j++)
     for (i=is; i<=ie; i++) {
       cc_pos(pGrid,i,j,k,&x1,&x2,&x3);
       pGrid->U[k][j][i].d =
         pow(( pow(rho_b,1.0/3.0) - g_const/(4.0*Ab)*x3 ), 3.0);
       pGrid->U[k][j][i].E = ( Ab * pow(pGrid->U[k][j][i].d, 4.0/3.0) )/Gamma_1;
       pGrid->U[k][j][i].M1 = 0.0;
       pGrid->U[k][j][i].M2 = 0.0;
       pGrid->U[k][j][i].M3 = 0.0;
       if(x3 < lz/2.0 && x3 > -lz/2.0)
         pGrid->U[k][j][i].M3 =
           amp*0.5*(cos(2.0*PI*x2/ly))*(1.0+cos(2.0*PI*x3/lz))
               *0.5*(1.0+cos(2.0*PI*x1/lx))*(1.0+0.1*(ran2(&iseed)-0.5));
       if (x3 > 0.0) {
         pGrid->U[k][j][i].d =
           pow(( pow(rho_a,1.0/3.0) - g_const/(4.0*Aa)*x3 ), 3.0);
         pGrid->U[k][j][i].M3 *=
           pGrid->U[k][j][i].d/pow((pow(rho_b,1.0/3.0)-g_const/(4.0*Ab)*x3),3.0);
         pGrid->U[k][j][i].E = (Aa * pow(pGrid->U[k][j][i].d, 4.0/3.0))/Gamma_1;
       }
       pGrid->U[k][j][i].E+=0.5*SQR(pGrid->U[k][j][i].M3)/pGrid->U[k][j][i].d;
     }

```

```c

/* Enroll gravitational potential.
 * Use special boundary condition routines. */
  StaticGravPot = grav_pot3;

  if (pDomain->Disp[2] == 0) bvals_mhd_fun(pDomain, left_x3,  reflect_ix3);
  if (pDomain->MaxX[2] == pDomain->RootMaxX[2])
    bvals_mhd_fun(pDomain, right_x3, reflect_ox3);
  dump_history_enroll(modulo_v2, "<v^2>");
  return;
} /* end of 3D initialization */

[...]

/*----------------------------------------------------------------------------*/
/*! \fn static void reflect_ix3(GridS *pGrid)
 *  \brief Special reflecting boundary functions in x3 for 3D sims
 */

static void reflect_ix3(GridS *pGrid){
  Real x1, x2, x3;
  int ks = pGrid->ks;
  int i,j,k,il,iu,jl,ju; /* i-lower/upper;  j-lower/upper */
  iu = pGrid->ie + nghost;
  il = pGrid->is - nghost;
  ju = pGrid->je + nghost;
  jl = pGrid->js - nghost;
  double g_const = M_WD/SQR(z0);
  double rho_b = pow(C_sonido, 6.0)*pow((pow(Aa,3.0)*Ab), -3.0/4.0);
  for (k=1; k<=nghost; k++)
    for (j=jl; j<=ju; j++)
      for (i=il; i<=iu; i++) {
        cc_pos(pGrid,i,j,ks-k,&x1,&x2,&x3);
        pGrid->U[ks-k][j][i].d =
          pow(( pow(rho_b,1.0/3.0) - g_const/(4.0*Ab)*x3 ), 3.0);
        pGrid->U[ks-k][j][i].E =
          ( Ab * pow(pGrid->U[ks-k][j][i].d, 4.0/3.0) )/Gamma_1;
        pGrid->U[ks-k][j][i].M1 = 0.0;
        pGrid->U[ks-k][j][i].M2 = 0.0;
        pGrid->U[ks-k][j][i].M3 = 0.0;
      }
  return;
}

/*----------------------------------------------------------------------------*/
/*! \fn static void reflect_ox3(GridS *pGrid)
 *  \brief Special reflecting boundary functions in x3 for 3D sims
 */

static void reflect_ox3(GridS *pGrid){
  Real x1, x2, x3;
  int ke = pGrid->ke;
  int i,j,k ,il,iu,jl,ju; /* i-lower/upper;  j-lower/upper */
  iu = pGrid->ie + nghost;
  il = pGrid->is - nghost;
  ju = pGrid->je + nghost;
  jl = pGrid->js - nghost;
  double g_const = M_WD/SQR(z0);
  double rho_a = pow(C_sonido, 6.0)*pow(Aa, -3.0);
  for (k=1; k<=nghost; k++)
    for (j=jl; j<=ju; j++)
      for (i=il; i<=iu; i++) {
        cc_pos(pGrid,i,j,ke+k,&x1,&x2,&x3);
        pGrid->U[ke+k][j][i].d =
          pow(( pow(rho_a,1.0/3.0) - g_const/(4.0*Aa)*x3 ), 3.0);
        pGrid->U[ke+k][j][i].E =
          ( Aa * pow(pGrid->U[ke+k][j][i].d, 4.0/3.0) )/Gamma_1;
        pGrid->U[ke+k][j][i].M1 = 0.0;
        pGrid->U[ke+k][j][i].M2 = 0.0;
        pGrid->U[ke+k][j][i].M3 = 0.0;
      }
  return;
}

/*! \fn static Real grav_pot3(const Real x1, const Real x2, const Real x3)
 *  \brief Gravitational potential; g = 0.1
 */
static Real grav_pot3(const Real x1, const Real x2, const Real x3){
  double g_const = M_WD/SQR(z0);
  return g_const*x3;
}

static Real modulo_v2(const GridS *pG, const int i, const int j, const int k) {
     return (SQR(pG->U[k][j][i].M1) +
             SQR(pG->U[k][j][i].M2) +
             SQR(pG->U[k][j][i].M3))/SQR(pG->U[k][j][i].d);
}
```

7.1.2 Point source gravity

src/prob/wd_Gvar.c

Lines with `[...]` denote unimportant code that is not specific for our problem and therefore it is not shown in this document.

```
[...]
#define z0 0.05              // Variable definitions
#define M_WD 1.0
#define Aa 1.1717
#define Ab 1.3303
#define C_sonido 2.0
[...]
void problem(DomainS *pDomain){
  [...]
  /* 3D PROBLEM ---------------------------------------------------------------*/
  /* Initialize problem with interface at z=0.0 */

  double rho_a = pow(C_sonido, 6.0)*pow(Aa, -3.0);
  double rho_b = pow(C_sonido, 6.0)*pow((pow(Aa,3.0)*Ab), -3.0/4.0);

  for (k=ks; k<=ke; k++)
    for (j=js; j<=je; j++)
      for (i=is; i<=ie; i++) {
        cc_pos(pGrid,i,j,k,&x1,&x2,&x3);
        pGrid->U[k][j][i].d =
          pow(( pow(rho_b,1.0/3.0) - M_WD/(4.0*Ab*z0)*x3/(x3+z0) ), 3.0);
        pGrid->U[k][j][i].E = ( Ab * pow(pGrid->U[k][j][i].d, 4.0/3.0) )/Gamma_1;
        pGrid->U[k][j][i].M1 = 0.0;
        pGrid->U[k][j][i].M2 = 0.0;
        pGrid->U[k][j][i].M3 = 0.0;
        if(x3 < lz/2.0 && x3 > -lz/2.0)
          pGrid->U[k][j][i].M3 =
            amp*0.5*(cos(2.0*PI*x2/ly))*(1.0+cos(2.0*PI*x3/lz))
               *0.5*(1.0+cos(2.0*PI*x1/lx))*(1.0+0.1*(ran2(&iseed)-0.5));
        if (x3 > 0.0) {
          pGrid->U[k][j][i].d =
            pow(( pow(rho_a,1.0/3.0) - M_WD/(4.0*Aa*z0)*x3/(x3+z0) ), 3.0);
          pGrid->U[k][j][i].M3 *=
            pGrid->U[k][j][i].d /
              pow(( pow(rho_b,1.0/3.0) - M_WD/(4.0*Ab*z0)*x3/(x3+z0) ), 3.0);
          pGrid->U[k][j][i].E = (Aa * pow(pGrid->U[k][j][i].d, 4.0/3.0))/Gamma_1;
        }
        pGrid->U[k][j][i].E+=0.5*SQR(pGrid->U[k][j][i].M3)/pGrid->U[k][j][i].d;
      }

/* Enroll gravitational potential.
 * Use special boundary condition routines. */
  StaticGravPot = grav_pot3;

  if (pDomain->Disp[2] == 0) bvals_mhd_fun(pDomain, left_x3,  reflect_ix3);
  if (pDomain->MaxX[2] == pDomain->RootMaxX[2])
    bvals_mhd_fun(pDomain, right_x3, reflect_ox3);
  dump_history_enroll(modulo_v2, "<v^2>");
  return;
} /* end of 3D initialization */

[...]

/*----------------------------------------------------------------------------*/
/*! \fn static void reflect_ix3(GridS *pGrid)
 *  \brief Special reflecting boundary functions in x3 for 3D sims
 */

static void reflect_ix3(GridS *pGrid){
  Real x1, x2, x3;
  int ks = pGrid->ks;
  int i,j,k,il,iu,jl,ju; /* i-lower/upper;  j-lower/upper */
  iu = pGrid->ie + nghost;
  il = pGrid->is - nghost;
  ju = pGrid->je + nghost;
  jl = pGrid->js - nghost;
  double rho_b = pow(C_sonido, 6.0)*pow((pow(Aa,3.0)*Ab), -3.0/4.0);
  for (k=1; k<=nghost; k++)
    for (j=jl; j<=ju; j++)
      for (i=il; i<=iu; i++) {
        cc_pos(pGrid,i,j,ks-k,&x1,&x2,&x3);
        pGrid->U[ks-k][j][i].d =
          pow(( pow(rho_b,1.0/3.0) - M_WD/(4.0*Ab*z0)*x3/(x3+z0) ), 3.0);
        pGrid->U[ks-k][j][i].E =
          Ab * pow(pGrid->U[ks-k][j][i].d, 4.0/3.0) )/Gamma_1;
        pGrid->U[ks-k][j][i].M1 = 0.0;
        pGrid->U[ks-k][j][i].M2 = 0.0;
        pGrid->U[ks-k][j][i].M3 = 0.0;
      }
  return;
}

/*----------------------------------------------------------------------------*/
/*! \fn static void reflect_ox3(GridS *pGrid)
 *  \brief Special reflecting boundary functions in x3 for 3D sims
 */

static void reflect_ox3(GridS *pGrid){
  Real x1, x2, x3;
  int ke = pGrid->ke;
  int i,j,k ,il,iu,jl,ju; /* i-lower/upper;  j-lower/upper */
  iu = pGrid->ie + nghost;
  il = pGrid->is - nghost;
  ju = pGrid->je + nghost;
```

```c
  jl = pGrid->js - nghost;
  double rho_a = pow(C_sonido, 6.0)*pow(Aa, -3.0);
  for (k=1; k<=nghost; k++)
    for (j=jl; j<=ju; j++)
      for (i=il; i<=iu; i++) {
        cc_pos(pGrid,i,j,ke+k,&x1,&x2,&x3);
        pGrid->U[ke+k][j][i].d =
          pow(( pow(rho_a,1.0/3.0) - M_WD/(4.0*Aa*z0)*x3/(x3+z0) ), 3.0);
        pGrid->U[ke+k][j][i].E =
          ( Aa * pow(pGrid->U[ke+k][j][i].d, 4.0/3.0) )/Gamma_1;
        pGrid->U[ke+k][j][i].M1 = 0.0;
        pGrid->U[ke+k][j][i].M2 = 0.0;
        pGrid->U[ke+k][j][i].M3 = 0.0;
      }
  return;
}

/*! \fn static Real grav_pot3(const Real x1, const Real x2, const Real x3)
 *  \brief Gravitational potential; g = 0.1
 */
static Real grav_pot3(const Real x1, const Real x2, const Real x3) {
  return -1/(x3+z0)*M_WD;
}

static Real modulo_v2(const GridS *pG, const int i, const int j, const int k) {
  return (SQR(pG->U[k][j][i].M1) +
          SQR(pG->U[k][j][i].M2) +
          SQR(pG->U[k][j][i].M3))/SQR(pG->U[k][j][i].d);
}
```

7.2 Input file

runs/wd_Gvar/athinput.wd_Gvar

```
<comment>
problem = White Dwarf with point source gravity
author  = Joan Marco Rimmek
config  = --with-problem=wd_gvar --with-order=3 --with-flux=hllc --with-eos=
    adiabatic --with-gas=hydro

<job>
problem_id   = wd_Gvar      # problem ID: basename of output filenames
maxout       = 3            # Output blocks number from 1 -> maxout
num_domains  = 1            # number of Domains in Mesh

<output1>
out_fmt = hst               # History data dump
dt      = 0.001             # time increment between outputs

<output2>
out_fmt = vtk               # Binary data dump
dt      = 0.001             # time increment between outputs

<output3>
out_fmt = rst
dt      = 0.1

<time>
cour_no         = 0.4       # The Courant, Friedrichs, & Lewy (CFL) Number
```

```
nlim             = 1000000  # cycle limit
tlim             = 2.0      # time limit

<domain1>
level            = 0        # refinement level this Domain (root=0)
Nx1              = 13       # Number of zones in X1-direction
x1min            = -0.0013  # minimum value of X1
x1max            = 0.0013   # maximum value of X1
bc_ix1           = 4        # boundary condition flag for inner-I (X1)
bc_ox1           = 4        # boundary condition flag for outer-I (X1)

Nx2              = 128      # Number of zones in X2-direction
x2min            = -0.0128  # minimum value of X2
x2max            = 0.0128   # maximum value of X2
bc_ix2           = 4        # boundary condition flag for inner-J (X2)
bc_ox2           = 4        # boundary condition flag for outer-J (X2)

Nx3              = 170      # Number of zones in X3-direction
x3min            = -0.017   # minimum value of X3
x3max            = 0.017    # maximum value of X3
bc_ix3           = 1        # boundary condition flag for inner-K (X3)
bc_ox3           = 1        # boundary condition flag for outer-K (X3)

<problem>
gamma = 1.333333333         # gamma = C_p/C_v
amp   = 0.01                # Amplitude of the perturbation
```

7.3 Modification of the integrator

7.3.1 Constant gravity

src/integrators/integrate_3d_ctu_Gconst.c

```
  float lz      = pD->RootMaxX[2] - pD->RootMinX[2];
  float fzone_lo = pD->RootMinX[2];
  float fzone_hi = pD->RootMinX[2] + lz/4.;
  float rzone_hi = pD->RootMaxX[2];
  float rzone_lo = pD->RootMaxX[2] - lz/4.;

  double g_const = M_WD/SQR(z0);
  double rho_a = pow(C_sonido, 6.0)*pow(Aa, -3.0);
  double rho_b = pow(C_sonido, 6.0)*pow((pow(Aa,3.0)*Ab), -3.0/4.0);

  for (k=ks; k<=ke; k++) {
    i = is;
    j = js;
    cc_pos(pG,i,j,k,&x1,&x2,&x3);
    // LOWER REGION (b)
    if (x3 > fzone_lo && x3 <= fzone_hi) {
      float factord = exp(-0.001*(fabs(fzone_hi-fzone_lo)+0.000001)
                              /(fabs(      x3-fzone_lo)+0.000001));
      float factorM = factord;
      float dwanted = pow(( pow(rho_b,1.0/3.0) - g_const/(4.0*Ab)*x3 ), 3.0);
      for (j=js; j<=je; j++) {
        for (i=is; i<=ie; i++) {
          cc_pos(pG,i,j,k,&x1,&x2,&x3);
          pG->U[k][j][i].M3 = factorM*pG->U[k][j][i].M3;
          pG->U[k][j][i].d  = dwanted + ( pG->U[k][j][i].d-dwanted )*factord;
```

```c
        float ewanted = ( Ab * pow(pG->U[k][j][i].d, 4.0/3.0) )/Gamma_1;
        ewanted +=
          0.5*(SQR(pG->U[k][j][i].M1)  + SQR(pG->U[k][j][i].M2))/dwanted;
        pG->U[k][j][i].E  = ewanted + ( pG->U[k][j][i].E-ewanted )*factord;
      }
    }
  }
  // UPPER REGION (a)
  if (x3 >= rzone_lo && x3 < rzone_hi) {
    float factord = exp(-0.001*(fabs(rzone_hi-rzone_lo)+0.000001)
                              /(fabs(      x3-rzone_hi)+0.000001));
    float factorM = factord;
    float dwanted = pow(( pow(rho_a,1.0/3.0) - g_const/(4.0*Aa)*x3 ), 3.0);
    for (j=js; j<=je; j++) {
      for (i=is; i<=ie; i++) {
        cc_pos(pG,i,j,k,&x1,&x2,&x3);
        pG->U[k][j][i].M3 = factorM*pG->U[k][j][i].M3;
        pG->U[k][j][i].d  = dwanted + ( pG->U[k][j][i].d-dwanted )*factord;
        float ewanted = ( Aa * pow(pG->U[k][j][i].d, 4.0/3.0) )/Gamma_1;
        ewanted +=
          0.5*(SQR(pG->U[k][j][i].M1)  + SQR(pG->U[k][j][i].M2))/dwanted;
        pG->U[k][j][i].E  = ewanted + ( pG->U[k][j][i].E-ewanted )*factord;
      }
    }
  }
}
```

7.3.2 Point source gravity

src/integrators/integrate_3d_ctu_Gvar.c

```c
float lz       = pD->RootMaxX[2] - pD->RootMinX[2];
float fzone_lo = pD->RootMinX[2];
float fzone_hi = pD->RootMinX[2] + lz/4.;
float rzone_hi = pD->RootMaxX[2];
float rzone_lo = pD->RootMaxX[2] - lz/4.;

double rho_a = pow(C_sonido, 6.0)*pow(Aa, -3.0);
double rho_b = pow(C_sonido, 6.0)*pow((pow(Aa,3.0)*Ab), -3.0/4.0);

for (k=ks; k<=ke; k++) {
  i = is;
  j = js;
  cc_pos(pG,i,j,k,&x1,&x2,&x3);
  // LOWER REGION (b)
  if (x3 > fzone_lo && x3 <= fzone_hi) {
    float factord = exp(-0.001*(fabs(fzone_hi-fzone_lo)+0.000001)
                              /(fabs(      x3-fzone_lo)+0.000001));
    float factorM = factord;
    float dwanted =
      pow(( pow(rho_b,1.0/3.0) - M_WD/(4.0*Ab*z0)*x3/(x3+z0) ), 3.0);
    for (j=js; j<=je; j++) {
      for (i=is; i<=ie; i++) {
        cc_pos(pG,i,j,k,&x1,&x2,&x3);
        pG->U[k][j][i].M3 = factorM*pG->U[k][j][i].M3;
        pG->U[k][j][i].d  = dwanted + ( pG->U[k][j][i].d-dwanted )*factord;
        float ewanted = ( Ab * pow(pG->U[k][j][i].d, 4.0/3.0) )/Gamma_1;
        ewanted +=
          0.5*(SQR(pG->U[k][j][i].M1)  + SQR(pG->U[k][j][i].M2))/dwanted;
        pG->U[k][j][i].E  = ewanted + ( pG->U[k][j][i].E-ewanted )*factord;
      }
    }
  }
  // UPPER REGION (a)
  if (x3 >= rzone_lo && x3 < rzone_hi) {
    float factord = exp(-0.001*(fabs(rzone_hi-rzone_lo)+0.000001)
                              /(fabs(      x3-rzone_hi)+0.000001));
    float factorM = factord;
    float dwanted =
      pow(( pow(rho_a,1.0/3.0) - M_WD/(4.0*Aa*z0)*x3/(x3+z0) ), 3.0);
    for (j=js; j<=je; j++) {
      for (i=is; i<=ie; i++) {
        cc_pos(pG,i,j,k,&x1,&x2,&x3);
        pG->U[k][j][i].M3 = factorM*pG->U[k][j][i].M3;
        pG->U[k][j][i].d  = dwanted + ( pG->U[k][j][i].d-dwanted )*factord;
        float ewanted = ( Aa * pow(pG->U[k][j][i].d, 4.0/3.0) )/Gamma_1;
        ewanted +=
          0.5*(SQR(pG->U[k][j][i].M1)  + SQR(pG->U[k][j][i].M2))/dwanted;
        pG->U[k][j][i].E  = ewanted + ( pG->U[k][j][i].E-ewanted )*factord;
      }
    }
  }
}
```