The Role of Physical and Numerical Viscosity in Hydrodynamical Instabilities

Master Thesis at the Faculty of Physics
Ludwig-Maximilians-Universität Munich

Submitted by
Tirso Marín Gilabert

Supervised by
Dr. Milena Valentini
Dr. Ulrich Steinwandel
PD Dr. Klaus Dolag

Munich, 26th of October 2021

Die Rolle der physikalischen und numerischen Viskosität in hydrodynamischen Instabilitäten

Masterarbeit an der Fakultät für Physik
Ludwig-Maximilians-Universität München

Eingereicht von
Tirso Marín Gilabert

Betreut von
Dr. Milena Valentini
Dr. Ulrich Steinwandel
PD Dr. Klaus Dolag

München, 26.10.2021

Contents

Introduction

In astrophysics there are several systems where a continuous fluid has a velocity shear or where two fluids in contact are streaming in opposite directions and which, as time passes, evolve to a turbulent regime where the two fluids mix. This mixing process is primarily driven by the Kelvin-Helmholtz Instability (KHI), where a small perturbation in the interface between the two fluids evolves to an instability in a curly shape ending up with the mix of the two fluids. The creation of this vortex, characteristic of the KHI, plays a fundamental role in many astrophysical processes such as galaxy evolution (Agertz et al., 2009), structure formation (Burkert, 2006) or clouds moving through the circumgalactic medium (Sander and Hensler, 2021). Due to the importance of this type of instability in nature, the KHI is commonly used to test numerical codes and, in particular, new astrophysical hydrodynamics codes (Springel (2010) with AREPO, Hopkins (2015) with GIZMO or Schaal et al. (2015) with TENET). If a code can reproduce this instability properly, it is expected to properly capture the mixing and turbulence in astrophysical simulations, which is fundamental if one wants to get the most accurate results. However, in the past there has been some controversy with SPH schemes due to the fact that they couldn't reproduce well the KHI (Agertz et al., 2007; McNally et al., 2012). This problem remained unsolved until the introduction of the new Wendland kernel by Dehnen and Aly (2012) where, with an increase of the number of neighbours, the KHI was finally successfully evolved also with SPH codes (Tricco and Price, 2013; Hu et al., 2014).

The KHI is also a fundamental process in the evolution of bubbles and cold fronts in the Intracluster Medium (ICM). This type of instabilities has been observed in cold

fronts moving through the ICM (Breuer et al., 2020; Ge et al., 2020), which can lead to a disruption of the cold front. The viscosity of the ICM plays a fundamental role in the growth of the KHI and observations of these instabilities together with simulations can give us an idea of the viscosity that the ICM has, which is still an open question. By measuring the amplitude of the KHI observed and comparing these observations with simulations, one could estimate the level of suppression experienced by the evolving perturbation and, therefore, the amount of viscosity of the medium. There are several studies in this field (Roediger et al., 2013a; ZuHone et al., 2015), but still no agreement has been reached. Other works analyse the effect of the viscosity of the ICM with the evolution of buoyant bubbles from AGN jets (Sijacki and Springel, 2006; Dong and Stone, 2009).

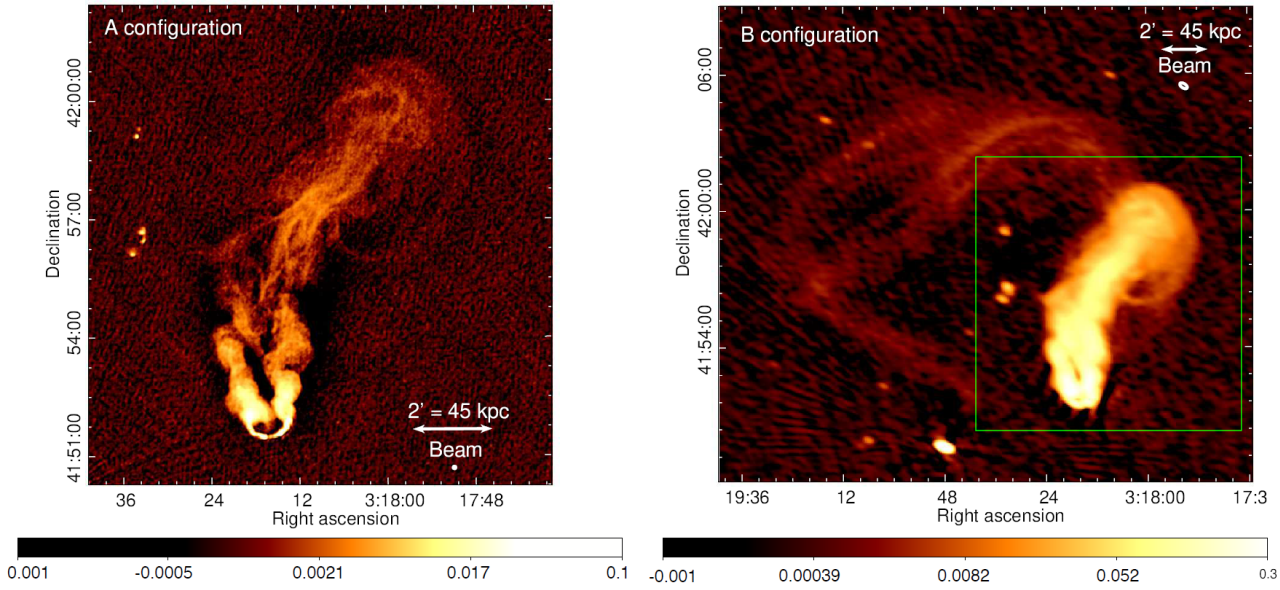

Figure 1: Galaxy NGC 1265 at 230-470 MHz. Color scales units are Jy beam^{-1}. We appreciate the KHI in the filaments of the tail of the galaxy. Image taken from Gendron-Marsolais et al. (2020).

This thesis is organised as follows: In chapter 1 we derive the equations of hydrodynamics and give a detailed description of the numerical implementation of the Smoothed Particle Hydrodynamics (SPH) and Meshless Finite Mass (MFM) schemes. In order to see how

these two numerical schemes work, we test different parameters and implementations of SPH and MFM in chapter 2. Then, in chapter 3, a deeper comparison between the two schemes and the different implementations within each code is performed. In chapter 4 we derive the equations of hydrodynamics when also the viscosity is taken into account and explain how it is implemented in our SPH codes. Then we perform an analysis with different amounts of viscosity and their effect in the evolution of the KHI. Finally, we test different initial conditions in chapter 5 to see the effect they have in triggering the KHI.

Chapter 1

The theory behind SPH and MFM

Having numerical codes that reproduce the behaviour of fluids in a realistic way is extremely important to study fluid dynamics and perform a number of experiments that cannot be done in laboratories. In this process, we need to describe the equations governing the motion of the fluids. In this first chapter we are going to derive those equations and see how they are described in the numerical schemes we are going to use: Smoothed Particle Hydrodynamics (SPH) and Meshless Finite Mass (MFM).

1.1 Equations of hydrodynamics

The particles in an ideal fluid move following the so called 'Euler equations' which describe the conservation of mass, momentum and energy. We are going to derive these equations following Landau and Lifshitz (1987) and Dullemond and Wang (2009). Let's first define the independent variables density $\rho(x, y, z, t)$, velocity $\vec{v}(x, y, z, t)$ and specific energy $e(x, y, z, t)$. To describe the fluid we could also use the pressure $P(x, y, z, t)$, which is related to the density following

$$P = A\,\rho^{\gamma}\,, \tag{1.1}$$

where A is an entropic function which is constant during adiabatic processes and γ is the adiabatic constant. It is important to note that the coordinates x, y and z refer to a point

in the space, not a specific particle of the fluid.

1.1.1 Conservation of mass

Let's now consider an arbitrary volume of the fluid V. We can define the total mass inside the volume as $\int \rho \, \mathrm{d}V$. We write the surface of this volume as S, and the unit vectors pointing outwards the volume as $\vec{n}$. Now, considering the conservation of mass, we can say that the change of the mass inside the volume is equals to the flow of mass that enters or exits the total surface:

$$\frac{\partial}{\partial t} \int \rho \, \mathrm{d}V = - \oint \rho \vec{v} \cdot \vec{n} \, \mathrm{d}S \, . \tag{1.2}$$

We now use the Gauss's theorem which relates the flux of a vector field through a surface with the divergence of this field inside the volume delimited by the surface

$$\oint \rho \vec{v} \cdot \vec{n} \, \mathrm{d}S = \int \nabla \cdot (\rho \vec{v}) \, \mathrm{d}V \, . \tag{1.3}$$

We write equation 1.2 as

$$\frac{\partial}{\partial t} \int \rho \, \mathrm{d}V = - \int \nabla \cdot (\rho \vec{v}) \, \mathrm{d}V \, , \tag{1.4}$$

and thus, we can write the continuity equation as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \, , \tag{1.5}$$

or in tensor form

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho v_k)}{\partial x_k} = 0 \, . \tag{1.6}$$

1.1.2 Conservation of momentum

For the conservation of momentum equation we can proceed in a similar way as in the conservation of mass, where now the total momentum in a volume V is given by $\int \rho \vec{v} \, \mathrm{d}V$. In this case we also have to add an additional term for the external forces applied to the volume along the surface $- \oint P \vec{n} \, \mathrm{d}S$. Thus we have

$$\frac{\partial}{\partial t} \int \rho \vec{v} \, \mathrm{d}V = - \oint \rho \vec{v} \vec{v} \cdot \vec{n} \, \mathrm{d}S - \oint P \vec{n} \, \mathrm{d}S \, . \tag{1.7}$$

In order to be able to apply the Gauss's theorem again, we need to express $P\,\vec{n}$ as $P\mathbf{I}\,\vec{n}$, where $\mathbf{I}$ is the unit tensor. By doing this change we have

$$\frac{\partial}{\partial t}\int \rho\vec{v}\,\mathrm{d}V = -\oint (\rho\vec{v}\vec{v} + \mathbf{I}P)\cdot\vec{n}\,\mathrm{d}S\,, \tag{1.8}$$

and applying the Gauss's theorem as before we get

$$\frac{\partial}{\partial t}\int \rho\vec{v}\,\mathrm{d}V = -\int \nabla\cdot(\rho\vec{v}\vec{v} + \mathbf{I}P)\,\mathrm{d}V\,. \tag{1.9}$$

Now we can write the conservation of momentum equation (also known as *Euler's equation*) as

$$\frac{\partial(\rho\vec{v})}{\partial t} + \nabla\cdot(\rho\vec{v}\vec{v} + \mathbf{I}P) = 0 \tag{1.10}$$

which is the same as

$$\frac{\partial(\rho\vec{v})}{\partial t} + \nabla\cdot(\rho\vec{v}\vec{v}) + \nabla P = 0\,, \tag{1.11}$$

or in tensor notation

$$\frac{\partial(\rho v_i)}{\partial t} + \frac{\partial}{\partial x_k}(\rho v_i v_k + \delta_{ik}P) = 0 \tag{1.12}$$

$$\frac{\partial(\rho v_i)}{\partial t} + \frac{\partial(\rho v_i v_k)}{\partial x_k} + \frac{\partial P}{\partial x_i} = 0\,. \tag{1.13}$$

We can now define the momentum flux density tensor Π_{ik} as

$$\Pi_{ik} = \rho v_i v_k + \delta_{ik}P \tag{1.14}$$

and rewrite equation 1.12 as

$$\frac{\partial(\rho v_i)}{\partial t} = -\frac{\partial\Pi_{ik}}{\partial x_k}\,. \tag{1.15}$$

Π_{ik} is a tensor of rank two that describes the ith component of the amount of momentum flowing in unit time through unit area perpendicular to the x_k-axis (Landau and Lifshitz, 1987).

If we continue operating equation 1.11

$$\rho\frac{\partial\vec{v}}{\partial t} + \vec{v}\frac{\partial\rho}{\partial t} + \rho\vec{v}\,\nabla\cdot\vec{v} + \vec{v}\,\nabla\cdot(\rho\vec{v}) + \nabla P = 0\,, \tag{1.16}$$

where the second and the fourth terms cancel out due to the continuity equation (1.5), let us write the Euler's equation as

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\ \vec{v} + \frac{1}{\rho} \nabla P = 0\,, \tag{1.17}$$

and in index notation

$$\frac{\partial v_i}{\partial t} + v_k \frac{\partial v_i}{\partial x_k} + \frac{1}{\rho} \frac{\partial P}{\partial x_i} = 0\,. \tag{1.18}$$

1.1.3 Conservation of energy

Finally, let's define the total energy inside the volume V as $\int \rho e \,\mathrm{d}V$, where e is the specific energy[1]. The change of the total energy with time is equals to the energy flowing through the surface plus the work produced from the outside. This work is given by the first law of thermodynamics and is defined as $-P\,\mathrm{d}V$, which is the surface integral of $-P\vec{v} \cdot \vec{n}\,\mathrm{d}S$. So the energy conservation equation can be written as

$$\frac{\partial}{\partial t} \int \rho e \,\mathrm{d}V = - \oint \rho e \vec{v} \cdot \vec{n}\,\mathrm{d}S - \oint P\vec{v} \cdot \vec{n}\,\mathrm{d}S \tag{1.19}$$

$$\frac{\partial}{\partial t} \int \rho e \,\mathrm{d}V = - \oint (\rho e \vec{v} + P\vec{v}) \cdot \vec{n}\,\mathrm{d}S\,. \tag{1.20}$$

and if we apply the Gauss's theorem we get

$$\frac{\partial}{\partial t} \int \rho e \,\mathrm{d}V = - \int \nabla \cdot [(\rho e + P)\ \vec{v}]\ \mathrm{d}V\,. \tag{1.21}$$

Now we can write the equation of conservation of energy as

$$\frac{\partial (\rho e)}{\partial t} + \nabla \cdot [(\rho e + P)\ \vec{v}] = 0 \tag{1.22}$$

and in tensor notation

$$\frac{\partial (\rho e)}{\partial t} + \frac{\partial}{\partial x_k} [(\rho e + P)\ v_k] = 0\,. \tag{1.23}$$

[1]The total specific energy is the addition of the specific internal energy and the specific kinetic energy: $e = u + \frac{1}{2}v^2$.

1.1.4 Lagrangian form of the equations of hydrodynamics

We have already seen the Eulerian form of the equations, which describes a fixed volume and the fluid flowing through it. The idea behind the Lagrangian form is to follow an element of fluid and see how its properties such as density, pressure or energy, change along the path. This description will also help us to understand the interpretation of the physics behind the equations. In order to write the previous equations in a Lagrangian form, we introduce the Lagrangian derivative[2]:

$$\frac{\mathrm{d}}{\mathrm{d}t} \equiv \frac{\partial}{\partial t} + \vec{v} \cdot \nabla \,. \tag{1.24}$$

Introducing it into equation 1.5, we can write the continuity equation in Lagrangian form as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{1.25}$$

$$\frac{\partial \rho}{\partial t} + \vec{v} \cdot \nabla \rho + \rho \nabla \cdot \vec{v} = 0 \tag{1.26}$$

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} + \rho \nabla \cdot \vec{v} = 0 \,. \tag{1.27}$$

This equation tells us that when the element of fluid is compressed (expanded)[3], the density of this element will increase (decrease).

The Euler's equation (1.17) is straightforward to derive:

$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} + \frac{1}{\rho} \nabla P = 0 \,, \tag{1.28}$$

which shows that the fluid element will accelerate if a force is applied to it.

For the case of the equation of energy conservation we expand the derivatives and write:

$$\rho \frac{\partial e}{\partial t} + e \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho e \vec{v}) + \nabla \cdot (P \vec{v}) = 0 \tag{1.29}$$

[2]Note that one can find the Lagrangian derivative with many different names in the literature: material derivative, co-moving derivative, convective derivative... but they all mean the same expression.

[3]A compression (expansion) corresponds to a negative (positive) value of $\nabla \cdot \vec{v}$.

$$\rho \frac{\partial e}{\partial t} + e \frac{\partial \rho}{\partial t} + e \nabla \cdot (\rho \vec{v}) + \rho \vec{v} \cdot \nabla e + \vec{v} \cdot \nabla P + P \nabla \cdot \vec{v} = 0 \,. \tag{1.30}$$

Now the second and third terms cancel out due to the continuity equation 1.5 and the first and fourth terms can be written with the Lagrangian derivative (1.24), leading to

$$\frac{\mathrm{d}e}{\mathrm{d}t} + \frac{P}{\rho} \nabla \cdot \vec{v} + \frac{1}{\rho} \vec{v} \cdot \nabla P = 0 \,, \tag{1.31}$$

which describes the variation of the total energy due to compression and external forces. If we now use the definition of the total energy $e = u + \frac{1}{2} v^2$ and calculate the derivative with respect to time

$$\frac{\mathrm{d}e}{\mathrm{d}t} = \frac{\mathrm{d}u}{\mathrm{d}t} + \vec{v} \frac{\mathrm{d}\vec{v}}{\mathrm{d}t} \,. \tag{1.32}$$

Now substituting into equation 1.31 and using the Lagrangian form of the equation of the conservation of momentum (1.28) we get that the change in the internal energy of the fluid parcel is given by

$$\frac{\mathrm{d}u}{\mathrm{d}t} + \frac{P}{\rho} \nabla \cdot \vec{v} = 0 \,. \tag{1.33}$$

This equation describes how the internal energy of the fluid increases (decreases) when the fluid is affected by an adiabatic compression (expansion).

If we make use now of the first law of thermodynamics we have

$$\mathrm{d}u = T \, \mathrm{d}s - P \, \mathrm{d} \left(\frac{1}{\rho} \right) \tag{1.34}$$

and if we compute the derivative with respect time, we get

$$\frac{\mathrm{d}u}{\mathrm{d}t} = T \frac{\mathrm{d}s}{\mathrm{d}t} - P \frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{1}{\rho} \right) \tag{1.35}$$

$$\frac{\mathrm{d}u}{\mathrm{d}t} = T \frac{\mathrm{d}s}{\mathrm{d}t} + \frac{P}{\rho} \frac{1}{\rho} \frac{\mathrm{d}\rho}{\mathrm{d}t} \,. \tag{1.36}$$

Using the continuity equation (1.27) we can write the equation above as

$$\frac{\mathrm{d}u}{\mathrm{d}t} = T \frac{\mathrm{d}s}{\mathrm{d}t} - \frac{P}{\rho} \nabla \cdot \vec{v} \,, \tag{1.37}$$

and with the equation of internal energy (1.33), the term on the left-hand side of the equation and the second term on the right-hand side cancel out, leading to

$$T \frac{\mathrm{d}s}{\mathrm{d}t} = 0\,, \tag{1.38}$$

showing that the entropy is conserved along the path of our fluid parcel. If we include this result into equation 1.36 we can also write the change of the internal energy as

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{P}{\rho^2} \frac{\mathrm{d}\rho}{\mathrm{d}t}\,. \tag{1.39}$$

1.2 SPH

Once we have defined the equations of hydrodynamics, the next step is to describe all these equations numerically in order to be able to simulate the behaviour of the fluids. In this project we are going to focus first on a SPH description of the equations (Lucy, 1977; Gingold and Monaghan, 1977), which is a Lagrangian description based on interpolating the physical quantities among the particles around. The fluid is discretized in a set of moving particles[4] with mass that move with the velocity of the fluid and describe its properties. To do the interpolation we use a weight function (kernel) that gives weight to the particles around the point we want to focus on to calculate a given fluid quantity. To see this in a clearer way, let's do an example calculating the density.

The question arises immediately: how can we compute the density from a distribution of point mass particles? The first idea we can think of is to divide the whole domain in cells and calculate the density by dividing the mass inside these cells by the volume of each cell. The problem of doing this is that you can oversample (undersample) dense (sparse) regions and there is also a loss of accuracy. We could think that, instead of establishing a fixed grid, we could draw a sphere around the desired point and compute the density by dividing the mass of the particles inside the sphere by the total volume of the sphere

$$\rho_i = \frac{\sum_{j=1}^{N} m_j}{\frac{4}{3}\,\pi\, R^3}\,. \tag{1.40}$$

[4]Note that when we refer to *particles* we don't mean real particles, but comoving coordinates.

With this approach we could define a bigger sphere for sparse areas and a smaller one for denser areas, reducing this way the over and undersampling. The problem here is that the sphere is very sensitive to particles near the edge and can introduce some noise. Whether a particle is just near the border but outside the sphere can really make a difference compared to the case in which this particle is near the border but inside. So in order to fix this problem, we introduce a kernel that gives a higher weight to the particles that are close to the point where we are calculating the density and a lower weight to the ones that are further away (Price, 2012). This way we can calculate the density at the desired point by summing up the weighted masses of all the particles inside a volume around it:

$$\rho_i = \sum_{j=1}^{N} m_j \, W(|r_i - r_j|, h_i) \,, \tag{1.41}$$

where here $W(|r_i - r_j|, h_i)$ is the kernel, which depends on the distance between the particles and the smoothing length h_i. This smoothing length determines the rate of fall-off of the kernel and is defined as

$$h = \eta \left(\frac{m}{\rho} \right)^{1/d} , \tag{1.42}$$

where η specifies the smoothing length in units of the average particle spacing and d indicates the number of spatial dimensions of the system. The number of neighbours N (see the definition of *neighbours* below) is usually kept constant by choosing the appropriate h.

The kernel satisfies the normalisation

$$\int W(|r_i - r_j|, h_i) \, \mathrm{d}V = 1 \,, \tag{1.43}$$

but it also must satisfy the following properties in order to be a suitable function for our purpose: It must be positive, decrease monotonically and have smooth derivatives; it must be symmetric and it must flatten in the center so that it doesn't overweight the density of a close neighbour. With all these properties, the most suitable function we can think of is a Gaussian. The problem with the Gaussian function is that its region of influence is infinite, so the number of interactions will be N^2, where N is the total number of the

particles inside the domain. In order to decrease the computational cost, we want to find a Gaussian-like kernel with compact support, which means that the kernel is truncated at a finite radius. The particles inside this finite radius (neighbours) are the ones taken into account for the computation, reducing the number of interactions to $N_{\mathrm{Ngb}} \cdot N$ now. This is fulfilled by the B-spline functions (Schoenberg, 1946; Monaghan and Lattanzio, 1985), which are generated as the Fourier transform

$$M_n(x,h) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\frac{\sin(kh/2)}{kh/2} \right]^n \cos(kx)\, \mathrm{d}k\,, \tag{1.44}$$

and gives better approximations to a Gaussian the higher the order n is. It is continuous up to the $(n-2)$th derivative and, since we need at least the first and second derivative to be continuous, the lowest order we can use is $n = 4$, which corresponds to the M_4 cubic spline kernel (Price, 2012). It is also the most common B-spline kernel used in SPH and is defined as

$$W_{ij}(r,h) = \frac{C_{norm}}{h^3}\, f(q)\,, \tag{1.45}$$

where C_{norm} is the normalisation constant and $f(q)$ is a dimensionless function of $q = |r_i - r_j|/h$. In the case of the cubic spline kernel in 3D: $C_{norm} = 1/\pi$ and

$$f(q) = \begin{cases} \frac{1}{4}(2-q)^3 - (1-q)^3, & 0 \le q < 1; \\ \frac{1}{4}(2-q)^3, & 1 \le q < 2; \\ 0. & q \ge 2, \end{cases} \tag{1.46}$$

with the compact support being $R = 2h$.

Once we have defined the kernel, we can write the derivative of ρ with respect to time as

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = \frac{1}{\Omega_i} \sum_{j=1}^{N} m_j \left(\vec{v_i} - \vec{v_j}\right) \nabla W_{ij}(r, h_i)\,, \tag{1.47}$$

where Ω_i is a term that depends on the derivative of the kernel with respect to the smoothing length, defined as

$$\Omega_i = \left[1 - \frac{\partial h_i}{\partial \rho_i} \sum_{j=1}^{N} m_j \frac{\partial W_{ij}(r,h_i)}{\partial h_i} \right]. \tag{1.48}$$

The gradient of the density is given by

$$\nabla\rho = \frac{\partial\rho_i}{\partial r_k} = \frac{1}{\Omega_i}\sum_{j=1}^{N} m_j\,\nabla_k W_{ij}(r,h_i)(\delta_{ik}-\delta_{jk})\,. \tag{1.49}$$

Once we know how to calculate the derivative of the density with respect to time, using the continuity equation (1.27) we get that

$$(\nabla\cdot\vec{v})_i = \frac{1}{\Omega_i\rho_i}\sum_{j=1}^{N} m_j\,(\vec{v_j}-\vec{v_i})\,\nabla W_{ij}(r,h_i)\,. \tag{1.50}$$

Making use of equation 1.39 we can also get the variation of the internal energy

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \frac{P_i}{\Omega_i\rho_i^2}\sum_{j=1}^{N} m_j\,(\vec{v_i}-\vec{v_j})\,\nabla W_{ij}(r,h_i)\,. \tag{1.51}$$

In order to derive the acceleration we are going to use the Lagrangian of a perfect fluid (Eckart, 1960), given by

$$L = \int\left[\frac{1}{2}\rho v^2 - \rho u(\rho,s)\right]\mathrm{d}V \tag{1.52}$$

and if we discretize it, we get

$$L = \sum_{j=1}^{N} m_j\left[\frac{1}{2}v_j^2 - u_j(\rho_j,s_j)\right]\,. \tag{1.53}$$

Now, the equations of motion for the particle i are given by the Euler-Lagrange equations

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial\vec{v_i}}\right) - \frac{\partial L}{\partial r_i} = 0\,. \tag{1.54}$$

The first term is equal to the variation of momentum of the particle

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial\vec{v_i}}\right) = \frac{\mathrm{d}}{\mathrm{d}t}(m_i\vec{v_i})\,, \tag{1.55}$$

while the second term, assuming that the entropy s is constant, can be written as

$$\frac{\partial L}{\partial r_i} = -\sum_{j=1}^{N} m_j\,\left.\frac{\partial u_j}{\partial\rho_j}\right|_s\frac{\partial\rho_j}{\partial r_i}\,. \tag{1.56}$$

From equation 1.39 we can see that $\partial u/\partial\rho = P/\rho^2$ and we already know $\partial\rho/\partial r$ (1.49), so we can write the previous equation as

$$\frac{\partial L}{\partial r_i} = -\sum_{j=1}^{N} m_j\frac{P_j}{\Omega_j\rho_j^2}\sum_{k=1}^{N} m_k\,\nabla_i W_{jk}(r,h_j)(\delta_{ji}-\delta_{ki})\,. \tag{1.57}$$

So the equation of motion from the Euler-Lagrange equations (1.54) become

$$\frac{\mathrm{d}\vec{v_i}}{\mathrm{d}t} = -\sum_{j=1}^{N} m_j \left[\frac{P_i}{\Omega_i \rho_i^2}\nabla_i W_{ij}(r,h_i) + \frac{P_j}{\Omega_j \rho_j^2}\nabla_i W_{ij}(r,h_j)\right] . \tag{1.58}$$

If we use equation 1.32, together with the expressions we already have for $\mathrm{d}u/\mathrm{d}t$ and $\mathrm{d}v/\mathrm{d}t$, we get the variation of the total energy

$$\frac{\mathrm{d}e_i}{\mathrm{d}t} = \sum_{j=1}^{N} m_j \left[\frac{P_i \vec{v_j}}{\Omega_i \rho_i^2}\nabla_i W_{ij}(r,h_i) + \frac{P_j \vec{v_i}}{\Omega_j \rho_j^2}\nabla_i W_{ij}(r,h_j)\right] . \tag{1.59}$$

Another option often used in SPH codes for ideal fluids is to use the entropic function $A(s)$ defined in equation 1.1 as the evolved variable, which evolves according to

$$\frac{\mathrm{d}A(s)}{\mathrm{d}t} = \frac{\gamma-1}{\rho^{\gamma-1}} T \frac{\mathrm{d}s}{\mathrm{d}t} = \frac{\gamma-1}{\rho^{\gamma-1}}\left(\frac{\mathrm{d}u}{\mathrm{d}t} - \frac{P}{\rho^2}\frac{\mathrm{d}\rho}{\mathrm{d}t}\right) = \frac{\gamma-1}{\rho^{\gamma-1}}\left(\frac{\mathrm{d}u}{\mathrm{d}t}\right)_{\mathrm{diss}} , \tag{1.60}$$

where the subscript "diss" refers to the dissipative part of the evolution of thermal energy. Then the internal energy can be computed as

$$u = \frac{\rho^{\gamma-1}}{\gamma-1} A . \tag{1.61}$$

Since $\mathrm{d}A/\mathrm{d}t$ is equal to 0 when there is not dissipation, using the entropic function A has the advantage that the evolution is independent of the time-integration algorithm (Price, 2012).

As a summary, the SPH code computes the density, velocity and internal energy of the particles as follows:

$$\rho_i = \sum_{j=1}^{N} m_j W_{ij}(r,h_i) , \tag{1.62}$$

$$\frac{\mathrm{d}\vec{v_i}}{\mathrm{d}t} = -\sum_{j=1}^{N} m_j \left[\frac{P_i}{\Omega_i \rho_i^2}\nabla_i W_{ij}(r,h_i) + \frac{P_j}{\Omega_j \rho_j^2}\nabla_i W_{ij}(r,h_j)\right] , \tag{1.63}$$

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \frac{P_i}{\Omega_i \rho_i^2}\sum_{j=1}^{N} m_j \left(\vec{v_i} - \vec{v_j}\right)\nabla W_{ij}(r,h_i) . \tag{1.64}$$

Once we have derived the basic equations of hydrodynamics in SPH, we are going to write now a general description by considering a field $F(r)$ defined at every position in

space. We can interpolate this quantity using

$$F(r) = \int F(r')\, W(r - r', h)\, \mathrm{d}r' \tag{1.65}$$

and if we discretize it, we get

$$F(r_i) \simeq \sum_{j=1}^{N} m_j \frac{F_j}{\rho_j}\, W(|r_i - r_j|, h)\,, \tag{1.66}$$

which corresponds to 1.62 if $F(r_i) = \rho(r_i)$. Using this we can write a general formulation for any physical quantity. For the gradient we just take the derivative of equation 1.66 and get

$$\nabla F(r_i) \simeq \sum_{j=1}^{N} m_j \frac{F_j}{\rho_j}\, \nabla W(|r_i - r_j|, h)\,. \tag{1.67}$$

If $F(r)$ is a vector field, we can also write the two following expressions

$$\nabla \cdot \vec{F}(r_i) \simeq \sum_{j=1}^{N} m_j \frac{\vec{F}_j}{\rho_j} \cdot \nabla W(|r_i - r_j|, h)\,, \tag{1.68}$$

$$\nabla \times \vec{F}(r_i) \simeq -\sum_{j=1}^{N} m_j \frac{\vec{F}_j}{\rho_j} \times \nabla W(|r_i - r_j|, h)\,. \tag{1.69}$$

Although this can lead to poor gradient estimates, these expressions can give us a general idea for interpreting SPH.

1.3 MFM

Since the SPH equations are derived directly from the Hamiltonian, the discrete system of particles is exactly conservative. This means that it has zero numerical dissipation, which can be a great advantage because the numerical solution will not diffuse entropy or energy artificially. However, this can become a problem when dissipation is required physically. When we have shocks or fluid mixing that tend to increase the entropy of the system, an artificial diffusion must be added explicitly[5] (Price, 2011).

[5]This will be treated in more detail in chapter 2.

When using a Godunov approach (i.e. grid methods), instead of discretizing the fluid in point mass particles as we did in SPH, we discretize the domain into cells that contain the information of the fluid in that particular region. Then, since there is a discontinuity between the different cells, one must solve a Riemann problem consisting of two piece-wise constant states that meet at a plane (Springel, 2014). The fact of solving a Riemann problem between each pair of cells avoids the need of adding numerical dissipation artificially. Therefore, a solution would be to derive a mixture between the two methods: a Godunov-like formulation to be a consistent Lagrangian meshless scheme from the basic principles (Gaburov and Nitadori, 2011), the so called "Godunov SPH". This would have the conservation properties of SPH, but the artificial dissipation would come up naturally, like in grid methods.

Hopkins (2015) introduced a method based on kernel discretization of the volume to a higher-order gradient estimator and a Riemann solver between the particles inside that volume, the Meshless Finite Mass (MFM) method. This allows the particles to move with the flow; energy, mass and momentum are conserved and there is no need of artificial diffusion terms.

For the derivation of the meshless equations of motion, we follow Hopkins (2015). First we write the equations of hydrodynamics described in 1.1 as a system of hyperbolic partial differential equations in a frame moving with velocity $\vec{v}_{\text{frame}}$

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{F} - \vec{v}_{\text{frame}} \otimes \mathbf{U}) = 0\,, \tag{1.70}$$

where $\cdot$ is the inner product and $\otimes$ the outer product. $\mathbf{U}$ is the "state vector" of the conserved quantities

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho\vec{v} \\ \rho e \end{pmatrix} = \begin{pmatrix} \rho \\ \rho\vec{v} \\ \rho u + \frac{1}{2}\rho|\vec{v}|^2 \end{pmatrix} \tag{1.71}$$

and the tensor $\mathbf{F}$ is given by

$$\mathbf{F} = \begin{pmatrix} \rho\vec{v} \\ \rho\vec{v} \otimes \vec{v} + P\mathbf{I} \\ (\rho e + P)\vec{v} \end{pmatrix} . \tag{1.72}$$

In order to solve it, we are going to employ the discontinuous Galerkin method, where we introduce a weak formulation of 1.70 by multiplying a test function ϕ to equation 1.70 (Luo et al., 2008). We now integrate over the volume V and using an integration by parts of $\phi(\nabla \cdot \mathbf{F})$ we get

$$\int \left(\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t}\,\phi - \mathbf{F} \cdot \nabla\phi \right) \mathrm{d}V + \oint (\mathbf{F}\phi) \cdot \vec{n}\,\mathrm{d}S = 0\,, \tag{1.73}$$

where we have used the Lagrangian derivative (1.24) with the frame velocity and the Gauss's theorem (1.3). $\phi(\vec{r}, t)$ is taken to be a differentiable Lagrangian function ($\mathrm{d}\phi/\mathrm{d}t = 0$). Assuming that the fluxes $\mathbf{F}$ and/or ϕ vanish at infinity, we can set the second term on the left-hand side to zero and write equation 1.73 as

$$\frac{\mathrm{d}}{\mathrm{d}t} \int \mathbf{U}\phi\,\mathrm{d}V - \int \mathbf{F} \cdot \nabla\phi\,\mathrm{d}V = 0\,. \tag{1.74}$$

For discretizing this integral, we consider a differential volume $\mathrm{d}V$ and we partition it fractionally among the neighbours assigning a weight to each of them with a kernel $W(|\vec{r} - \vec{r_i}|, h)$ as it is done in SPH. Each fraction of the volume is given by

$$\psi_i(\vec{r}) = \frac{W(|\vec{r} - \vec{r_i}|, h)}{\sum_{j=1}^{N} W(|\vec{r} - \vec{r_j}|, h)}\,. \tag{1.75}$$

This weighting function indicates how the partition of the volume is made at any point $\vec{r}$ among the volumes associated with the tracer i (Hopkins, 2015). All the properties we explained for the SPH kernels apply to this kernel as well (see section 1.2). If instead of this weighted partition, we had divided the volume with a Voronoi mesh, we would have obtained the moving-mesh method codes like AREPO (Springel, 2010). Applying this definition of the volume partition to an arbitrary function $f(\vec{r})$ to second order accuracy we get

$$\int f(\vec{r})\,\mathrm{d}V = \sum_i \int f(\vec{r})\psi_i\,\mathrm{d}V = \sum_i f_i(\vec{r_i}) \int \psi_i\,\mathrm{d}V + \mathcal{O}(h_i^2) = \sum_i f_i V_i + \mathcal{O}(h_i^2)\,, \tag{1.76}$$

where $V_i = \int \psi(\vec{r})\,\mathrm{d}V$ is the "effective volume" of the particle i. If we now apply this to equation 1.74, we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_i V_i \mathbf{U}_i \phi_i - \sum_i V_i \mathbf{F}_i \cdot (\nabla\phi)_{\vec{r}=\vec{r_i}} = \sum_i \left[\phi_i \frac{\mathrm{d}}{\mathrm{d}t}(V_i \mathbf{U}_i) - V_i \mathbf{F}_i \cdot (\nabla\phi)_{\vec{r}=\vec{r_i}}\right] = 0\,, \quad (1.77)$$

where $(\nabla\phi)_{\vec{r}=\vec{r_i}}$ is the gradient of ϕ evaluated at $\vec{r} = \vec{r_i}$.

For the estimate of the gradient, we use locally-centered least-squares matrix gradient operators, which can exactly reproduce polynomial functions across the particles independently on their spatial configuration. For a second-order accuracy we have

$$(\nabla f)_i^\alpha = \sum_j \sum_{\beta=1}^{\beta=\nu} (f_j - f_i)\mathbf{B}_i^{\alpha\beta}(\vec{r_j} - \vec{r_i})^\beta \psi_j(\vec{r_i}) + \mathcal{O}(h_i^2) = \sum_j (f_j - f_i)\tilde{\psi}_j^\alpha(\vec{r_i}) \quad (1.78)$$

$$\tilde{\psi}_j^\alpha(\vec{r_i}) = \sum_{\beta=1}^{\beta=\nu} \mathbf{B}_i^{\alpha\beta}(\vec{r_j} - \vec{r_i})^\beta \psi_j(\vec{r_i})\,. \quad (1.79)$$

The matrix $\mathbf{B}_i$ is evaluated at each i by taking the inverse of matrix $\mathbf{E}_i$

$$\mathbf{B}_i = \mathbf{E}_i^{-1} \quad (1.80)$$

$$\mathbf{E}_i^{\alpha\beta} = \sum_j (\vec{r_j} - \vec{r_i})^\alpha (\vec{r_j} - \vec{r_i})^\beta \psi_j(\vec{r_i})\,. \quad (1.81)$$

Making use of this computation of the gradient (equation 1.78), the second term of equation 1.77 can be written as

$$\sum_i V_i \mathbf{F}_i^\alpha \cdot (\nabla\phi)_i^\alpha = \sum_i \sum_j V_i \mathbf{F}_i^\alpha (\phi_j - \phi_i)\tilde{\psi}_j^\alpha(\vec{r_i}) = -\sum_i \phi_i \sum_j (V_i \mathbf{F}_i^\alpha \tilde{\psi}_j^\alpha(\vec{r_i}) - V_j \mathbf{F}_j^\alpha \tilde{\psi}_j^\alpha(\vec{r_j}))\,, \quad (1.82)$$

so we can write equation 1.77 as

$$\sum_i \phi_i \left(\frac{\mathrm{d}}{\mathrm{d}t}(V_i \mathbf{U}_i) + \sum_j [V_i \mathbf{F}_i^\alpha \tilde{\psi}_j^\alpha(\vec{r_i}) - V_j \mathbf{F}_j^\alpha \tilde{\psi}_j^\alpha(\vec{r_j})]\right) = 0\,. \quad (1.83)$$

The inside of the parenthesis must be zero, so we get an expression independent of the test function ϕ:

$$\frac{\mathrm{d}}{\mathrm{d}t}(V_i \mathbf{U}_i) + \sum_j [V_i \mathbf{F}_i^\alpha \tilde{\psi}_j^\alpha(\vec{r_i}) - V_j \mathbf{F}_j^\alpha \tilde{\psi}_j^\alpha(\vec{r_j})] = 0\,. \quad (1.84)$$

If we now calculate $\mathbf{F}$ at positions i and j independently, we will need artificial dissipation, as happens in SPH. Instead of that, we calculate the flux between these two particles by solving a Riemann problem, which automatically includes the dissipation terms. We write the expression above in terms of the flux $\tilde{\mathbf{F}}_{ij}$ between the two particles

$$\frac{\mathrm{d}}{\mathrm{d}t}(V_i \mathbf{U}_i) + \sum_j \tilde{\mathbf{F}}^{\alpha}_{ij}[V_i \tilde{\psi}^{\alpha}_j(\vec{r_i}) - V_j \tilde{\psi}^{\alpha}_j(\vec{r_j})] = 0\,. \tag{1.85}$$

Finally, we define $\vec{A_{ij}} = |A_{ij}|\hat{A}_{ij}$, where $\vec{A^{\alpha}_{ij}} = V_i \tilde{\psi}^{\alpha}_j(\vec{r_i}) - V_j \tilde{\psi}^{\alpha}_j(\vec{r_j})$ and the equation becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}(V_i \mathbf{U}_i) + \sum_j \tilde{\mathbf{F}}_{ij} \cdot \vec{A_{ij}} = 0\,. \tag{1.86}$$

The temporal evolution of the particle-volume integrated value of the conserved quantity of particle i depends on the sum of the fluxes $\tilde{\mathbf{F}}_{ij}$ through an effective area $\vec{A_{ij}}$.

Since the fluxes are computed directly from the conserved quantities between the particles, the conserved quantities described by the equations of hydrodynamics (mass, linear momentum and energy) will be conserved[6].

Mainly, SPH is a Lagrangian scheme that discretize the fluid in particles. The evolution is computed following the motion of the particles with the fluid and the physical quantities are calculated interpolating among the particles around. This allows a perfect conservation of the energy and entropy with zero intrinsic dissipation. On the other hand, the MFM schemes also follow the motion of the particles, which allows the conservation of energy; but the computation of the physical quantities is done by solving a Riemann problem between the interacting particles. This produces diffusion, which is needed in many physical processes, but it is also more expensive computationally.

[6]For a more detailed explanation of the scheme, see Gaburov and Nitadori (2011) and Hopkins (2015).

Chapter 2

Selection of the parameters

Once we have explained the equations that describe the movement of the fluids and how we solve those equations using numerical schemes (in our case SPH and MFM), the next step in the process is to test the codes. In order to do it, we need tests of which we know the solution. If we are able to reproduce the results expected in those tests, we could rely on the code for more complex systems. In this project we are testing the SPH and the MFM codes with the Kelvin-Helmholtz Instability (KHI), a really common test in hydrodynamics to see if the codes perform well or they need some improvements.

As we have mentioned in the introduction, there has been some controversy during the past years due to the fact that people couldn't develop the KHI using SPH codes (Agertz et al., 2007; McNally et al., 2012) and, since these kind of instabilities are key in fluid mixing, getting them grow became a challenge. In this chapter we are going to perform different tests using OpenGadget3 in order to see which parameters are necessary to reproduce the KHI. We are going to start from a similar version of the so called 'standard' SPH (Springel, 2005) and we are going to add different improvements to the code as it is done in Beck et al. (2015), explaining every different implementation we consider. The aim of this chapter is to be able to determine the parameters we need to implement within our code in order to reproduce the instability and test the impact that the addition or lack of these parameters has in our simulations. Additionally, we are also going to try to develop

the instability using MFM codes, testing them in a similar way to SPH. In this first chapter we are going to analyse the growth of the instability from a qualitatively point of view, checking that the shape corresponds to the one we expect. In the following chapters we are going to analyse the intrinsic properties of each code quantitatively and compare them to try to determine the most appropriate parameters to use.

2.1 Kelvin-Helmholtz Instability

The first thing we must do is to introduce the instability we are going to analyse. The KHI arises when two fluids in contact move in opposite directions, where the shear velocity, and possibly the density, changes discontinuously at the interface. This shear flow is unstable and a small perturbation causes exponential growth of velocity orthogonal to the flow. This leads into vortices that curl back upon themselves forming the typical KHI rolls (see figure 2.1). Once the instability becomes non-linear, secondary instabilities appear generating turbulence and mixing the two fluids along the interface (McNally et al., 2012; Tricco, 2019).

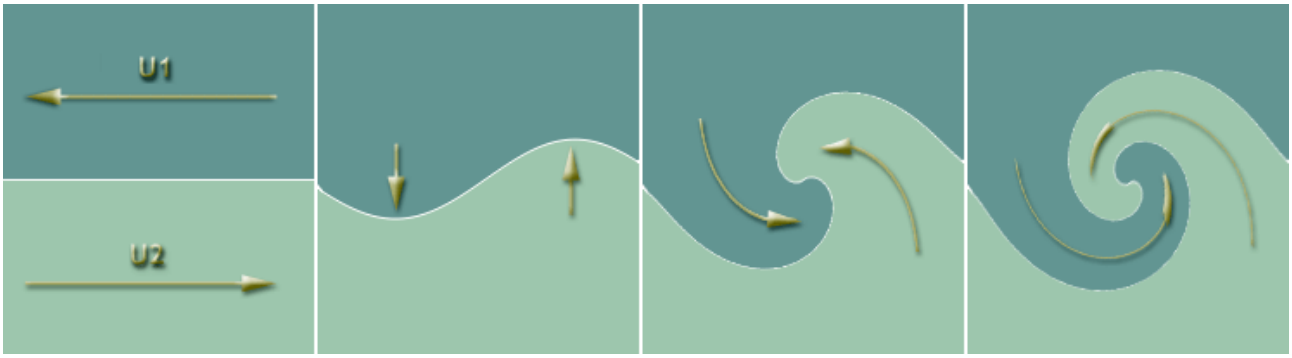

Figure 2.1: Scheme of how the KHI is triggered (modified image taken from KHI cloud structure).

From a linear analysis in two dimensions, one can derive that the time evolution of the

y-velocity of the perturbation evolves exponentially ($\sim \exp[i \cdot n \cdot t]$)[1]. n is the mode of the perturbation and is given by

$$n = \left[k^2(\Delta v_x)^2(\alpha_2 - \alpha_1)\right] + i\left[\frac{\nu k^2}{2} \pm \sqrt{\frac{\nu^2 k^4}{4} + 4k^2(\Delta v_x)^2\alpha_1\alpha_2}\right], \tag{2.1}$$

where k is the wavenumber of the perturbation, Δv_x is the difference between the two velocities, ν is the viscosity and α_1 and α_2 are defined as

$$\alpha_1 = \frac{\rho_1}{\rho_1 + \rho_2}, \qquad \alpha_2 = \frac{\rho_2}{\rho_1 + \rho_2}. \tag{2.2}$$

The real part of equation 2.1 describes the oscillation provoked by the perturbation (which is not of interest here), while the imaginary part determines the growth or decay of the instability damped by a viscosity ν. The positive solution of the square root leads to an exponential decay, whereas the negative solution to and exponential growth, which is the case of the KHI (Junk et al., 2010).

2.2 Initial conditions

To set up the initial conditions for the simulations, we follow the ones suggested by Beck et al. (2015). We create a 3D box with 774144 particles of equal mass ($m = 3.13 \cdot 10^{-8}$) set up using a cubic lattice with periodic boundary conditions. The size of the box in internal units[2] is $\Delta x = 256$, $\Delta y = 256$ and $\Delta z = 8$ and the domain satisfies:

$$\rho, T, v_x = \begin{cases} \rho_1, T_1, v_1 & |y| < 64 \\ \rho_2, T_2, v_2 & |y| > 64 \end{cases} \tag{2.3}$$

where the densities are $\rho_1 = 6.26 \cdot 10^{-8}$ and $\rho_2 = 3.13 \cdot 10^{-8}$; the temperatures $T_1 = 2.5 \cdot 10^6$ and $T_2 = 5 \cdot 10^6$ and the x-velocities $v_1 = -40$ and $v_2 = 40$. The density and temperature ratio given by $R_\rho = \rho_1/\rho_2 = T_2/T_1$ is constant and equals to 2, ensuring a pressure equilibrium in the system. With these initial conditions we get a Mach number of $M_1 = v_1/c_1 \approx 0.23$ for the first fluid and $M_2 = -v_2/c_2 \approx 0.17$ for the second one.

[1] For a complete derivation see Junk et al. (2010).

[2] The conversion from internal units to physical units can be seen in table 2.1.

In order to trigger the instability we add a small perturbation in the y-velocity at $y_{\text{Int}} = \pm 64$ (equation 2.4), similar to the one introduced by Read et al. (2010) but varying the values so it can be adapted to our initial conditions:

$$v_y = -\delta v_y \left[\sin\left(\frac{2\pi(x+\lambda/2)}{\lambda}\right) \exp\left(-\left(\frac{y-y_{\text{Int}}}{\sigma}\right)^2\right) + \sin\left(\frac{2\pi x}{\lambda}\right) \exp\left(-\left(\frac{y+y_{\text{Int}}}{\sigma}\right)^2\right)\right] \tag{2.4}$$

where $\lambda = 128$ is the wavelength of the perturbation, $\delta v_y = |v_x|/10 = 4$ is the amplitude of the perturbation and $\sigma = 0.2\lambda$ is a scale parameter to control the width of the perturbation layer (Junk et al., 2010).

With these initial conditions, we can calculate the dynamical timescale of the perturbation with equation 2.5 (see derivation in appendix A)

$$\tau_{KH} = \frac{\lambda}{\Delta v_x} \frac{(\rho_1 + \rho_2)}{(\rho_1\, \rho_2)^{1/2}}\,. \tag{2.5}$$

With our values we get a growth time of $\tau_{KH} = 3.39$. This means that the velocity of the KHI will grow exponentially until approximately $t = 3.39$.

Physical Magnitudes	Mass	Time	Length	Temperature
Physical Units (Code Units)	$10^{10}\,\mathrm{M}_\odot$	$1\,\frac{\text{kpc}}{\text{km}}\text{s}$	$1\,\text{kpc}$	$1\,\text{K}$
Physical Units (cgs)	$1.989 \cdot 10^{43}\,\text{g}$	$3.086 \cdot 10^{16}\,\text{s}$	$3.086 \cdot 10^{21}\,\text{cm}$	$1\,\text{K}$

Table 2.1: Conversion of one unit from internal to physical units. In this text we will always refer to internal units.

2.3 Smoothed Particle Hydrodynamics

Once we have defined our initial conditions in order to reproduce the Kelvin-Helmholtz Instability, we can start running the simulations. We are going to start from a basic SPH code and we are going to keep adding different features to improve it in order to get the results we expect. In each step we are going to explain how that implementation works

and why it is relevant for the correct operation of the code. In this project we are going to use OpenGadget3 (Beck et al., 2015), which is an improved version of Gadget2 (Springel, 2005).

2.3.1 Standard SPH

As we have already mentioned, the first code we use is a similar version to the 'standard' SPH implemented in Gadget2. We don't use the actual 'standard' SPH, but instead of it we have disabled some features already implemented in Gadget3 in order to have a similar behaviour. In this version, we are using a cubic spline kernel without bias correction and 64 neighbours. This version also uses a constant viscosity, no conductivity and no wake-up[3] (in the next sections we will explain each of these features).

After running this first simulation, we get what we see in figure 2.2.

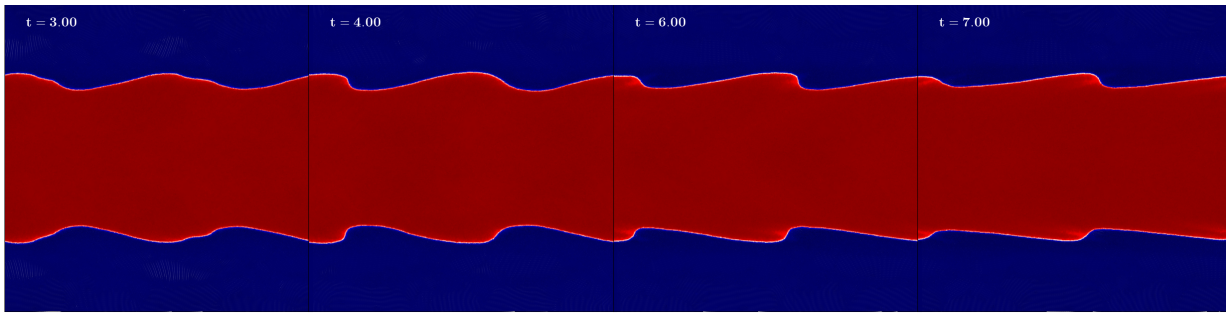

Figure 2.2: Colormap of the density for the basic SPH at $t = 3$, $t = 4$, $t = 6$ and $t = 7$.

Under these parameters the instability does not develope, as it was suggested in the literature already mentioned. The results show a small growth of the instability, but much smaller than the one expected. Also, the mixing is nonexistent due to the artificial surface tension between the two fluids, meaning that this basic SPH is not good enough to simulate the mixing of two fluids.

[3]For a more detailed description of the code, see Springel (2005) and Beck et al. (2015).

2.3.2 Addition of artificial conductivity

One of the best characteristics of SPH is that, since it is built from the Lagrangian, it conserves energy and entropy intrinsically and doesn't diffuse or dissipate energy artificially, which doesn't happen in grid codes (see sections 1.2 and 1.3). Unlike SPH, in grid codes the discretization of the domain and the computation of the quantities between the cells solving a Riemann problem introduces a numerical diffusion (Springel, 2014). Although this is a huge advantage for SPH, it can also be a big disadvantage. If we don't treat the discontinuities properly, we will never be able to reproduce the correct mix of two fluids (see figure 2.2). When the two fluids in the KHI mix, the entropy of the system tends to increase and, if the code doesn't allow this, these fluids will never mix. SPH has zero intrinsic (numerical) dissipation, which means that it must be added artificially.

In order to solve this problem, Price (2008) introduced an artificial conductivity (AC), which removes the surface tension and allows the mix between the two fluids. If we implement this artificial conductivity in our code, we can see that the surface tension is no longer a problem and the two fluids start mixing (figure 2.3).

In OpenGadget3, the AC implemented can be time dependent, as described in Beck et al. (2015). The variation of internal energy due to AC is given by

$$\left.\frac{\mathrm{d}u_i}{\mathrm{d}t}\right|_{\mathrm{cond}} = \sum_{j=1}^{N} \frac{m_j}{\rho_{ij}} (u_j - u_i)\, \alpha_{ij}^c v_{ij}^{\mathrm{sig},c} \overline{F}_{ij}\,, \tag{2.6}$$

where $\overline{F}_{ij} = (F_{ij}(h_i) + F_{ij}(h_j))/2$ is the symmetrised scalar part of the kernel gradient terms $\nabla_i W_{ij}(h_i) = F_{ij}\,\hat{r}_{ij}$; $\alpha_{ij}^c = (\alpha_i^c + \alpha_j^c)/2$ is the symmetrised conduction coefficient and $v_{ij}^{\mathrm{sig},c}$ is the signal velocity. The signal velocity depends on the pressure gradient, as suggested by Price (2008)

$$v_{ij}^{\mathrm{sig},c} = \sqrt{\frac{|P_i - P_j|}{\rho_{ij}}}\,. \tag{2.7}$$

The AC coefficient is defined as

$$\alpha_i^c = \frac{h_i}{2} \frac{|\nabla u|_i}{|u_i|}\,, \tag{2.8}$$

where the time dependence is given by the internal energy[4] and the gradient of the internal energy, which is computed using

$$(\nabla u)_i = \frac{1}{\rho_i} \sum_{j=1}^{N} m_j \, (u_j - u_i) \nabla_i W_{ij} \, . \tag{2.9}$$

When α_i^c is bigger than a threshold value (in our case $\alpha_{\max} = 1.0$), the value of α_i^c is set to $\alpha_{\max}$; while if it is smaller than a minimum value (we set $\alpha_{\min} = 0$, so in our case there was not minimum value), α_i^c is set to $\alpha_{\min}$.

However, in our simulations we set a constant artificial conductivity (unless otherwise specified, see section 3.5.1) with an AC coefficient of $\alpha_i^c = 1.0$.

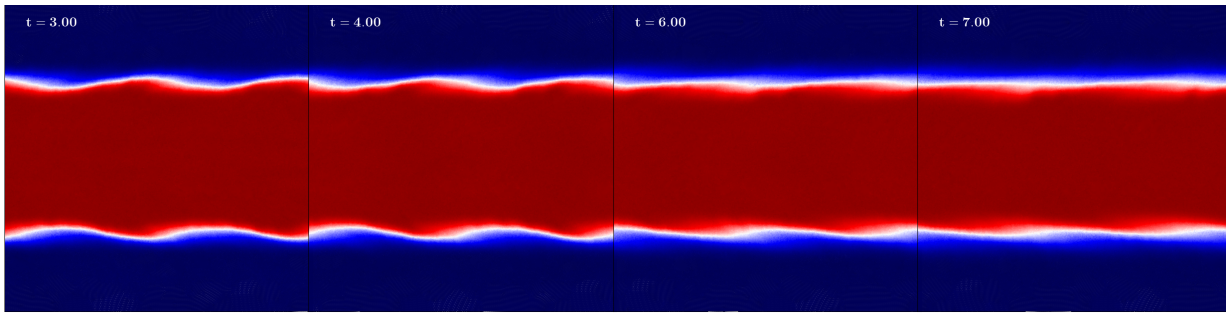

Figure 2.3: Colormap of the density for the 'standard' SPH + artificial conductivity at $t = 3$, $t = 4$, $t = 6$ and $t = 7$.

With the implementation of the artificial conductivity the problem of the surface tension is solved, but we still don't get the roll shape of the KHI, which means that artificial conductivity is not the only feature we have to add to our code.

2.3.3 Selection of the kernel

By adding the artificial conductivity we have solved the problem we had with the surface tension, but we cannot reproduce the typical roll of the KHI yet. Read et al. (2010) explained that the fact that the particles are not perfectly arranged and that there is some

[4]Since the internal energy evolves with time, the dependence of α_i^c with u_i also implies a dependence of α_i^c with time.

level of particle disorder can introduce some error, the so called 'E_0 error'. This error is the attempt of SPH to restore the particle order and it can dominate the simulation resulting in a wrong performance of our code (Price, 2012). A density step (which is the case of our simulations) is a clear case of an irregular particle distribution, suggesting that the fact that our code fails to reproduce the correct mixing in our runs can be due to this type of error. The best way to reduce the 'E_0 error' is to increase the number of neighbours with which the code computes the density. The problem is that with a cubic spline kernel, a high number of neighbours leads to 'pairing instability' (Schuessler and Schmitt, 1981), produced when two particles are very close to each other. When decreasing the distance between the particles, the pressure gradient reaches a maximum and, if the distance decreases even more, the force instead of being repulsive, it becomes attractive, causing that the two particles form a "pair" and they end up collapsing in the same position[5].

Dehnen and Aly (2012) demonstrated that those smoothing kernels whose Fourier transform is not non-negative will always produce pairing instability for a large number of neighbours and, to solve that, they proposed a Wendland function (Wendland, 1995) as a kernel. This type of function has a Gaussian-like shape with compact support and, unlike the B-spline functions, has a non-negative Fourier transform, which means that it is stable for any number of neighbours. The kernel for a Wendland function is defined by equation 1.45, but with $C_{norm} = 1365/512\pi$ and

$$f(q) = \begin{cases} (1-\frac{q}{2})^8(4q^3 + \frac{25q^2}{4} + 4q + 1), & q < 2; \\ 0. & q \geq 2, \end{cases} \tag{2.10}$$

for the case of a C^6 kernel in 3D (Price et al., 2018), the one that we are going to use in our simulations.

In this section we have used the same parameters as in the section before, but we have changed the cubic spline kernel for a Wendland C^6 kernel with 295 neighbours.

Figure 2.4 shows that with a higher number of neighbours we can finally get the KHI

[5]In appendix B we can see the effect that the pairing instability has in our simulations.

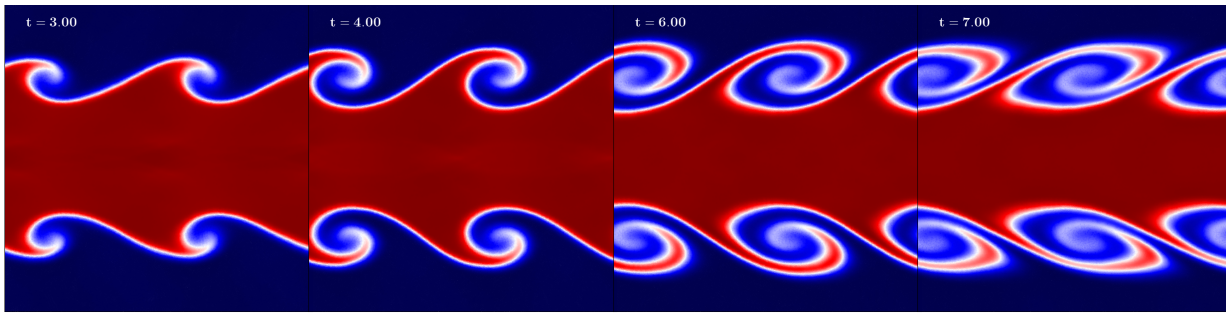

Figure 2.4: Same images as in figure 2.3 but using in this case a Wendland C^6 kernel with 295 neighbours.

but, in order to be able to use this high number of neighbours, we need a Wendland C^6 kernel[6]. We, therefore, agree with Dehnen and Aly (2012), who concluded that "the Wendland functions are ideal candidates for SPH smoothing kernels, in particular when large N_H are desired", which is the case of a shear flow.

2.3.4 Addition of an adaptive artificial viscosity

The addition of artificial dissipation terms to the code is fundamental in order to treat the discontinuities properly. We already added artificial conductivity and we could see that it played an important role in smoothing the discontinuity, so the next step is to add an adaptive artificial viscosity (Monaghan and Gingold, 1983; Monaghan, 1992). The artificial viscosity (AV) is fundamental to treat shocks properly due to it removes the post-shocks oscillations and noise. But since we are dealing with ideal fluids, we don't want artificial viscosity where we don't have shocks, that's why it is so important an efficient switch that switches on the viscosity where there are shocks and switches it off away from them. In our version of Gadget we use the method explained in Beck et al. (2015), which adds a term in the equation of movement of the form

$$\left.\frac{\mathrm{d}\vec{v_i}}{\mathrm{d}t}\right|_{\mathrm{visc}} = \frac{1}{2}\sum_{j=1}^{N}\frac{m_j}{\rho_{ij}}\left(\vec{v_j}-\vec{v_i}\right)\alpha_{ij}^{v}\, f_{ij}^{\mathrm{shear}}\, v_{ij}^{\mathrm{sig,v}}\overline{F}_{ij}\,. \tag{2.11}$$

[6]Later we are going to study how the results vary depending on the number of neighbours.

In order to conserve the total energy, we need to compensate the work done against the viscous force in the thermal reservoir by adding another term to the energy equation

$$\left.\frac{\mathrm{d}u_i}{\mathrm{d}t}\right|_{\mathrm{visc}} = -\frac{1}{2}\sum_{j=1}^{N}\frac{m_j}{\rho_{ij}}\,(\vec{v_j}-\vec{v_i})^2\,\alpha_{ij}^{v}\,f_{ij}^{\mathrm{shear}}\,v_{ij}^{\mathrm{sig,v}}\overline{F}_{ij}\,, \tag{2.12}$$

where $\rho_{ij} = (\rho_i + \rho_j)/2$ is the symmetrised density, $\alpha_{ij}^{v} = (\alpha_i^{v} + \alpha_j^{v})/2$ is the symmetrised viscosity coefficient, $f_{ij}^{\mathrm{shear}} = (f_i^{\mathrm{shear}} + f_j^{\mathrm{shear}})/2$ the symmetrised shear flow limiter and $v_{ij}^{\mathrm{sig,v}}$ the pairwise signal velocity.

The purpose of the signal velocity[7] (Monaghan, 1997) is to switch on the AV when two particles are approaching ($\vec{v_{ij}}\cdot\hat{r}_{ij} \leq 0$) and switch it off when they are not ($\vec{v_{ij}}\cdot\hat{r}_{ij} > 0$). It also determines the strength of the AV and measures the particle disorder

$$v_{ij}^{\mathrm{sig,v}} = \begin{cases} c_i^s + c_j^s - \beta\,\vec{v_{ij}}\cdot\hat{r}_{ij}\,, & \vec{v_{ij}}\cdot\hat{r}_{ij} \leq 0; \\ 0\,, & \vec{v_{ij}}\cdot\hat{r}_{ij} > 0, \end{cases} \tag{2.13}$$

where c^s is the sound speed of the particle and β is a pre-factor chosen to be $\beta = 3$.

The aim of the artificial viscosity is to act only in shocks and to be deactivated in shear flows. In order to avoid a shear viscosity that could lead to bad results in simulations of shear flows, Balsara (1995) suggested the shear flow limiter

$$f_i^{\mathrm{shear}} = \frac{|\nabla\cdot\vec{v}|_i}{|\nabla\cdot\vec{v}|_i + |\nabla\times\vec{v}|_i + \sigma_i}\,, \tag{2.14}$$

with $\sigma_i = 0.0001 c_i^s/h_i$ for numerical stability reasons. When there is a shock, the limiter is dominated by $|\nabla\cdot\vec{v}|_i$ and thus, $f_i^{\mathrm{shear}} \simeq 1$, while if there is a shearing flow, the limiter is dominated by $|\nabla\times\vec{v}|_i$ and $f_i^{\mathrm{shear}} \simeq 0$.

The calculation of the viscosity coefficient α_i^{v} is based on the switch suggested by Cullen and Dehnen (2010), which uses a shock indicator

$$R_i = \frac{1}{\rho_i}\sum_{j=1}^{N}\mathrm{sign}(\nabla\cdot\vec{v})_j m_j W_{ij}\,, \tag{2.15}$$

where $\mathrm{sign}(\nabla\cdot\vec{v})_j$ is negative and, therefore, $R_i \simeq -1$ when there is a shock. Nevertheless, R_i cannot distinguish between pre and post-shocks regions, so another parameter must

[7] Note that this signal velocity is not the same as the one explained for the artificial conductivity (2.7).

be added. To determine the direction of the shock, this factor must depend on the time derivative of the velocity divergence

$$A_i = \xi_i \max(0, -(\dot{\nabla} \cdot \vec{v})_i) \,, \tag{2.16}$$

where $(\dot{\nabla} \cdot \vec{v})_i < 0$ indicates a pre-shock region and $(\dot{\nabla} \cdot \vec{v})_i > 0$ a post-shock region. ξ_i indicates the ratio of strength of the shock and is given by

$$\xi_i = \frac{|2(1-R_i)^4(\nabla \cdot \vec{v})_i|^2}{|2(1-R_i)^4(\nabla \cdot \vec{v})_i|^2 + |\nabla \times \vec{v}|_i^2} \,. \tag{2.17}$$

Now we can define the target value $\alpha_i^{\mathrm{loc},v}$ of AV as

$$\alpha_i^{\mathrm{loc},v} = \alpha_{\max} \frac{h_i^2 A_i}{h_i^2 A_i + (v_i^{\mathrm{sig}})^2} \,. \tag{2.18}$$

When $\alpha_i^{\mathrm{loc},v}$ is smaller than the current α_i^v, we set the coefficient to $\alpha_i^{\mathrm{loc},v}$. When it is bigger, we let α_i^v decay and we calculate it by integrating

$$\dot{\alpha_i^v} = \left(\alpha_i^{\mathrm{loc},v} - \alpha_i^v\right) \frac{v_i^{\mathrm{sig}}}{l h_i} \,, \tag{2.19}$$

where we set $l = 4.0$ (Beck et al., 2015), which specifies the decay length of the AV.

If the artificial viscosity works properly, we shouldn't see a significant difference in our results if we compare them with the ones obtained in the previous section. After adding the artificial viscosity to our code, we get the results shown in figure 2.5.

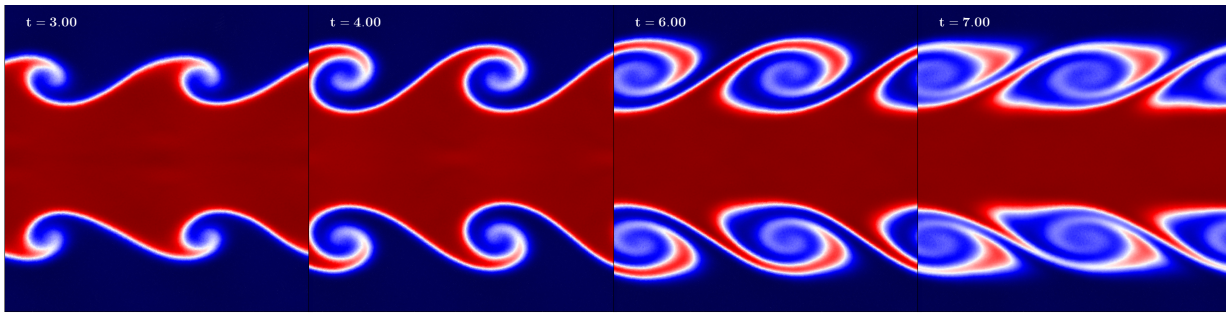

Figure 2.5: Same images as in figure 2.4 but after adding artificial viscosity to the code.

We cannot find a big difference between figures 2.4 and 2.5, meaning that the switch works fine. In order to see this effect in a more quantitative way, figure 2.6 shows the

variation of the artificial viscosity with time for three different initial values of the artificial viscosity.

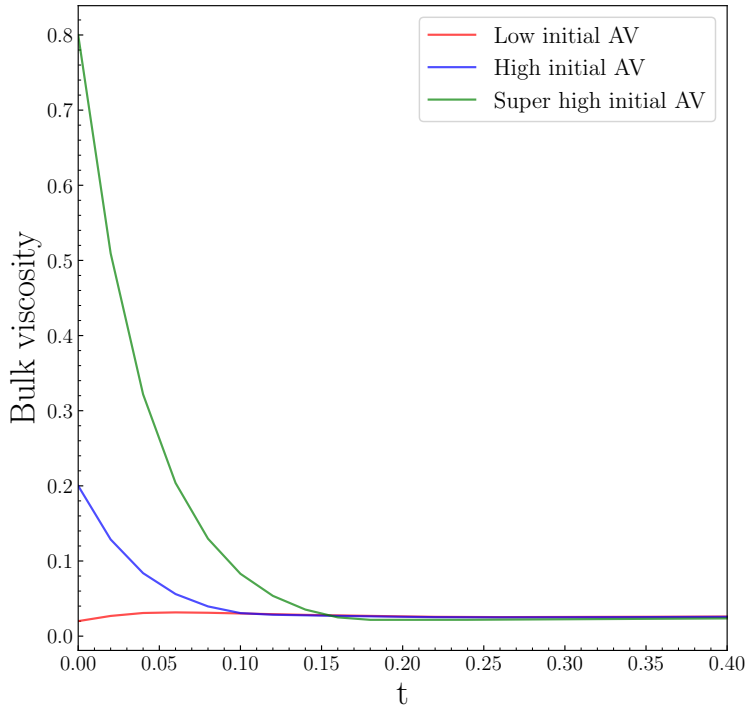

Figure 2.6: Change of the average artificial viscosity within the whole domain with time for three different initial values: *Low initial AV* $= 0.02$; *High initial AV* $= 0.2$ and *Super high initial AV* $= 0.8$.

The code successfully reduces the artificial viscosity to almost zero in a few timesteps even if we start with a very high initial value of the artificial viscosity. This also explains that we don't see big differences between the run without artificial viscosity and the one with it.

2.3.5 Addition of wake-up

In order to reduce the computational costs of the simulation, the particles are divided in active particles and inactive particles. The former are faster, which require a shorter timestep and hence, the quantities are calculated in the current timestep; while the quantities

of the latter can be calculated with longer timesteps. The problem arises when an active particle enters a region of inactive particles and these don't notice its presence leading to non-physical results. That's why we implement the so called wake-up (Saitoh and Makino, 2009; Pakmor et al., 2012). In each timestep, the timestep of every active particle is re-computed using

$$\Delta t_i = \frac{C\, h_i}{v_i^{\mathrm{sig}}}\,, \tag{2.20}$$

where C is the Courant factor and v_i^{sig} is the maximum signal velocity. During the force computation the code compares the signal velocities between the active particle and the other particles within the kernel by evaluating

$$v_{ij}^{\mathrm{sig}} < f_\omega\, v_j^{\mathrm{sig}}\,, \tag{2.21}$$

with f_ω being the tolerance factor (in our case $f_\omega = 3$). This can notice sudden changes in the pairwise signal velocity and wake-up the inactive particle.

This tool is really useful in simulations with a big speed difference between the particles (i.e. shocks), which is not our case. But in cosmological simulations we can have both shear discontinuities and shocks, so it is important to test this implementation with our setup to see whether it affects or it doesn't change the results.

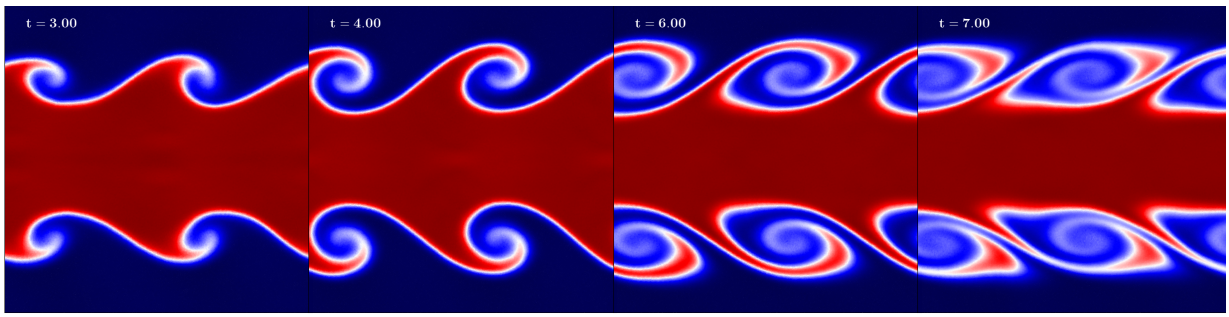

Figure 2.7: Same images as in figure 2.5 but including wake-up to the code.

Figure 2.7 shows that the results don't vary much from the ones without wake-up (figures 2.4 and 2.5).

After having tested the implementation described in Beck et al. (2015) of Gadget3 code compared to Gadget2 and, after having seen the improvement obtained with these

modifications, we are going to keep this additional features for all the simulations from now on, unless we specify something different.

2.3.6 Change on the number of neighbours

In section 2.3.3 we switched the kernel and started using a Wendland C^6 kernel with 295 neighbours for the computation of the physical quantities. We mentioned that using a Wendland kernel allows us to use a higher number of neighbours and, therefore, obtain better results. But what does 'higher number of neighbours' mean? We have used 295, but could we have used 150? And 350? In this section we are going to run different simulations varying the number of neighbours to see how this affects the results.

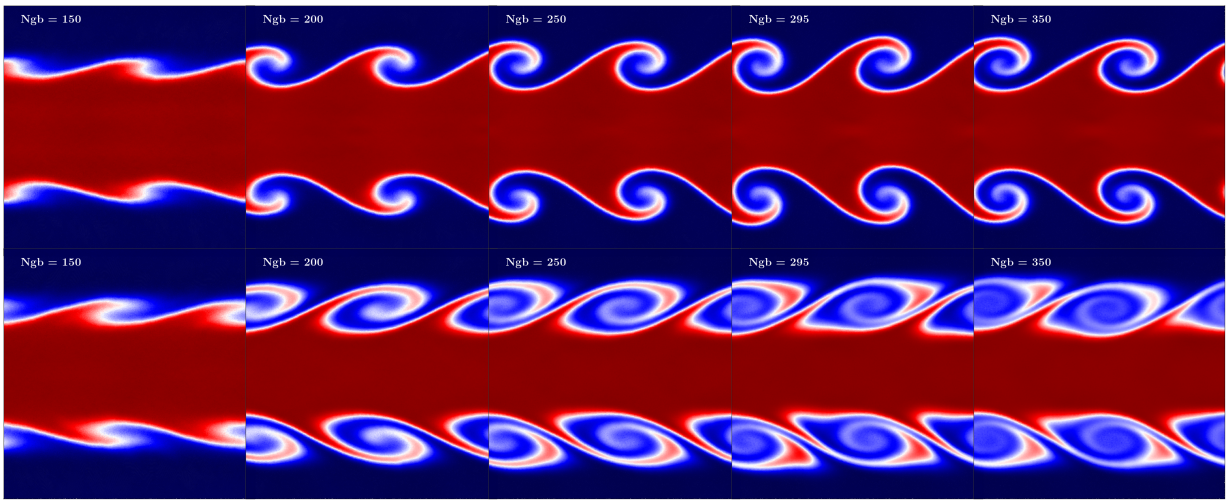

Figure 2.8: Comparison of the results at $t = 4$ (top row) and $t = 7$ (bottom row) for different numbers of neighbours.

Analysing figure 2.8 we find a correlation between the number of neighbours employed to calculate the hydrodynamic quantities and the growth of the instability. For 150 neighbours we don't even get the roll; for 200 neighbours we get a non-fully developed roll; while with 250 neighbours and above we can already get a reasonable shape of the roll. In order to study this discrepancy in more detail, in the next chapter (3) we are going to perform some quantitative tests and try to extract some conclusions.

2.4 Meshless Finite Mass

Throughout this project we have also tested this setup using a MFM scheme, also included in Gadget. As we mentioned in section 1.3, MFM doesn't need artificial diffusion and, therefore, we haven't performed all the tests we did with SPH, but we have tested directly the difference in the kernels and the number of neighbours. For these tests we are going to use a new and improved version implemented in Gadget (we will refer to it as OpenGadget-MFM) and compare it with the version described in Hu et al. (2014) and Steinwandel et al. (2020) (from now on, P-Gadget3-MFM).

2.4.1 Cubic spline kernel

In the case of MFM, since we don't have the 'E_0 error' problem (see section 2.3.3), in principle we should be able to get satisfactory results with a low number of neighbours and, therefore, there is no need of using a Wendland kernel. That is one of the reasons why the cubic spline kernel is widely used with MFM in cosmological simulations (Davé et al., 2016; Hopkins et al., 2018). In this section we first compare the results obtained with a cubic spline kernel and 32 neighbours with both codes, OpenGadget-MFM and P-Gadget3-MFM. In figure 2.9 we can see both results.

Both codes are able to develop the roll, something that didn't happen with the SPH code and a cubic spline kernel. Also, both simulations are able to mix the two fluids successfully, as we would expect in this type of systems. Despite these improvements with respect to the SPH code with a cubic spline kernel, we note that the simulation is not symmetric due to some secondary instabilities triggered by numerical noise that grow provoking the breakdown of the billow.

Comparing the two MFM codes, we find less secondary instabilities in OpenGadget-MFM than in P-Gadget3-MFM, which indicates that there is less numerical noise, but still there is no symmetric solution.

Since using 32 neighbours we don't get successful results, the next step is to increase the number of neighbours to 50 and see if there is an improvement. The results of both

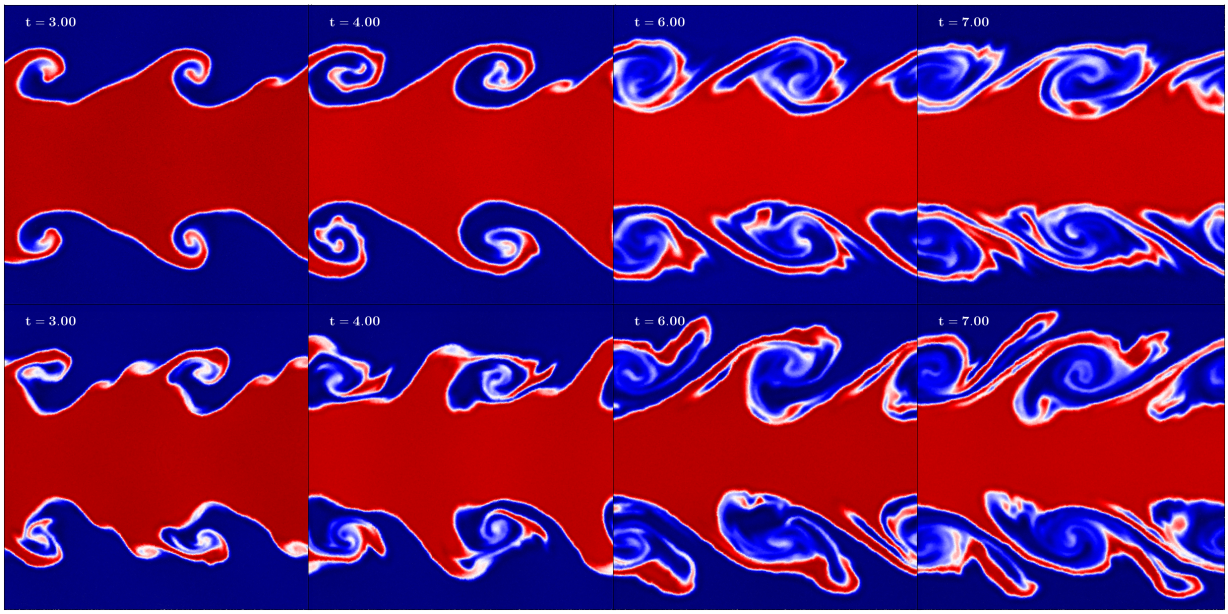

Figure 2.9: Simulations with a cubic spline kernel and 32 neighbours with OpenGadget-MFM (top row) and P-Gadget3-MFM (bottom row).

codes with 50 neighbours can be seen in figure 2.10.

Using 50 neighbours we can appreciate an improvement with respect to the results with 32, but some secondary instabilities are still produced, mainly with the P-Gadget3-MFM code, but also with OpenGadget-MFM. This small instabilities triggered numerically lead to a non-fully symmetric result.

Overall, we could say that, unlike the SPH code, the MFM codes with a cubic spline kernel can develop the instability we are looking for. But although the roll is able to grow, the numerical noise plays an important role in the growth of secondary instabilities pushing away the simulations from the theoretical result.

2.4.2 Wendland C^6 kernel

An increase in the number of neighbours reduces the numerical error and improves the results, so in this section we are going to switch the kernel to a Wendland C^6 kernel, which allows a higher number of neighbours. As we did with SPH, we are going to use

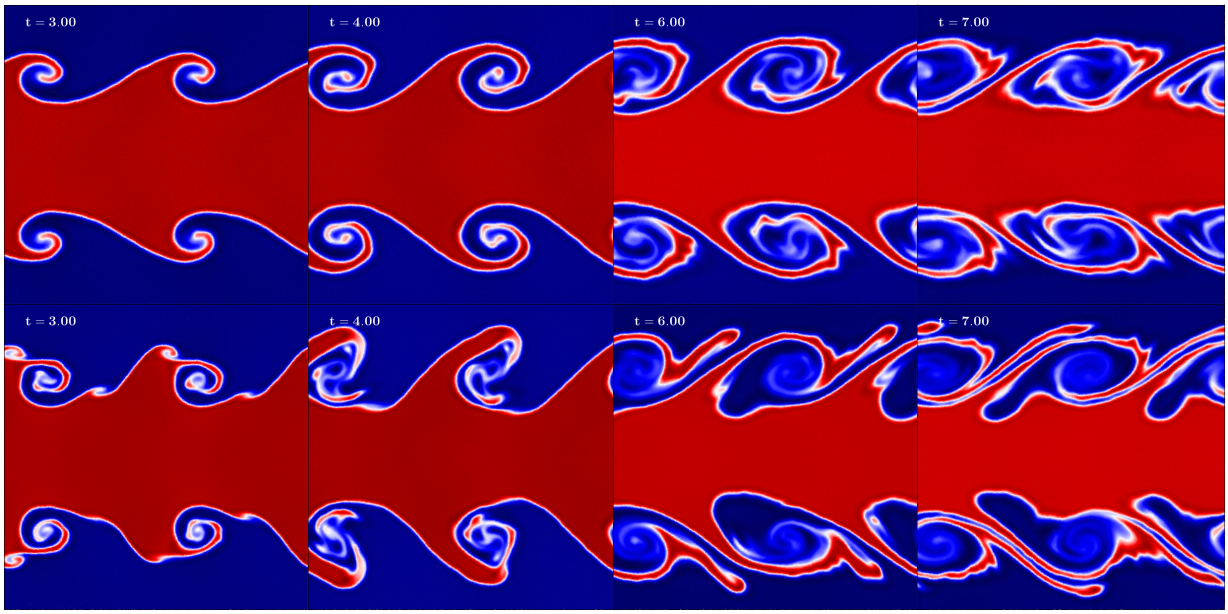

Figure 2.10: Results using a cubic spline kernel and 50 neighbours with OpenGadget-MFM (top row) and P-Gadget3-MFM (bottom row).

295 neighbours and analyse if it improves the previous results. As before, we have run the simulation using OpenGadget-MFM and P-Gadget3-MFM (figure 2.11) to compare the results with both codes.

There is a noticeable improvement with a smoother shape of the roll in both cases if we compare them with the cubic spline run. Specially in the OpenGadget-MFM simulation we can clearly appreciate the spiral of the billows, indicating a low diffusion, and a high symmetry, showing that the numerical noise is well suppressed and can hardly develop into secondary instabilities. If we compare both results we can also see that the roll of P-Gadget3-MFM grows slightly faster than the one in OpenGadget-MFM, which will be analysed later in chapter 3. The results with 150 neighbours can be seen in the appendix D.

Note that an increase in the number of neighbours also implies a larger number of Riemann problems the program has to solve and, therefore, an increase of the computational time. However, with the improvement seen in the results, the higher computational cost

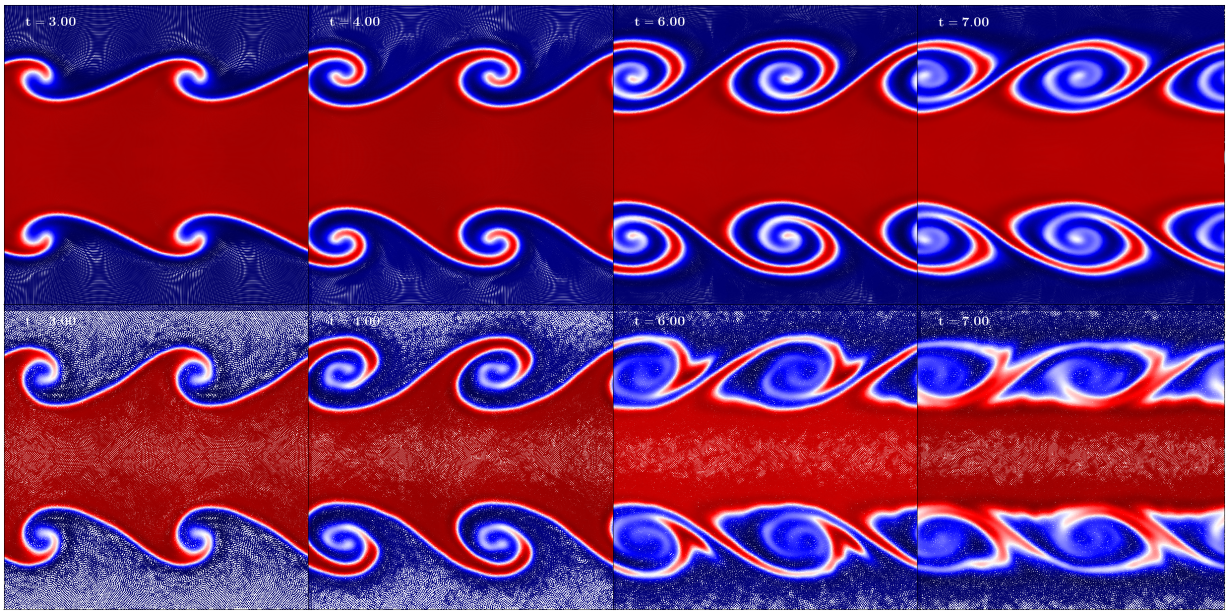

Figure 2.11: Colormaps of the density using a Wendland C^6 kernel and 295 neighbours with OpenGadget-MFM (top row) and P-Gadget3-MFM (bottom row).

can be worth, depending on the aim of the work.

2.4.3 Selection of the slope limiter

As in all Riemann-problem based methods, the slope limiter is fundamental in MFM simulations in order to get realistic results. In every time-step, an average flux is computed at the center of the cell, so in order to get a more accurate result on the interfaces, a piece-wise linear reconstruction is needed. Then the discontinuity at the interface is solved by a Riemann solver, but in order to do that, a gradient must be estimated (see the estimation of the gradient in section 1.3). In the cases where the difference of a quantity is too big, this estimated gradient can give unphysical results (e.g. a negative density). In order to avoid that, we limit the gradient with a slope limiter such that the linear extrapolations of the cell states to the cell interfaces do not introduce new extrema (Barth and Jespersen, 1989). Once we have a slope limited gradient, we can now estimate the left and right states adjacent to an interface by extrapolating from the center of the cell to the

edges (Springel, 2014).

Since the choice of the slope limiter can determine the results we get, we want to test different slope limiters. In the previous section we used OpenGadget-MFM with the Springel slope limiter, introduced by Springel (2010), and now we are going to compare those results to the ones obtained using also OpenGadget-MFM, but in this case with the GIZMO slope limiter, introduced by Hopkins (2015).

The Springel slope limiter is defined as

$$\langle \nabla \phi \rangle_i' = \alpha_i \ \langle \nabla \phi \rangle_i \ , \tag{2.22}$$

where ϕ is the computed quantity and the slope limiter $0 \leq \alpha_i \leq 1$ is calculated as

$$\alpha_i = \min(1, \psi_{ij}) \, . \tag{2.23}$$

We take the minimum with respect to all the neighbours of i. ψ_{ij} is defined as

$$\psi_{ij} = \begin{cases} (\phi_i^{\max} - \phi_i)/\Delta\phi_{ij} \, , & \Delta\phi_{ij} > 0; \\ (\phi_i^{\min} - \phi_i)/\Delta\phi_{ij} \, , & \Delta\phi_{ij} < 0; \\ 1 \, , & \Delta\phi_{ij} = 0, \end{cases} \tag{2.24}$$

where $\Delta\phi_{ij} = \langle \nabla \phi \rangle_i \cdot (f_{ij} - s_i)$ is the estimated difference between the value at the centroid f_{ij} of the face and the center of the cell i, and $\phi_i^{\max} = \max(\phi_j)$ and $\phi_i^{\min} = \min(\phi_j)$ are the maximum and minimum value of ϕ among the neighbours of i (including i itself).

In the case of GIZMO, the slope limiter α_i in equation 2.22 is given by

$$\alpha_i = \min \left[1, \beta_i \min \left(\frac{\phi_i^{\max} - \phi_i}{\phi_{i,\mathrm{mid}}^{\max} - \phi_i}, \frac{\phi_i - \phi_i^{\min}}{\phi_i - \phi_{i,\mathrm{mid}}^{\min}} \right) \right] \, , \tag{2.25}$$

where $\phi_{i,\mathrm{mid}}^{\max}$ and $\phi_{i,\mathrm{mid}}^{\min}$ are the maximum and minimum values of the reconstructed ϕ at the i-side of the interface between particles i and j. If there is a good particle order and the gradients are trust-worthy, we should use a more "aggressive" (larger value of β) limiter, while if this is not the case, a more "stable" (smaller value of β) one is more appropriate. A more stable slope limiter will lead to more diffusive results compared to a more aggressive

one. After some experimentation, Hopkins (2015) found the following expression for β

$$\beta_i = \max\left[\beta_{\min}, \beta_{\max}\min\left(1, \frac{N_{\text{cond}}^{\text{crit}}}{N_{\text{cond}}^{i}}\right)\right], \tag{2.26}$$

with $\beta_{\min} = 1$, $\beta_{\max} = 2$ and N_{cond} being the condition number (see Hopkins (2015)).

We ran the GIZMO slope limiter simulation both with a cubic spline kernel and 32 neighbours and a Wendland C^6 kernel and 295 neighbours in order to be able to compare the results with the ones obtained with the Springel slope limiter (shown in figures 2.9 and 2.11 respectively) and we show the results in figure 2.12.

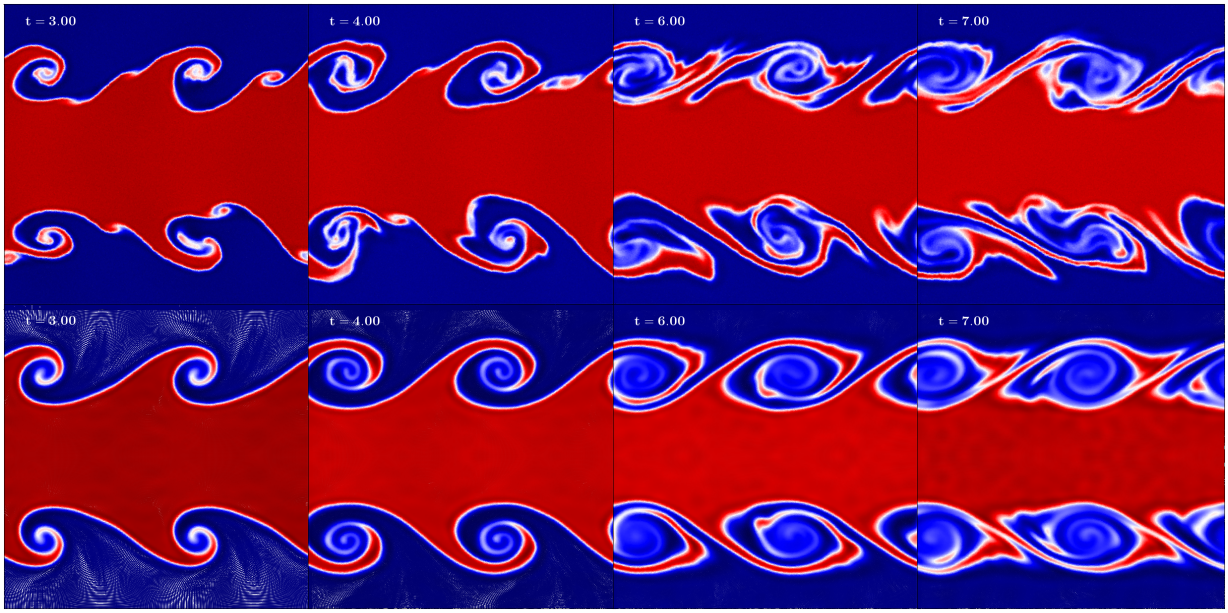

Figure 2.12: Results using OpenGadget-MFM with the GIZMO slope limiter with a cubic spline 32 neighbours (top row) and a Wendland C^6 295 neighbours (bottom row).

In the case of cubic spline kernel with 32 neighbours, the result is similar compared to the one obtained with the Springel slope limiter (top row of figure 2.9). In both cases the secondary instabilities triggered by numerical perturbation evolve to a non-symmetric system. With the GIZMO limiter we get a slightly larger amount of secondary instabilities, but in any case, both results are non-symmetric at late times.

With the Wendland C^6 kernel and 295 neighbours there are some differences in the results if we compare them with the ones obtained with the Springel slope limiter (top

row of figure 2.11). The roll with the GIZMO limiter evolves faster and, hence, the billow pairing occurs at earlier times. Also, it appears to be more diffusive than the Springel slope limiter.

Chapter 3

SPH vs MFM

Once we have analysed how the different parameters we choose affect the results of our simulations just by judging the shape of the roll, in this chapter we are going to compare the results obtained with the SPH code and the ones obtained with MFM in a more quantitative way. For this purpose, we are going to compare the measurements of different characteristics of the rolls in each simulation.

3.1 Height of the billows

The first feature we are going to study is the height the roll reaches in each code. Ideally, the initial perturbation initiates the instability and the amplitude of this instability starts to increase approximately linearly until it saturates and, after it, decreases as each billow pairs with the adjacent billows (Rahmani et al., 2014). The height of the roll indicates how good the code is to develop the KHI and, conversely, which code suppresses more the instability and prevent its growth.

The way we computed the height of the billows is similar to the one employed in Roediger et al. (2013b). We focus only on the upper half of the domain (since the domain is symmetric, we choose only the upper half and, therefore, the total amount of particles is reduced by half). Then we mark every particle depending if they are "red" (denser fluid)

or "blue" (lighter fluid) at $t = 0$ so we can trace them later. Once we have marked every particle, we divide the half domain in 100 parts (bins) along the y direction and calculate the amount of "red" and "blue" particles in every bin for every snapshot. The top of the billow will be the lowest bin where at least 95% of the particles are "blue", while the bottom will be the highest bin with at least 95% of "red" particles. Finally, we compute the amplitude of the roll by calculating the distance between the top and the bottom of the billow for every different snapshot. Figure 3.1 shows this procedure, where on the left we have the individual particles identified as "blue" or "red" and the two yellow lines indicating the top and the bottom of the rolls, while on the right we have the same snap, but in this case we have plotted the density.

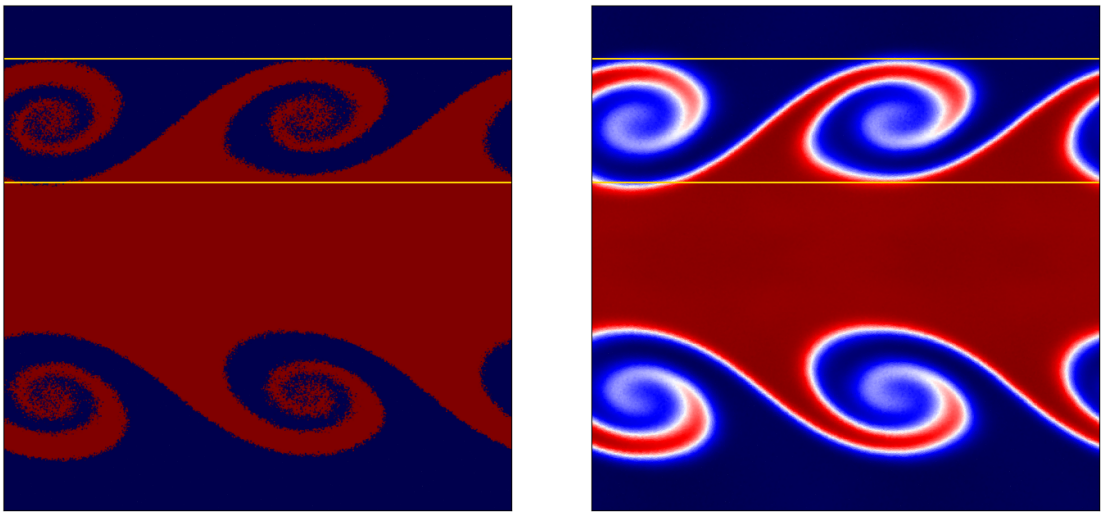

Figure 3.1: Example at time $t = 5$ of the way we computed the amplitude of the billows where we show the particles with their initial "color" (left) and the colormap of the density (right) indicating the top and bottom of the rolls with two yellow lines.

We followed this procedure for our SPH code (OpenGadget-SPH), as well as with the two MFM versions (OpenGadget-MFM and P-Gadget3-MFM). In all the cases we are using a Wendland C^6 kernel with different number of neighbours to see how the results change depending on the number of neighbours used in the computation. We decided not to use the data taken from the simulations with the cubic spline kernel due to the fact that, as we

saw in the previous chapter, the Wendland C^6 kernel produces much better results than the cubic spline. In figure 3.2 we can compare the heights of the billows for the different codes and the different number of neighbours.

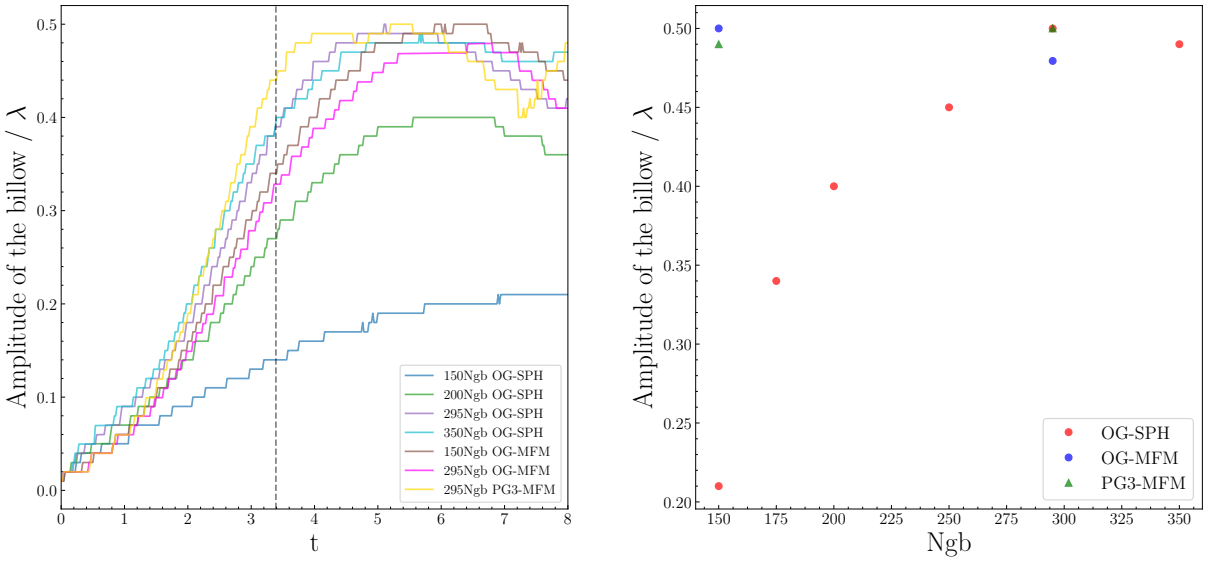

Figure 3.2: Temporal evolution of the height of the rolls (left) and maximum height reached depending on the number of neighbours (right). Note that in the left panel we are only considering a sub-set of the simulations in the right panel.

The left plot of figure 3.2 shows a general linear growth until approximately $t = \tau_{KH}$[1] (the vertical dashed line), which is what we would expect for the ideal case. However, despite we are trying to reproduce an ideal case, we have some intrinsic viscosity (see section 2.3.4). This explains why it takes longer than $t = \tau_{KH}$ for the instability to reach the maximum amplitude. Each code has a different amount of numerical viscosity[2], which also explains why, for instance, the rolls simulated using P-Gadget3-MFM evolve faster than the ones with OpenGadget-MFM, in agreement with the results of the previous chapter.

Looking at the plot on the right, that shows the maximum height of the roll reached

[1]Remember that the KH time-scale of the ideal case with our initial conditions is $\tau_{KH} = 3.39$.

[2]This will be analysed in more detail in section 3.6.

by each simulation, we can see a poor performance of the SPH codes with a low number of neighbours, as we already saw in the previous chapter. The SPH codes reach the largest amplitude with 295 and 350 neighbours, while all the MFM codes perform similarly, approaching an amplitude of $\sim \lambda/2$, in agreement with the height suggested in Roediger et al. (2013b). Despite the rolls simulated with P-Gadget3-MFM grow faster than the ones using OpenGadget-MFM, they both reach a similar amplitude, meaning that the code slows down the growth of the instability, but it doesn't suppress it.

3.2 Growth of the velocities

Another way to quantify the growth of the KHI is measuring the rate of change of the y-velocity component of the perturbation. The y-velocity grows exponentially with time until the KH time-scale (see section 2.1). In order to study the growth of v_y, we use the discrete convolution suggested by McNally et al. (2012) and also used in their analysis by Obergaulinger and Aloy (2020) to compute the amplitude M of the initially excited mode. We have adapted the formula to our initial conditions, leading to

$$s_i = v_y h_i^3 \sin\left(\frac{2\pi\left(x+\frac{\lambda}{2}\right)}{\lambda}\right) \exp\left(-\frac{2\pi}{\lambda}\left|64-y\right|\right) \tag{3.1}$$

$$c_i = v_y h_i^3 \cos\left(\frac{2\pi\left(x+\frac{\lambda}{2}\right)}{\lambda}\right) \exp\left(-\frac{2\pi}{\lambda}\left|64-y\right|\right) \tag{3.2}$$

$$d_i = h_i^3 \exp\left(-\frac{2\pi}{\lambda}\left|64-y\right|\right) \tag{3.3}$$

$$M = 2\sqrt{\left(\frac{\sum_{i=1}^{N} s_i}{\sum_{i=1}^{N} d_i}\right)^2 + \left(\frac{\sum_{i=1}^{N} c_i}{\sum_{i=1}^{N} d_i}\right)^2}, \tag{3.4}$$

where h_i is the smoothing length, λ the wavelength of the perturbation (in our case $\lambda = 128$) and N is the total number of particles in the domain. For the computation of M we used only one quarter of the full domain in order to take only one perturbation for the calculation. The results of this computation are shown in figure 3.3.

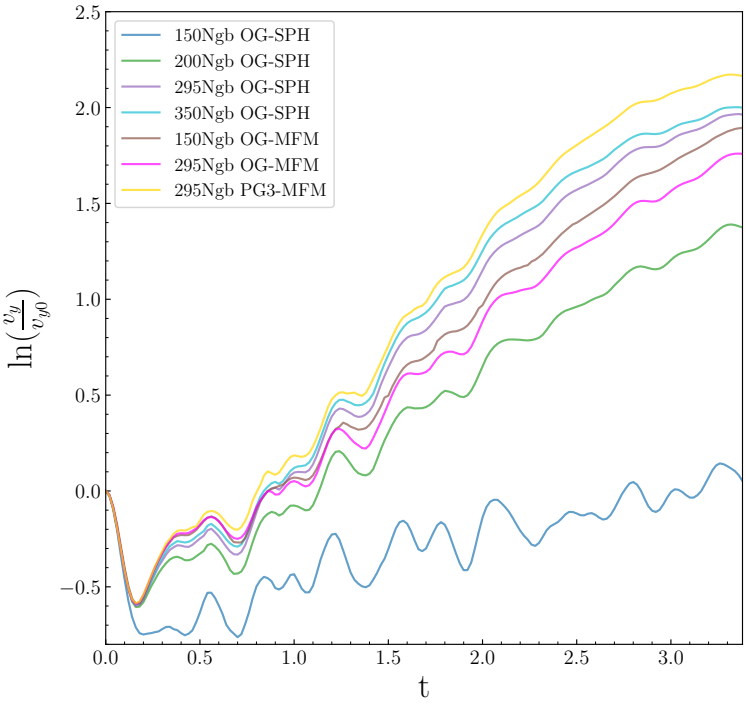

Figure 3.3: Change of amplitude of the y-velocity with time for the different codes until $t = \tau_{KH}$.

We plotted the natural logarithm of the normalized y-velocity in order to see the linear growth with time more clearly until $t = \tau_{KH}$. We can observe that at $t \leq 0.16$ ($t \leq 0.20$ for the case of OpenGadget-SPH and 150 neighbours) the amplitude of v_y decreases. This is due to the loss of kinetic energy by moving along the y axis through a fluid flowing in the opposite direction (Junk et al., 2010). In the case of a fluid with a higher viscosity this loss of kinetic energy is more significant, which is the case of OpenGadget-SPH with 150 neighbours.

With this analysis, we can see the slower growth performed by OpenGadget-MFM that we mentioned in the previous section and the faster growth of P-Gadget3-MFM. Focusing on the results with SPH, we can see a correlation between the number of neighbours and the growth rate of the y-velocity, with the simulation with 350 growing the fastest. This difference in the growth rate can be also explained by the numerical viscosity of each code, which will be studied in detail in section 3.6.

3.3 Numerical noise

When we run simulations, we would like to keep the numerical error as little as possible. The perturbations due to this numerical noise can grow exponentially, leading to a bad performance of the code and, therefore, undesired results. We have already seen in section 2.4 how the numerical error can produce secondary instabilities that lead to a loss of symmetry in our system.

In this section we want to measure the amount of numerical error produced by each code and whether it is suppressed or it can evolve into spurious instabilities. To do so, we use the same methods employed in the previous two sections to measure how fast the amplitude of the billows grow and how the y-velocity changes with time because of the noise. While with the latter method we measure only one mode of the perturbation (corresponding to $\lambda = 128$), this can still provide useful information.

In order to see if the numerical noise can trigger the instability, we are going to change the initial conditions and, instead of adding an initial perturbation (see section 2.2), we are not going to add any perturbation. By doing this, we know that any movement in the y axis is seeded numerically and not by any initial perturbation (Rosswog, 2015). In figure 3.4 we can compare the amplitude and the growth of the y-velocity of the instabilities triggered numerically for the different codes. In this case we also included the results with the OpenGadget-MFM with a cubic spline kernel and 32 neighbours because we wanted to see the amount of numerical error produced by this code.

The plot on the left shows an initial quick growth of the amplitude in the SPH codes due to the artificial conductivity implemented in the code (this will be analysed in section 3.5 in more detail), reason why this doesn't happen with the MFM codes. Apart from that, the results using SPH and a low number of neighbours suppress pretty well the noise, possibly due to their bigger amount of intrinsic viscosity (see section 3.6), while for a high number of neighbours the instability can be triggered more easily. In the case of MFM, the simulations using a lower number of neighbours let the instabilities grow more easily than the simulations with 295 neighbours, which suppresses the numerical noise quite well

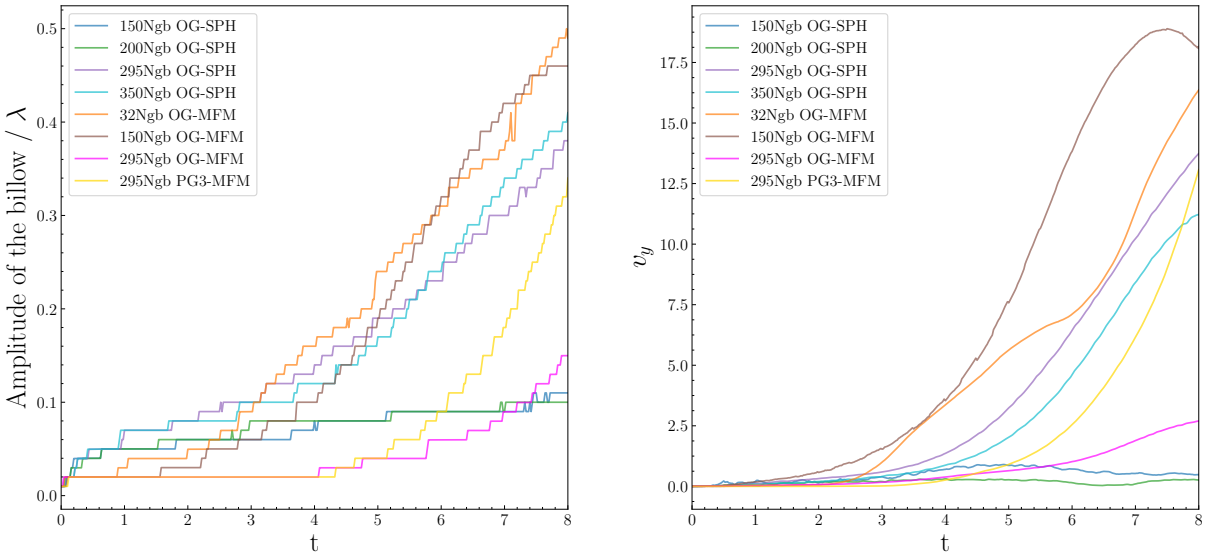

Figure 3.4: Amplitude of the instabilities seeded numerically (left) and growth rate of y-velocity if the instabilities triggered numerically (right).

and, hence, the amplitude of the instabilities remain small. This behaviour could already be seen in section 2.4, where the runs with a lower number of neighbours couldn't suppress the secondary instabilities.

The growth of ${v_y}^3$ (right) for the case of SPH shows that the velocity in codes with a lower number of neighbours barely increases, while with a higher number of neighbours the growth rate is much faster. The results of the MFM codes indicate that in the codes with a low number of neighbours the velocity increases faster, specially in the case of OpenGadget-MFM with 150 neighbours. It is worth noting the great performance of OpenGadget-MFM with 295 neighbours in suppressing the numerical noise.

The run with OpenGadget-MFM and 150 neighbours deserves some attention, not only because of its bad performance in comparison with the other codes, but also because if we plot the density colormap to see how the instability is triggered, we find some interesting results. Usually, the numerical error generates perturbations of small wavelength that end up generating small rolls that grow faster than the big ones (see equation 2.5) and combine

[3]Note that in this case we haven't plotted the v_y in logarithmic scale.

generating bigger rolls. However, in the case of OpenGadget-MFM with 150 neighbours this doesn't happen (see figure 3.5).

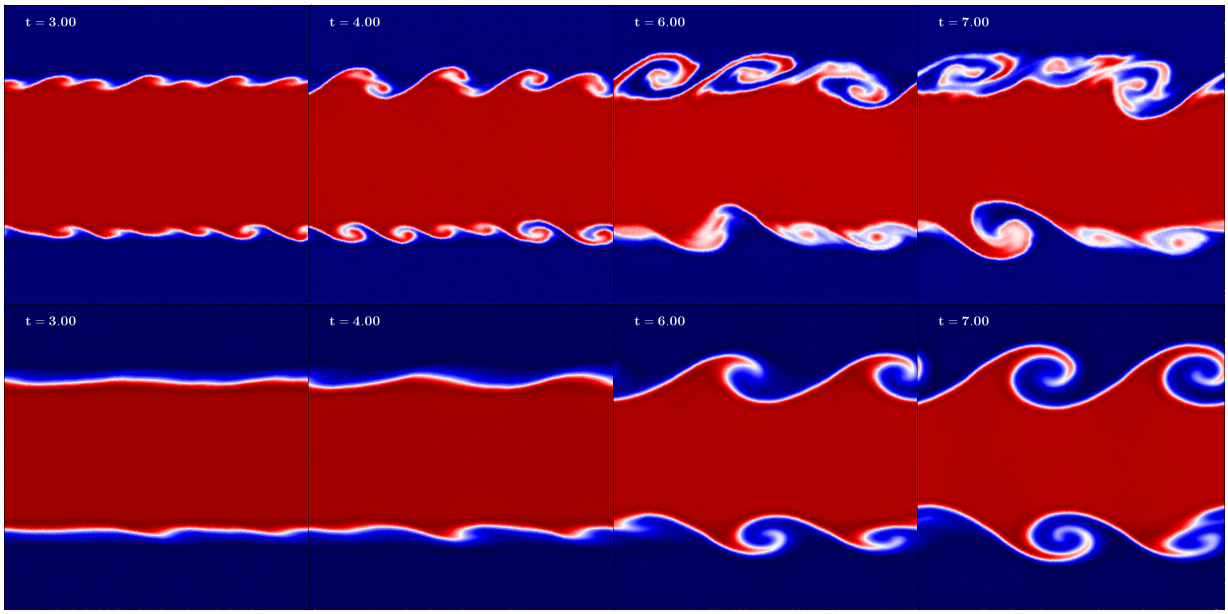

Figure 3.5: Comparison of the growth of the perturbation from numerical noise for the case of OpenGadget-MFM with a cubic spline and 32 neighbours (top row) and OpenGadget-MFM with a Wendland C^6 kernel and 150 neighbours (bottom row).

The code successfully suppresses the perturbations of small wavelength, but a perturbation of wavelength of $\lambda = 128$ is triggered numerically and grows to generate a roll similar to the ones we generated with the initial conditions explained in 2.2. If we compare these results with, for instance, the ones obtained with OpenGadget-MFM with a cubic spline kernel and 32 neighbours, we can clearly see the difference. As we have already explained, in the case with 32 neighbours the instabilities produced by perturbations of small wavelength grows faster.

3.4 Conservation of energy

We can also analyse the performance of the different codes by studying how well they conserve the energy during the simulation. Since we are reproducing a close system with periodic boundary conditions, the total energy should be conserved in time. As the simulation runs, the interactions between particles produce a loss of kinetic energy, turning it into internal energy, but always conserving the total energy. To test this, we first computed the mean kinetic energy per unit mass of the whole domain by summing up the kinetic energy contributed by each particles and dividing by the total number of particles, as well as the mean internal energy per unit mass of the system. In figure 3.6 we can see the evolution of the kinetic and internal energy of the system (normalized to their initial value) for each code.

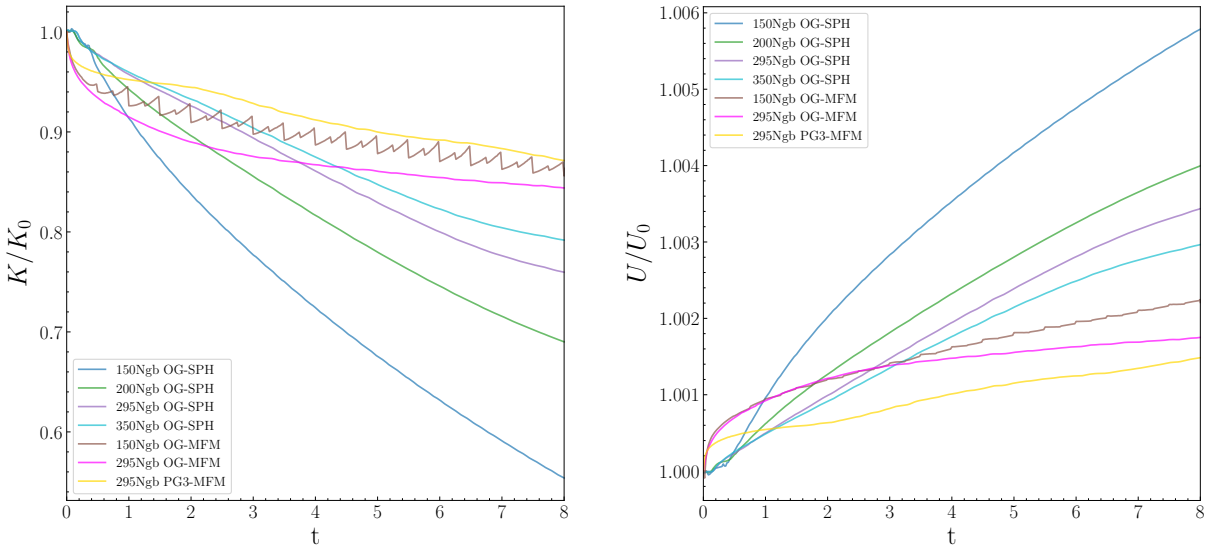

Figure 3.6: Temporal evolution of the mean kinetic energy (left) and mean internal energy (right) of the whole system.

As expected, the mean kinetic energy decreases with time, while the mean internal energy increases, meaning that the kinetic energy of the particles is transformed into internal energy. Again, the intrinsic viscosity plays an important role in turning the kinetic energy into internal energy due to the friction between the particles. This can be seen by

realising how the codes follow a trend similar to the ones we saw in sections 3.1 and 3.2, where the simulation using OpenGadget-SPH with 150 neighbours had a slower growth and P-Gadget3-MFM a faster growth of the roll.

However, this doesn't give us enough information about the conservation of energy performed by each code. To see how the codes conserve the energy, we sum up specific kinetic and internal energy and plot the result in figure 3.7.

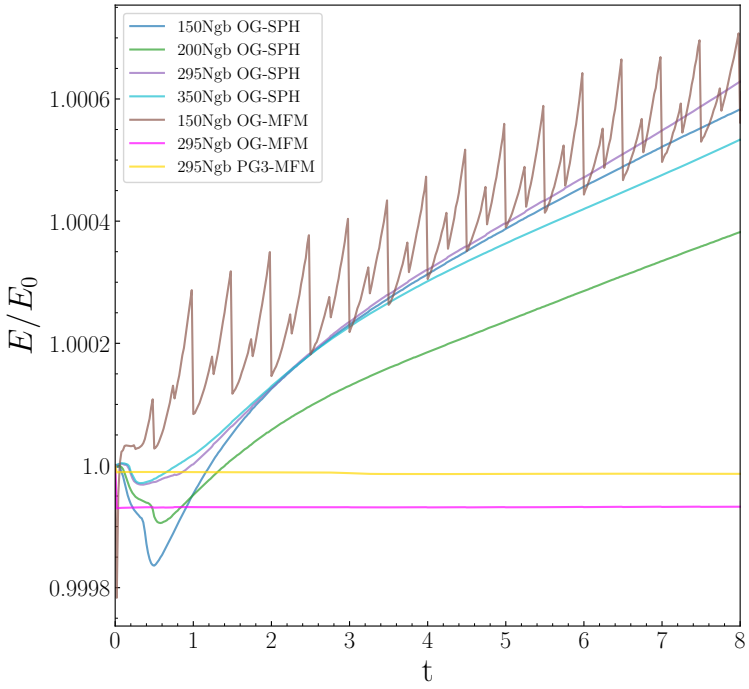

Figure 3.7: Conservation of energy per unit mass with time.

In the worst-case scenario, the variation of the energy is only of a 0.06%, which is negligible in our simulations. If we analyse in detail, the worst performance is by the SPH codes, which tend to increase the total energy of the system as well as OpenGadget-MFM and 150 neighbours. The two simulations with OpenGadget-MFM have a jump at the first snapshot, probably due to a mismatch between the set-up of the code and the initial conditions, but despite this jump, the run with 295 neighbours maintains a constant energy during all the simulation. In the case of P-Gadget3-MFM, the energy is well conserved at every time.

We also computed the mean entropy[4] per unit volume in order to see how it behaves depending on the code employed. Since we have two fluids mixing, we would expect the entropy to increase and, actually, that is one of the main reasons why SPH codes need an artificial diffusion (see section 2.3.2). The variation of entropy is somehow a way to study the degree of mixing performed by each code (we will see this in more detail in the next section). In figure 3.8 we can see the variation of the total entropy per unit volume for the different codes.

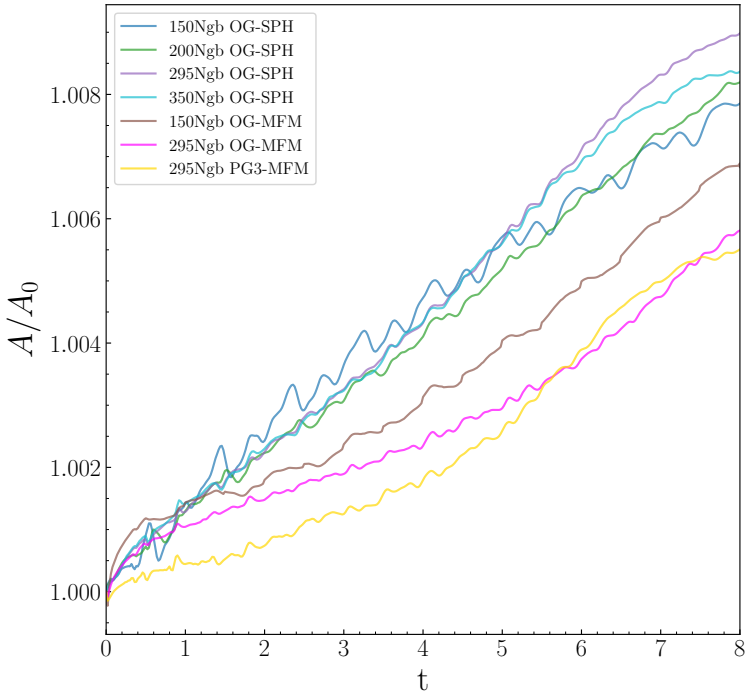

Figure 3.8: Variation of entropy per unit volume with time.

We can observe a very similar behaviour among the SPH codes, probably due to the addition of artificial diffusion, which increases the internal energy (and therefore the entropy) in a similar way in all the SPH codes. The increase of entropy within simulations with the MFM codes is lower than with SPH codes, but still noticeable.

[4]When we talk about entropy, we are actually referring to the entropic function A (1.60).

3.5 Mixing of the fluids

An important result we want to know from these type of simulations, is the degree of mixing of the two fluids due to the KHI. Although we already got an idea from figure 3.8, which indicates us that the SPH codes tend to mix the fluids more than MFM and, hence, the increase of entropy is higher, in this section we want to study this effect in more detail. We can divide the mixing of the fluids in two main processes, the diffusion and the mixing due to the KHI.

3.5.1 Diffusion

The process of diffusion occurs due to the molecular movement of the particles and tends to go from higher to lower concentration regions. In the case of MFM this process happens automatically being the code intrinsically diffusive, while in the case of SPH we have to add it artificially with the artificial conductivity explained in section 2.3.2. As we saw in that section, artificial conductivity plays an important role in resolving discontinuities and it is fundamental in reproducing the KHI.

Diffusion is the main mixing process at early times due to the fact that the roll hasn't formed yet. What diffusion does is to widen the fluid interface due to the random movement of the particles and, therefore, in order to measure the amount of diffusion of each code, we are going to analyse the thickness of the interface at early times. We divide the whole initial density domain in 20 bins and we choose the bins with the highest and lowest density. These are going to be our thresholds to consider if a particle belongs to the high density part or to the low one (see left plot of figure 3.9). Now, for each snapshot, we are going to take the positive values of y, divide them in 115 bins and compute the mean density in each bin. Then we consider the width of the interface to be the distance between the rightmost 'dense' bin and the leftmost 'light' bin (see right plot of figure 3.9).

In order to measure the diffusion produced by each code, we use the results of the simulations without ad-hoc seeded perturbation (see section 3.3). Since we can already see some instabilities growing at times $t > 2$ in figure 3.4, we have calculated this diffusion

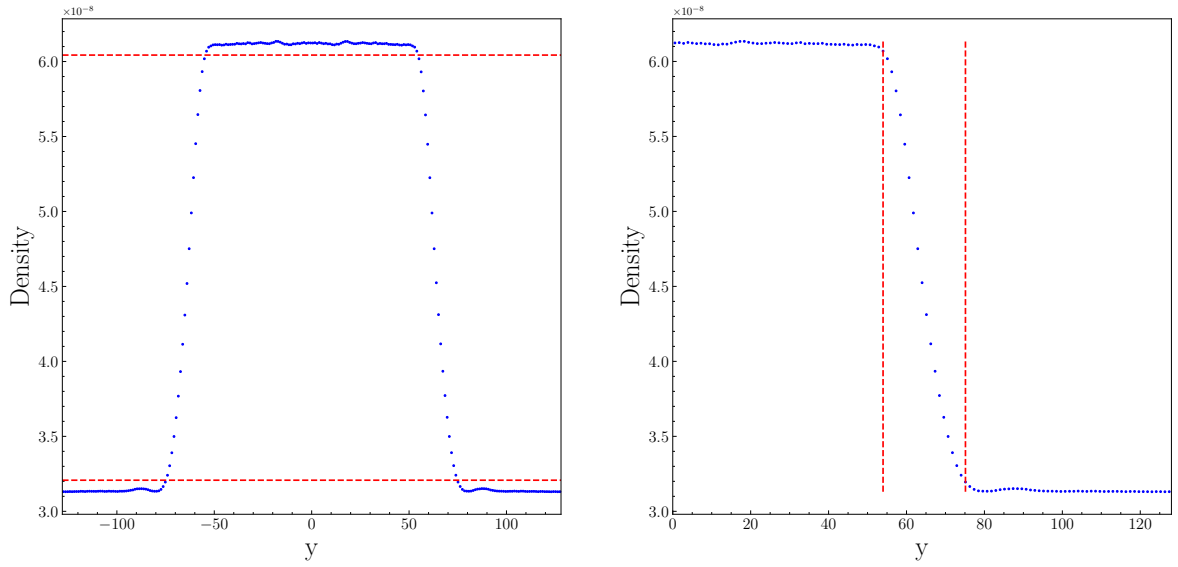

Figure 3.9: Plot of density against y position at $t = 1.5$, where the upper red dashed line indicates the minimum density for a particle to be considered 'dense' and the lower red dashed line the maximum density for a particle to be considered 'light' (left panel). Plot of the density against y position for positive values of y at $t = 1.5$, where the two vertical red dashed lines indicate the width of the interface (right panel).

until $t = 2$ as we see in figure 3.10.

The diffusion in the SPH codes is higher due to the fact that it is added artificially, while in the MFM codes the growth rate of the diffusion is lower. This could mean that we add too much artificial diffusion to the SPH codes so, in order to control this, we use a time dependent artificial conductivity (TDAC) which only takes place when needed (see section 2.3.2). Figure 3.11 shows how the results vary with the implementation of this TDAC, where we used a Wendland C^6 kernel and 295 neighbours.

The rolls are much less diffusive than in the case with a constant artificial conductivity, but the instability still grows. If we now compute the diffusion at early times as we did before we get the plot we show in figure 3.12.

The SPH simulation with TDAC follows the path of the simulations using MFM, instead of the ones with SPH. The artificial conductivity is reduced at early times, but as we can

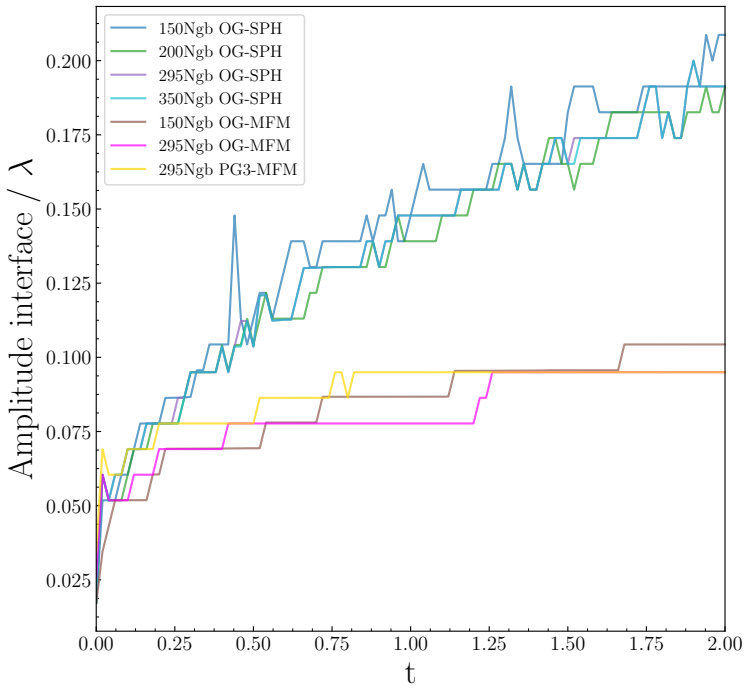

Figure 3.10: Measurement of the diffusion for the different codes at early times.

see in figure 3.11, the KHI is still able to grow and develop the desired roll. Although the simulations without AC cannot reproduce the KHI (see section 2.3.1), we plotted the results without AC in figure 3.12 just to make sure that the TDAC doesn't reduces the effect of AC to zero, but until the minimum necessary value.

3.5.2 Mixing due to the KHI

The effect of diffusion is more important in SPH than in MFM (see figure 3.10), but does that translate into more mixing in SPH than in MFM? In this section we are going to test this and, also, how the growth of the billow affects the mixing.

To do this analysis, we first plot a histogram of the density to see how it is distributed through time (figure 3.13). These histograms tell us how the number of particles per bins of density vary with time. Since we focus on the effect of the roll, we show only the results at $t = 6$ and $t = 8$.

In the case of OpenGadget-SPH code with 150 neighbours, a big amount of particles

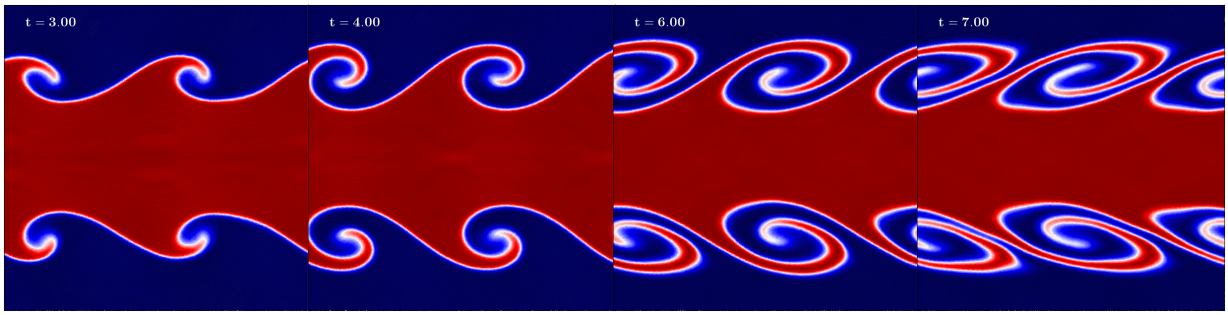

Figure 3.11: Results using a time dependent artificial conductivity with a Wendland C^6 kernel and 295 neighbours.

remain at the left and rightmost bins and there are not many particles with densities in between. This can be explained because the roll is not formed (see section 2.3.6) and, therefore, the two fluids are not mixed due to the KHI. With OpenGadget-SPH and 295 neighbours we can see much more particles in between with a peak of particles with a density around $4.1 \cdot 10^{-8}$. This peak could be a consequence of the artificial conductivity and this is likely why it is not seen in the MFM runs. Finally, for the case of the simulations using MFM, there is a similar behaviour in the two cases, where the particles are uniformly distributed among the different densities, but there is still a considerable amount of particles at the edges. We could expect this behaviour by looking at the colormaps in figure 2.11, where we could see a lack of mixing compared to SPH.

In order to analyse this effect more quantitatively, we are going to compute the ratio of 'mixed' particles/total particles through time. To do so, we set a low and high density threshold (black dashed lines in figure 3.13), between which we consider that a particle is mixed. This threshold is a 10% of the total difference between the two initial densities above (below) the initial low (high) density, so every particle that is outside that region, we consider that is not mixed yet because its density hasn't changed by more than a 10% from its initial value. Then we calculate the ratio of the particles inside this region and the total amount of particles and get the ratio of 'mixed' particles with time (figure 3.14).

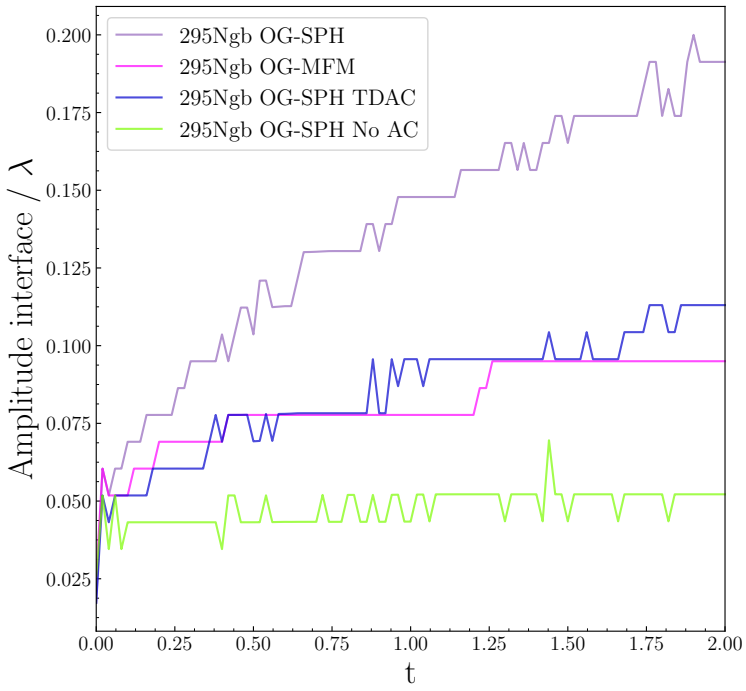

Figure 3.12: Measurement of the diffusion for the different codes at early times including the results with a time dependent artificial conductivity (TDAC) and no artificial conductivity (No AC).

Choosing a 10% for the threshold might be somewhat arbitrary, but we do not expect that this value affects the general discussion. If we look at the plots of figure 3.14, we can first see that, overall, the SPH codes tend to mix the fluids more than the MFM codes. At early times, the mixing rate in SPH increases due to artificial diffusion and, in the case that a roll is not formed (150 neighbours), the slope remains constant. In the case that the roll is formed, the slope becomes steeper due to the mixing in the billow.

This result is also in agreement with section 3.5.1: at early times the mixing rate in SPH is higher than in MFM. However, when the roll is formed[5] we can see a steeper slope for the MFM codes, which means that they start to mix the fluids more.

This analysis suggests that the mixing in SPH is mainly supported by artificial diffusion although we see a contribution of the billow, while in MFM it is more supported by

[5]Remember that the KH time-scale is $\tau_{KH} = 3.39$

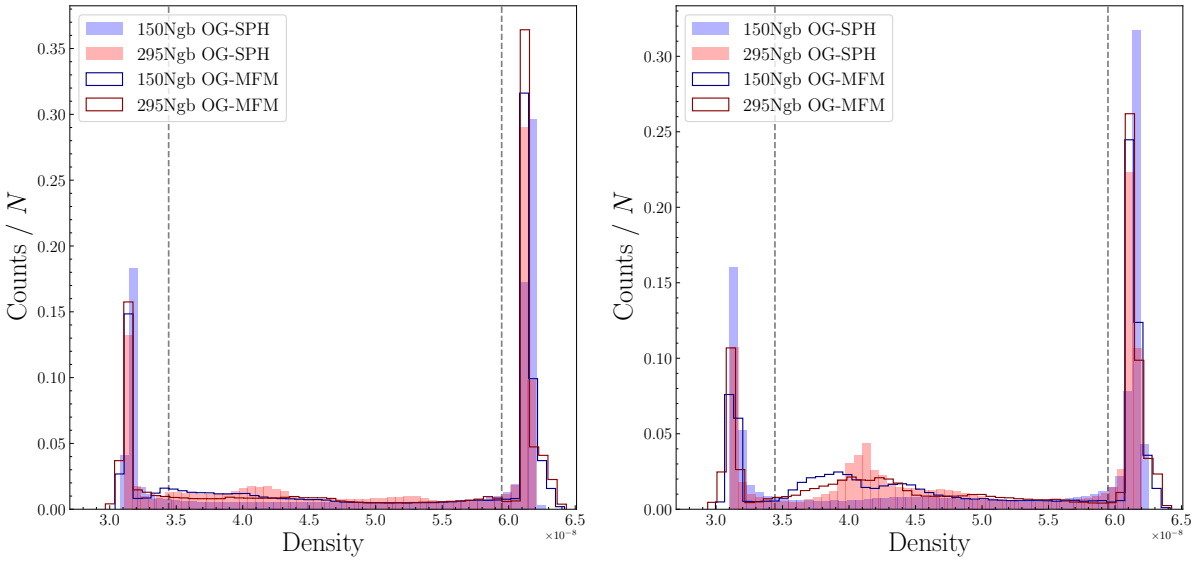

Figure 3.13: Histograms showing how the density is distributed among the particles for the cases of OpenGadget SPH and MFM with 150 and 295 neighbours at $t = 6$ and $t = 8$.

mechanical mixing due to the KH billow (Rahmani et al., 2016). This is the reason why at early times the mixing with SPH codes is steeper than MFM (see section 3.5.1), but later, when the mixing due to the billow takes place, the results with both codes have a similar slope.

3.6 Intrinsic viscosity of the codes

In section 2.3.4 we explained that we needed to add some artificial viscosity to the SPH codes in order to treat the discontinuities properly and, since this artificial viscosity is useful mainly in shocks, we wanted to keep it close to zero in our simulations. We did this using the switch explained in that section and we could see in figure 2.6 how this was successfully done by the code. In the case of MFM, the numerical diffusion comes up from solving the Riemann problem between the particles, so we don't have to take care of the amount of artificial viscosity we add. However, despite the switch in SPH keeps the artificial viscosity close to zero, we have some intrinsic viscosity that can be relevant in the

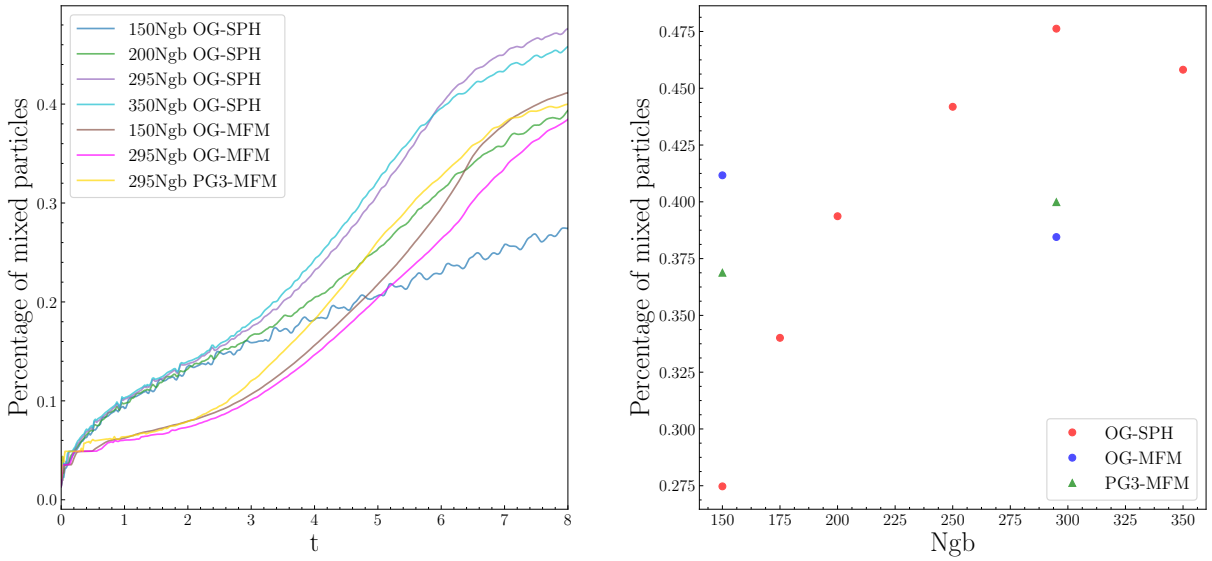

Figure 3.14: Change of mixing rate with time for the different codes (left) and maximum mixing rate depending on the number of neighbours (right). Note that in the left panel we are only considering a sub-set of the simulations in the right panel.

results which should be reduced to zero (or almost zero). In this section, we are going to measure the effective viscosity of the system for each code.

The effect of viscosity is to smooth out the velocity gradient between the two fluids by momentum diffusion and, therefore, the more viscosity a system has, the more difficult it is to develop the instability (this will be explained in section 4.4). This effect produces that the velocity in the x direction is smoothed out with time following an error function, as described in Roediger et al. (2013b)

$$v_x(y) = |v_{x_0}| \,\mathrm{erf}\left(\frac{y}{2\sqrt{\nu t}}\right), \tag{3.5}$$

where the interface is set at $y = 0$, $|v_{x_0}|$ is the initial x-velocity of one of the fluids[6] and ν is the kinematic viscosity of the system. From this formula we can see that, at $t = 0$ we have a discontinuity (the two fluids move with two different x-velocities and there is

[6]We assume that the two fluids have the same speed but in opposite directions, which is the case in our initial conditions.

a discontinuity in the y direction), but as soon as time passes, the x-velocity is smoothed out.

In order to calculate the intrinsic viscosity of our codes, we are going to simulate our two fluids without any initial perturbation, as we did in section 3.3, and we are going to fit the analytical function 3.5 to our data at different times and with the kinematic viscosity as a free parameter. To do so, we use only the top half of the full domain and we displace it to set the interface at $y = 0$. We then divide the half domain in 127 bins and compute the mean x-velocity of each bin. Now we can plot the mean value of v_x depending on y and fit the analytic formula to the data leaving ν as a free parameter. As discussed in section 3.3, at late times some perturbations can grow due to numerical noise and the two flows are not parallel to the interface anymore, that is why we decided to do the fit only until $t = 4$ to avoid spurious results. Finally, we calculated the average value of the four fits in order to get a value for the viscosity for each code. The results are shown in figure 3.15.

These results are in agreement with the ones in the previous sections and explain the behaviour of the codes in the different tests. OpenGadget-SPH with 150 is the code with the largest amount of intrinsic viscosity and, as soon as we increase the number of neighbours in the SPH code, we get less viscosity. The codes with 250, 295 and 350 neighbours have a very similar viscosity. This could mean that there is a minimum numerical viscosity that the SPH codes can reach. In the case of the MFM codes there is a big difference between OpenGadget-MFM and P-Gadget3-MFM. The P-Gadget3-MFM code has much less intrinsic viscosity than the OpenGadget-MFM code and, although this doesn't affect the height of the billow (see section 3.1), it affects its growth rate (see section 3.2).

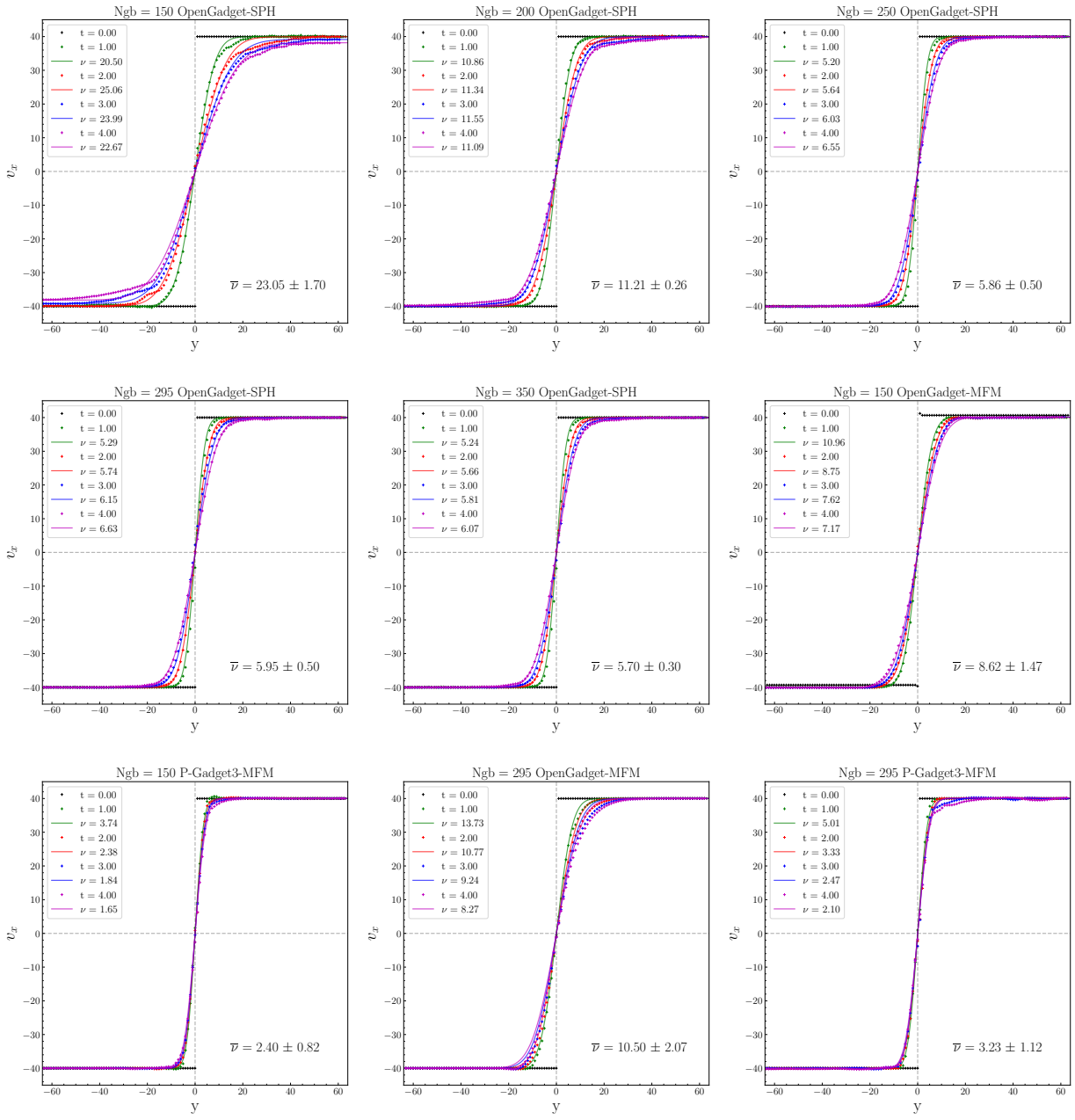

Figure 3.15: Fit to the data from our simulations with the analytic formula 3.5 to calculate the intrinsic kinematic viscosity of the different codes.

Chapter 4

Physical viscosity

So far, all the analysis made and the results discussed were always using inviscid fluids, but this is not a realistic way to describe actual fluids. In reality, fluids are viscous and the amount of viscosity of the fluid can determine its properties and behaviour. In this chapter, we will introduce viscous fluids and quantify how previous results change when the assumption of inviscid fluid does not hold anymore. We are going to implement a Braginskii viscosity as described in Sijacki and Springel (2006) and perform the same tests we did in previous sections to analyse how this physical viscosity affects the growth of the KHI.

4.1 Theory behind the physical viscosity

To describe theoretically the Braginskii viscosity, we start from the equations of hydrodynamics derived in section 1.1 and add the additional terms that account for the effect of the viscosity. For this derivation we follow Landau and Lifshitz (1987).

The continuity equation doesn't change when we add viscosity, so it remains the same also for viscous fluids, whereas the Euler equation and the heat transfer equation are altered.

4.1.1 Navier-Stokes equation

Viscosity adds an additional force to the particles due to the friction between them and, for that reason, the momentum of the particles changes. This change is included in the momentum flux density tensor (1.14) as

$$\Pi_{ik} = \rho v_i v_k + \delta_{ik} P - \sigma_{ik} \,, \tag{4.1}$$

where σ_{ik} is the viscous stress tensor defined as

$$\sigma_{ik} = \eta \left(\frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} - \frac{2}{3} \delta_{ik} \frac{\partial v_l}{\partial x_l} \right) + \zeta \delta_{ik} \frac{\partial v_l}{\partial x_l} \,. \tag{4.2}$$

The second term on the right-hand side is the bulk viscosity term, where ζ is the bulk viscosity coefficient. We can see that it only depends on the divergence of the velocity, which means that it only becomes relevant when we have a rapid compression or expansion of the fluid, i.e. shocks. The first term on the right-hand side is the shear viscosity term and η is the shear viscosity coefficient (dynamic viscosity), which was derived by Braginskii (Braginskii, 1958; Braginskii, 1965) for a fully ionized, unmagnetized plasma as:

$$\eta = 0.406 \frac{m_i^{1/2} T_i^{5/2}}{(Ze)^4 \ln \Lambda} \,, \tag{4.3}$$

where m_i is the mass of the proton, T_i is the temperature of the plasma, Ze is the ion charge and $\ln \Lambda$ is the Coulomb logarithm.

We can now write the equation of conservation of momentum (1.18) with the additional term given by the viscosity as

$$\rho \left(\frac{\partial v_i}{\partial t} + v_k \frac{\partial v_i}{\partial x_k} \right) = - \frac{\partial P}{\partial x_i} + \frac{\partial \sigma_{ik}}{\partial x_k} \,, \tag{4.4}$$

$$\rho \left(\frac{\partial v_i}{\partial t} + v_k \frac{\partial v_i}{\partial x_k} \right) = - \frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_k} \left[\eta \left(\frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} - \frac{2}{3} \delta_{ik} \frac{\partial v_l}{\partial x_l} \right) \right] + \frac{\partial}{\partial x_i} \left(\zeta \frac{\partial v_l}{\partial x_l} \right) \,. \tag{4.5}$$

Both η and ζ are positive and functions of the temperature and pressure and, since these two vary along the fluid, η and ζ cannot be taken off the partial derivatives. If the

two coefficients are constant, we get the Navier-Stokes equation, which in vector form is written as:

$$\rho\left[\frac{\partial\vec{v}}{\partial t}+(\vec{v}\cdot\nabla)\vec{v}\right]=-\nabla P+\eta\nabla^2\vec{v}+\left(\zeta+\frac{1}{3}\eta\right)\nabla\left(\nabla\cdot\vec{v}\right)\,. \tag{4.6}$$

If we rewrite equation 4.4 in Lagrangian form, we get

$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t}+\frac{1}{\rho}\nabla P=\frac{\nabla\cdot\sigma}{\rho}\,, \tag{4.7}$$

with σ being the viscous stress tensor (4.2). We can clearly see how the friction due to viscosity can be understood as an extra force affecting the movement of the particles.

4.1.2 Heat transfer equation

In this case, we start from the equation of conservation of energy in ideal fluids (1.22):

$$\frac{\partial(\rho e)}{\partial t}=-\nabla\cdot\left[(\rho e+P)\ \vec{v}\right]\,. \tag{4.8}$$

The change of the total energy of the fluid (left-hand side) must be always the same, but the energy flux density (right-hand side) doesn't. The internal friction also generates a flux, which we describe as $\vec{v}\cdot\sigma$. If the temperature is not constant, there is also a heat transfer due to thermal conduction, which is given by $\kappa\nabla T$, where κ is the coefficient of thermal conductivity (Landau and Lifshitz, 1987). If we add those terms, the general heat transfer becomes

$$\frac{\partial(\rho e)}{\partial t}=-\nabla\cdot\left[(\rho e+P)\ \vec{v}-\vec{v}\cdot\sigma-\kappa\nabla T\right]\,. \tag{4.9}$$

We can also write the heat transfer equation in terms of entropy doing some algebra

$$\frac{\partial(\rho e)}{\partial t}=-\vec{v}\cdot\nabla(\rho e+P)-(\rho e+P)\nabla\cdot\vec{v}+\nabla\cdot(\vec{v}\cdot\sigma)+\nabla\cdot(\kappa\nabla T) \tag{4.10}$$

$$\frac{\partial(\rho e)}{\partial t}=-\vec{v}\cdot\nabla(\rho e)-\vec{v}\cdot\nabla P-(\rho e+P)\nabla\cdot\vec{v}+\nabla\cdot(\vec{v}\cdot\sigma)+\nabla\cdot(\kappa\nabla T) \tag{4.11}$$

$$\frac{\mathrm{d}(\rho e)}{\mathrm{d}t}=-\vec{v}\cdot\nabla P-\rho e\nabla\cdot\vec{v}-P\nabla\cdot\vec{v}+\nabla\cdot(\vec{v}\cdot\sigma)+\nabla\cdot(\kappa\nabla T) \tag{4.12}$$

$$e\frac{\mathrm{d}\rho}{\mathrm{d}t} + \rho\frac{\mathrm{d}e}{\mathrm{d}t} = -\vec{v}\cdot\nabla P - \rho e\nabla\cdot\vec{v} - P\nabla\cdot\vec{v} + \nabla\cdot(\vec{v}\cdot\sigma) + \nabla\cdot(\kappa\nabla T)\,. \tag{4.13}$$

Now we can use the continuity equation (1.27) with the first term and replace the second term with $e = v^2/2 + u$.

$$-e\rho\nabla\cdot\vec{v} + \rho\vec{v}\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} + \rho\frac{\mathrm{d}u}{\mathrm{d}t} = -\vec{v}\cdot\nabla P - \rho e\nabla\cdot\vec{v} - P\nabla\cdot\vec{v} + \nabla\cdot(\vec{v}\cdot\sigma) + \nabla\cdot(\kappa\nabla T)\,. \tag{4.14}$$

We can substitute the time derivative of u with the first law of thermodynamics (1.36) and after cancelling out some terms, we get

$$\rho\vec{v}\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} + \rho T\frac{\mathrm{d}s}{\mathrm{d}t} + \frac{P}{\rho}\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\vec{v}\cdot\nabla P - P\nabla\cdot\vec{v} + \nabla\cdot(\vec{v}\cdot\sigma) + \nabla\cdot(\kappa\nabla T)\,. \tag{4.15}$$

If we use the continuity equation with the third term on the left-hand side, we can write the previous equation as

$$\rho\vec{v}\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} + \rho T\frac{\mathrm{d}s}{\mathrm{d}t} - P\nabla\cdot\vec{v} = -\vec{v}\cdot\nabla P - P\nabla\cdot\vec{v} + \nabla\cdot(\vec{v}\cdot\sigma) + \nabla\cdot(\kappa\nabla T)\,. \tag{4.16}$$

We can again cancel out some terms and make use of the rule $\nabla\cdot(c\cdot M) = c\cdot(\nabla\cdot M) + M : \nabla c$, where c is a vector, M is a tensor and ":" is defined as $A : B = \mathrm{tr}(A^{\mathrm{T}}\cdot B) = \mathrm{tr}(A\cdot B^{\mathrm{T}})$.

$$\rho\vec{v}\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} + \rho T\frac{\mathrm{d}s}{\mathrm{d}t} = -\vec{v}\cdot\nabla P + \vec{v}\cdot(\nabla\cdot\sigma) + \sigma : \nabla\vec{v} + \nabla\cdot(\kappa\nabla T)\,. \tag{4.17}$$

Next step is to multiply the equation of conservation of momentum (4.7) by $\vec{v}$ so it can be written as

$$\rho\vec{v}\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = -\vec{v}\cdot\nabla P + \vec{v}\cdot(\nabla\cdot\sigma)\,. \tag{4.18}$$

Plugging this into equation 4.17, we can cancel out the first and the second terms on the right-hand side and get the general equation of heat transfer in terms of entropy:

$$\rho T\frac{\mathrm{d}s}{\mathrm{d}t} = \sigma : \nabla\vec{v} + \nabla\cdot(\kappa\nabla T) \tag{4.19}$$

and in index notation

$$\rho T\frac{\mathrm{d}s}{\mathrm{d}t} = \sigma_{ik}\frac{\partial v_i}{\partial x_k} + \nabla\cdot(\kappa\nabla T)\,. \tag{4.20}$$

On the left-hand side we have the variation of the entropy multiplied by ρT. $\mathrm{d}s/\mathrm{d}t$ is the rate of change per unit mass as the volume moves throughout the fluid, $T(\mathrm{d}s/\mathrm{d}t)$ is the amount of heat gained by this unit mass per unit time and $\rho T(\mathrm{d}s/\mathrm{d}t)$ is the quantity of heat gained per unit volume. At variance with the ideal case where the variation of entropy was zero, in this case the amount of heat gained comes from the energy dissipated into heat due to viscosity (first term on the right-hand side) and the heat conducted into the volume (second term on the right-hand side) (Landau and Lifshitz, 1987).

4.1.3 Implementation

For the numerical implementation into SPH we use Greek letters (α, β, γ) for the components of the particle and Latin letters (i, j) for the different particles[1]. We can discretize the viscous stress tensor (4.2) as velocity gradients and velocity divergence as

$$\sigma_{\alpha\beta}\Big|_i = \eta \left(\frac{\partial v_\alpha}{\partial x_\beta}\Big|_i + \frac{\partial v_\beta}{\partial x_\alpha}\Big|_i - \frac{2}{3}\delta_{\alpha\beta} \frac{\partial v_\gamma}{\partial x_\gamma}\Big|_i \right) + \zeta\delta_{\alpha\beta} \frac{\partial v_\gamma}{\partial x_\gamma}\Big|_i , \tag{4.21}$$

where for the computation of the gradient in SPH we use

$$\frac{\partial v_\alpha}{\partial x_\beta}\Big|_i = \frac{1}{\rho_i} \sum_{j=1}^{N} m_j \ (\vec{v_j} - \vec{v_i})|_\alpha \ (\nabla_i W_{ij}(r, h_i))|_\beta \tag{4.22}$$

and for the divergence of the velocity we employ what we defined in section 1.2:

$$(\nabla \cdot \vec{v})_i \equiv \frac{\partial v_\alpha}{\partial x_\alpha}\Big|_i = \frac{1}{\rho_i} \sum_{j=1}^{N} m_j \ (\vec{v_j} - \vec{v_i})|_\alpha \ (\nabla_i W_{ij}(r, h_i))|_\alpha \ . \tag{4.23}$$

For the implementation, we make a difference between the acceleration due to shear viscosity and the one due to bulk viscosity. For the shear viscosity:

$$\frac{\mathrm{d}v_\alpha}{\mathrm{d}t}\Big|_{i,\ \mathrm{shear}} = \sum_{j=1}^{N} m_j \left[\frac{\eta_i \, \sigma_{\alpha\beta}\big|_i}{\rho_i^2} \left(\nabla_i W_{ij}(r, h_i)\right)\Big|_\beta + \frac{\eta_j \, \sigma_{\alpha\beta}\big|_j}{\rho_j^2} \left(\nabla_i W_{ij}(r, h_i)\right)\Big|_\beta \right] , \tag{4.24}$$

where the product of η and σ gives the shear part of the viscous stress tensor.

[1] We use the same notation as in Sijacki and Springel (2006).

For the case of the bulk viscosity, it is calculated using

$$\left.\frac{\mathrm{d}v_\alpha}{\mathrm{d}t}\right|_{i,\ \mathrm{bulk}} = \sum_{j=1}^{N} m_j \left[\frac{\zeta_i \nabla\cdot v_i}{\rho_i^2}\,\nabla_i W_{ij}(r,h_i) + \frac{\zeta_j \nabla\cdot v_j}{\rho_j^2}\,\nabla_i W_{ij}(r,h_i)\right]. \tag{4.25}$$

As we have already explained, the friction due to viscosity causes an increase in the entropy. Using equation 4.20 we can write the increase of the entropy as

$$T\frac{\mathrm{d}s}{\mathrm{d}t} = \frac{1}{\rho}\,\sigma_{ik}\frac{\partial v_i}{\partial x_k} \tag{4.26}$$

where we can write $\sigma_{ik}\,\partial v_i/\partial x_k$ as

$$\sigma_{ik}\frac{\partial v_i}{\partial x_k} = \eta\frac{\partial v_i}{\partial x_k}\left(\frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} - \frac{2}{3}\delta_{ik}\frac{\partial v_l}{\partial x_l}\right) + \zeta\frac{\partial v_i}{\partial x_k}\delta_{ik}\frac{\partial v_l}{\partial x_l} \tag{4.27}$$

$$\sigma_{ik}\frac{\partial v_i}{\partial x_k} = \frac{1}{2}\eta\left(\frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} - \frac{2}{3}\delta_{ik}\frac{\partial v_l}{\partial x_l}\right)^2 + \zeta\frac{\partial v_i}{\partial x_i}\frac{\partial v_l}{\partial x_l} \tag{4.28}$$

$$\sigma_{ik}\frac{\partial v_i}{\partial x_k} = \frac{1}{2}\eta\sigma_{ik}^2 + \zeta\left(\nabla\cdot\vec{v}\right)^2. \tag{4.29}$$

We now use the entropic function (1.60)

$$\frac{\mathrm{d}A(s)}{\mathrm{d}t} = \frac{\gamma-1}{\rho^{\gamma-1}}\,T\frac{\mathrm{d}s}{\mathrm{d}t} = \frac{\gamma-1}{\rho^{\gamma-1}}\,\frac{\sigma_{ik}}{\rho}\frac{\partial v_i}{\partial x_k}, \tag{4.30}$$

so the contribution to the increase of the entropy in the case of the shear and bulk viscosity is implemented in the code as

$$\left.\frac{\mathrm{d}A_i}{\mathrm{d}t}\right|_{\mathrm{shear}} = \frac{1}{2}\frac{\gamma-1}{\rho_i^{\gamma-1}}\frac{\eta_i}{\rho_i}\,\sigma_i^2 \qquad \left.\frac{\mathrm{d}A_i}{\mathrm{d}t}\right|_{\mathrm{bulk}} = \frac{\gamma-1}{\rho_i^{\gamma-1}}\frac{\zeta_i}{\rho_i}\left(\nabla\cdot v_i\right)^2. \tag{4.31}$$

If we want to write it in terms of the ratio of change of internal energy, we have

$$\left.\frac{\mathrm{d}u_i}{\mathrm{d}t}\right|_{\mathrm{shear}} = \frac{1}{2}\frac{\eta_i}{\rho_i}\,\sigma_i^2 \qquad \left.\frac{\mathrm{d}u_i}{\mathrm{d}t}\right|_{\mathrm{bulk}} = \frac{\zeta_i}{\rho_i}\left(\nabla\cdot v_i\right)^2. \tag{4.32}$$

4.2 Results

After having explained how the physical viscosity is implemented in our SPH code, we perform different tests in order to analyse how this Braginskii viscosity affects the growth

of the KHI. In this section we show the results we obtained for different amounts of viscosity and compared to the results obtained without physical viscosity.

As we already mentioned, since we don't have shocks in our simulations, we set the bulk viscosity coefficient ζ to zero and we work only with a shear viscosity. This affects all the particles in our domain, adding an additional friction and slowing them down (see section 4.5). We are interested in studying how different amounts of viscosity affect the system microscopically and, to do so, we are going to take fractions of the shear viscosity coefficient η (4.3). This way we can vary the amount of dynamic viscosity applied in each run and compare the results.

Since the physical viscosity has been implemented only in SPH, all the simulations made were using OpenGadget-SPH with a Wendland C^6 kernel and 295 neighbours, so it's easier to compare. We assumed for the fluids a constant temperature of $3 \cdot 10^7$ K, which is inside the range of temperatures in the intracluster medium (ICM).

4.2.1 Growth of the KHI

Here we are going to show the density colormaps of different runs to see how the growth and shape of the rolls change depending on the amount of viscosity of the fluids. This can be seen in figure 4.1.

From these results we can already see that with a large amount of viscosity the instability is suppressed and we don't get the roll up[2], but as soon as we decrease it, the instability appears and grows with higher amplitude. For the case of a viscosity of $\hat{\eta} = 10^{-4}\,\eta$, we get a similar shape to the ones we got without physical viscosity (see figure 2.7).

4.2.2 Numerical description of the growth of the KHI

From figure 4.1 we can see that the viscosity clearly affects the growth of the KHI, but how much is it affected? In order to study the effect that physical viscosity has on our results, we run 11 different simulations with 11 different fractions of Braginskii viscosity. Table

[2]This is explained in section 4.4.

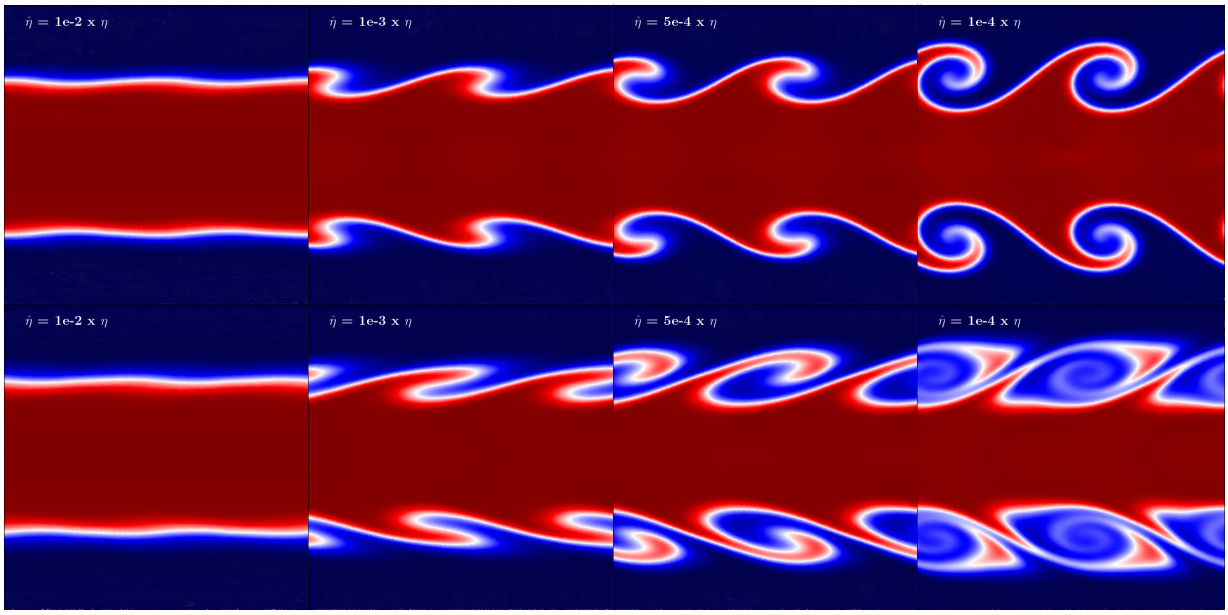

Figure 4.1: Results of the growth of the KHI using different amounts of viscosity at $t = 4$ (top row) and at $t = 7$ (bottom row).

4.1 shows the 11 different fractions with the corresponding value of the dynamic viscosity in internal units and in physical units ($\mathrm{g{\cdot}cm^{-1}{\cdot}s^{-1}}$), as well as the corresponding value of kinematic viscosity (ν).

Note that the kinematic viscosity is defined as

$$\nu = \frac{\eta}{\rho}\,, \tag{4.33}$$

which means that it's different depending on the density of the fluid. Since we have two different fluids but the dynamic viscosity is the same for both of them, in our analysis we have computed the kinematic viscosity using the mean value of the two densities.

For the quantitative study of the simulations we perform the analysis made in sections 3.1 and 3.2. With these two analysis we are able to study the growth of the KHI depending on the amount of viscosity employed. In figure 4.2 we can see how the amplitude of the billows varies depending on how much viscosity we add (left plot) and how it also affects the rate of change of the y-velocity (right plot).

The amplitude is reduced when the fluids are more viscous due to the fact that the

Fraction	$\hat{\eta}$ (g·cm^{-1}·s^{-1})	$\hat{\eta}$ (Internal Units)	ν (Internal Units)
10^{-4}	0.029	$1.379 \cdot 10^{-7}$	2.938
$2.5 \cdot 10^{-4}$	0.072	$3.449 \cdot 10^{-7}$	7.345
$5 \cdot 10^{-4}$	0.144	$6.897 \cdot 10^{-7}$	14.690
$7.5 \cdot 10^{-4}$	0.216	$1.035 \cdot 10^{-6}$	22.035
10^{-3}	0.288	$1.379 \cdot 10^{-6}$	29.380
$1.5 \cdot 10^{-3}$	0.432	$2.069 \cdot 10^{-6}$	44.071
$2 \cdot 10^{-3}$	0.576	$2.759 \cdot 10^{-6}$	58.761
$2.5 \cdot 10^{-3}$	0.720	$3.449 \cdot 10^{-6}$	73.451
$5 \cdot 10^{-3}$	1.441	$6.897 \cdot 10^{-6}$	146.902
$7.5 \cdot 10^{-3}$	2.161	$1.035 \cdot 10^{-5}$	220.354
10^{-2}	2.882	$1.379 \cdot 10^{-5}$	293.804

Table 4.1: Different amounts of viscosity employed in our simulations.

friction between particles reduces the kinetic energy (see section 4.5) causing the KHI to not develop. This means that we will have no instability at all for the simulations with the highest viscosities. At early times we can see some increase of the amplitude due to the mixing of the fluids thanks to the thermal conduction, but after this, the amplitude doesn't grow anymore.

Also, if we look at the rate of growth of y-velocity, we can see the effect already mentioned. Due to friction, there is a point where the y-velocity, instead of increasing exponentially, decreases exponentially. This is the reason why the instability doesn't develop above a certain viscosity threshold. This will be analysed in section 4.4.

It is important to note also that the results with no viscosity and with $\hat{\eta} = 10^{-4}\,\eta$ are really similar. This means that such an amount of physical viscosity is too low to be relevant and the system is governed by the numerical viscosity. Actually, if we compare the results of the intrinsic viscosity obtained in section 3.6 and the theoretical physical viscosity in table 4.1, we can see that the numerical viscosity of the code is higher than

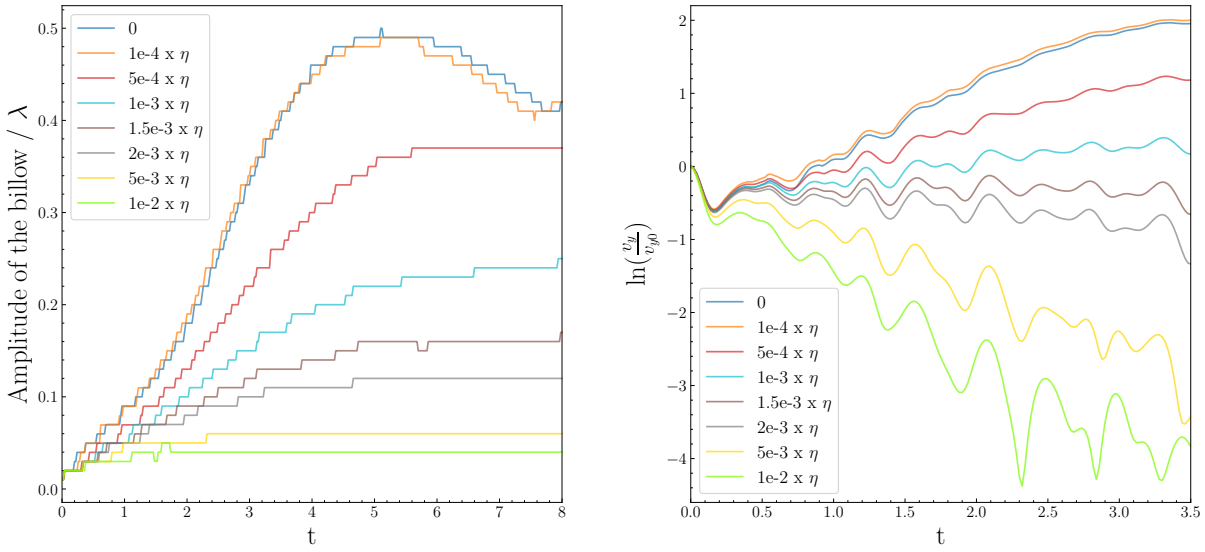

Figure 4.2: Amplitude of the simulations with different viscosities (left) and rate of growth of y-velocity depending on the amount of viscosity (right).

the physical viscosity, which can explain why both results are so similar.

The correlation between the amplitude of the billow and the viscosity is really important in order to get an idea of the amount of viscosity of a system. By measuring the amplitude of the rolls, we can estimate the suppression it's suffering and, therefore, the amount of viscosity of the system. For instance, rolls of different amplitudes can give a hint of how viscous the ICM is.

4.3 Measurement of the viscosity

In table 4.1 we can see the theoretical amount of viscosity we are implementing in the code, but is that the effective viscosity we are working with? In order to see the actual viscosity used in our simulations, we do the same analysis we made in section 3.6, but this time we have a theoretical viscosity to compare with. As we have already explained, the more viscosity we have, the more the velocity gradient between the two fluids is smoothed out. We can clearly see this effect in figure 4.3, where we compare the x-velocity profile

for two different viscosities: $\hat{\eta} = 10^{-4}\,\eta$ and $\hat{\eta} = 10^{-2}\,\eta$.

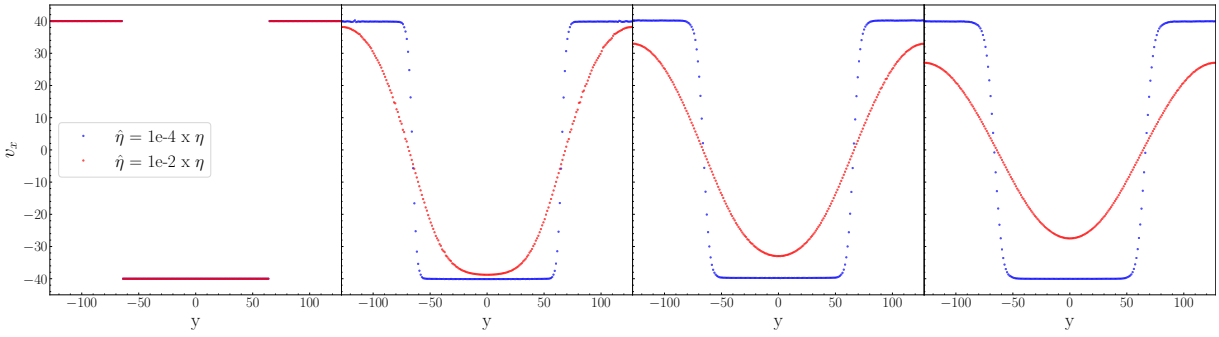

Figure 4.3: Comparison between two models assuming different values of physical viscosity: x-velocity profile at $t = 0$, $t = 1$, $t = 2$ and $t = 3$.

Figure 4.3 shows how the physical viscosity smooths out the velocity gradient between the two fluids. This is what we used in section 3.6 to measure the intrinsic viscosity of the codes and what we are also going to use here for the Braginskii viscosity.

We proceed the same way we did in section 3.6: we simulate the two fluids without perturbation in the y-axis and use equation 3.5 to fit the function to the data for different times leaving ν as a free parameter. In this case, since the viscosity suppresses the growth of numerical instabilities, we decided to use up to $t = 5$ to compute the kinematic viscosity. Here we only show three plots (figure 4.4), the rest of them can be found in appendix E.

We get fits with smaller deviation among different times for lower viscosities, while the fits predict values which are more different from one another for higher viscosities (compare e.g. right and left panel of figure 4.4). By doing this we can compute the actual viscosity that is taking place in our simulations, which is not exactly the same as the one we implemented. We can now redo table 4.1 adding the actual values of the kinematic viscosity we have just computed (table 4.2).

We can see a big discrepancy at high values of the viscosity, but up to $\hat{\eta} = 2.5 \cdot 10^{-3}\,\eta$ we can see a good correlation between the theoretical value and the computed one. Note that up to this value, the errors[3] are quite low (9% at $\hat{\eta} = 2.5 \cdot 10^{-3}\,\eta$), while at higher

[3]The errors are calculated dividing the absolute error by the value and multiplying by 100.

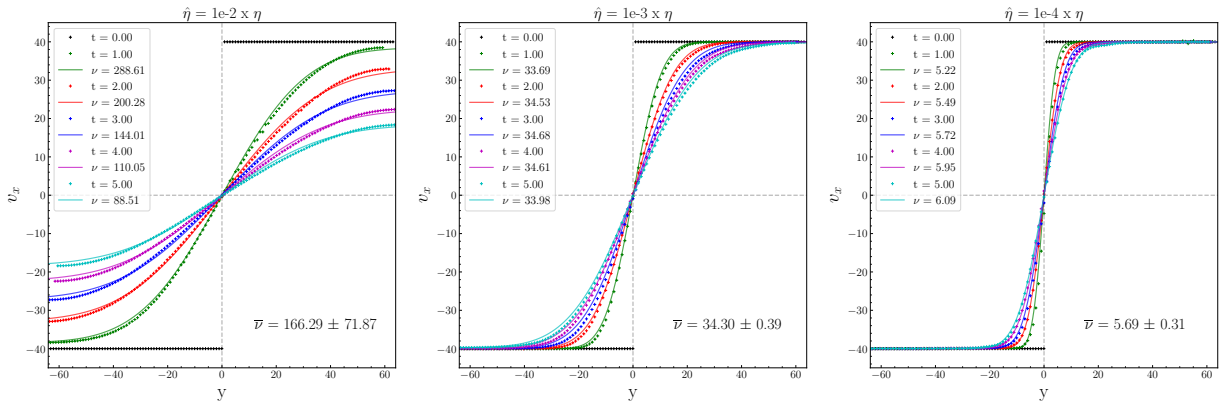

Figure 4.4: Fit to the data from our simulations with the analytic formula 3.5 to calculate the physical kinematic viscosity for the different simulations (additional plots in appendix E).

viscosities the error is higher (24% at $\hat{\eta} = 5 \cdot 10^{-3}\,\eta$).

In order to see the correlation between the theoretical (input) value and the actual (from the fit) value of the viscosity for the different runs, in figure 4.5 we have plotted the viscosity we obtained numerically versus the theoretical one. Since they are expected to be the same, the data should follow a linear relation with the intercept at $y = 0$, so we fitted our points to a linear function, but we discarded the last three values of table 4.2 due to their high error and discrepancy.

We can clearly see a linear trend with a slope of 1.0119 ± 0.0003, meaning that the growth is what we could expect, but an intercept of 3.5 ± 0.5, which means that there is a systematic shift. This systematic increase can be due to the intrinsic viscosity of the code, which takes place in addition to the implemented physical viscosity. Sijacki and Springel (2006) already stated that we still need the artificial viscosity, even if the physical viscosity is implemented, meaning that the effect of the artificial viscosity is noticeable along with the physical viscosity.

Fraction	$\hat{\eta}$ (g·cm^{-1}·s^{-1})	$\hat{\eta}$ (Internal Units)	ν (Internal Units)	Actual ν (Internal units)
10^{-4}	0.029	$1.379 \cdot 10^{-7}$	2.938	5.69 ± 0.31
$2.5 \cdot 10^{-4}$	0.072	$3.449 \cdot 10^{-7}$	7.345	10.15 ± 0.21
$5 \cdot 10^{-4}$	0.144	$6.897 \cdot 10^{-7}$	14.690	18.20 ± 0.32
$7.5 \cdot 10^{-4}$	0.216	$1.035 \cdot 10^{-6}$	22.035	26.32 ± 0.34
10^{-3}	0.288	$1.379 \cdot 10^{-6}$	29.380	34.30 ± 0.39
$1.5 \cdot 10^{-3}$	0.432	$2.069 \cdot 10^{-6}$	44.071	49.47 ± 1.40
$2 \cdot 10^{-3}$	0.576	$2.759 \cdot 10^{-6}$	58.761	63.6 ± 3.7
$2.5 \cdot 10^{-3}$	0.720	$3.449 \cdot 10^{-6}$	73.451	76.1 ± 6.9
$5 \cdot 10^{-3}$	1.441	$6.897 \cdot 10^{-6}$	146.902	122 ± 29
$7.5 \cdot 10^{-3}$	2.161	$1.035 \cdot 10^{-5}$	220.354	149 ± 53
10^{-2}	2.882	$1.379 \cdot 10^{-5}$	293.804	166 ± 72

Table 4.2: Different amounts of viscosity employed in our simulations with the actual viscosity computed.

4.4 Viscosity threshold

In section 4.2.2 we mentioned that there is a viscosity threshold above which the KHI does not develop anymore and, in this section, we are going to estimate which is this value. In section 2.1 we derived the growth of the KHI with viscosity, but that is only true for a steady background flow ($\partial v_x/\partial t = 0$), which applies only to low values of the viscosity. For higher values of viscosity, we cannot assume a steady background flow anymore ($\partial v_x/\partial t \neq 0$) due to the fact that the viscosity smooths out the x-velocity gradient. Therefore, we cannot get an analytical solution and we need to make different assumptions to estimate this viscosity threshold. To do so, we follow Roediger et al. (2013b), where they estimate three different values of the threshold depending on different assumptions.

For the numerical value of this threshold, we use the right plot of figure 4.2. In this plot we can see how there is a certain value of viscosity where the slope changes from

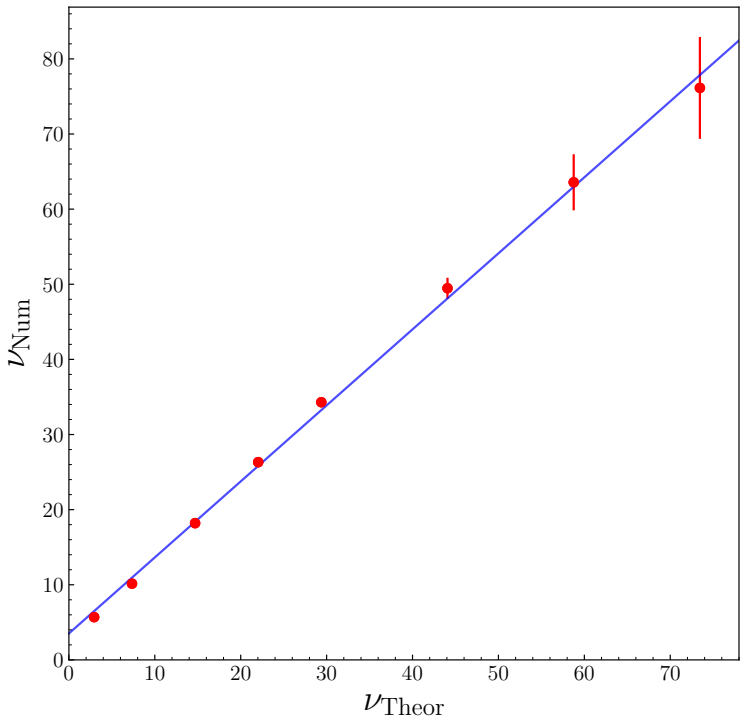

Figure 4.5: Linear fit to our data to see the correlation between theoretical and effective values.

being positive (increases exponentially) to negative (decreases exponentially). This is the value for which we consider the KHI to be totally suppressed and, therefore, our viscosity threshold. In figure 4.6 we plot the y-velocity for the lowest viscosity where the slope is positive and the highest one with a negative slope. Then we fit a linear function to the data to see if it's indeed increasing or decreasing. To do this, we used the data from $t = 0.15$ on in order to avoid the decrease explained in section 3.2 at early times.

By looking at figure 4.6 we can see that the viscosity threshold calculated numerically is between $\hat{\eta} = 1.5 \cdot 10^{-3}\,\eta$ and $\hat{\eta} = 2 \cdot 10^{-3}\,\eta$. In terms of kinematic viscosity this means that the threshold is inside a range of $\nu = 44.071 - 58.761$, which corresponds to an actual viscosity in our simulations of $\nu \approx 49 - 64$.

For the first estimate, just like Roediger et al. (2013b), we use the fact that the physical viscosity smooths out the velocity gradient over a length $\pm d$ above and below the interface.

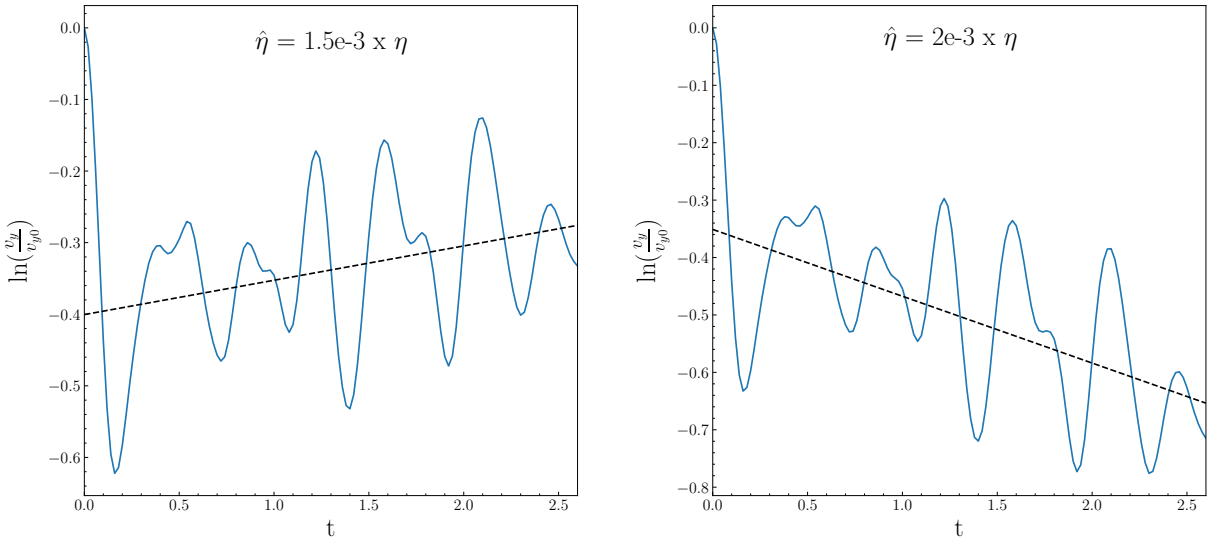

Figure 4.6: Linear fit to the y-velocity growth rate in order to see an exponential increase or decrease.

And, as demonstrated in Chandrasekhar (1961), the KHI is suppressed for wavelengths smaller than $\sim 10d$. We make use now of the diffusion length $l_D = \pm 2\sqrt{\nu\, t}$, which measures how much the interface gets widened by diffusion at time t. If we take into account that in the inviscid case, it takes $t = \tau_{KH}$ for the instability to grow, we can calculate the width of the interface at that time and see if the instability is able to grow or not. If $\lambda < 10 l_D(t = \tau_{KH})$, the KHI will be suppressed and, otherwise, it will grow. So using the definition of τ_{KH} (equation 2.5), we get

$$\lambda < 10 l_D(\tau_{KH}) = 20\sqrt{\nu\, \tau_{KH}} \tag{4.34}$$

$$\frac{\lambda^2}{400} < \nu\, \frac{\lambda}{\Delta v_x} \frac{(\rho_1 + \rho_2)}{(\rho_1\, \rho_2)^{1/2}} \tag{4.35}$$

$$\nu > \nu_{\text{Crit}} = \frac{\lambda\, \Delta v_x}{400} \frac{(\rho_1\, \rho_2)^{1/2}}{(\rho_1 + \rho_2)}\,. \tag{4.36}$$

If we plug our values in, we get a critical viscosity of $\nu_{\text{Crit}} = 12.07$, which is below what we calculated numerically. However, as Roediger et al. (2013b) state, the interface is being

smoothed out continuously and, therefore, comparing the wavelength of the perturbation with the diffusion length at $t = \tau_{KH}$ is somewhat arbitrary.

For the second estimate, we assume that the effect of the viscosity dominates when the viscous dissipation time-scale, which is given by $\tau_\nu = d^2/\nu$, is shorter than the KH time-scale τ_{KH}. As mentioned before, the KHI is suppressed if $\lambda < 10d$, so we can write d as $d = \lambda/10$. Now if we compare both time-scales, we get

$$\tau_{KH} > \tau_\nu \tag{4.37}$$

$$\frac{\lambda}{\Delta v_x}\frac{(\rho_1+\rho_2)}{(\rho_1\,\rho_2)^{1/2}} > \frac{\lambda^2}{100\nu} \tag{4.38}$$

$$\nu > \nu_{\text{Crit}} = \frac{\lambda\,\Delta v_x}{100}\frac{(\rho_1\,\rho_2)^{1/2}}{(\rho_1+\rho_2)}\,. \tag{4.39}$$

Under these assumptions, the critical value of the viscosity is four times bigger than before, leading to $\nu_{\text{Crit}} = 48.27$. This threshold correlates much better with our results and lies in the middle of the range of values we mentioned before.

Finally, we made a third estimate assuming that the instability is suppressed if, when it reaches its maximum height, the width of the x-velocity gradient is bigger that the height of the roll. The roll usually reaches a height of $\lambda/2$ at $t = \tau_{KH}$ (see figure 3.2), so this means that at $t = \tau_{KH}$ the width of the x-velocity gradient must be larger than $\lambda/2$

$$\frac{\lambda}{2} < l_D(\tau_{KH}) = 2\sqrt{\nu\,\tau_{KH}} \tag{4.40}$$

$$\frac{\lambda^2}{16} < \nu\,\frac{\lambda}{\Delta v_x}\frac{(\rho_1+\rho_2)}{(\rho_1\,\rho_2)^{1/2}} \tag{4.41}$$

$$\nu > \nu_{\text{Crit}} = \frac{\lambda\,\Delta v_x}{16}\frac{(\rho_1\,\rho_2)^{1/2}}{(\rho_1+\rho_2)}\,. \tag{4.42}$$

This gives us a value of $\nu_{\text{Crit}} = 301.70$, which is too large for our simulations. This can be due to the fact that we assumed that the maximum height is reached at $t = \tau_{KH}$, while if we see figure 3.2, it is reached at later times. This would decrease our estimate.

With the values given by these estimates, we can see that the results we obtained for the viscosity threshold are in agreement with what we have estimated. In any case, these estimates were made using very general and ideal assumptions and we cannot take the exact value we got. However, we can appreciate how our results are in keeping with theoretical expectations.

4.5 Conservation of energy

As we did with the inviscid case, we also want to see how the energy varies with time for the different runs with different amounts of viscosity. Nevertheless, the purpose here is totally different. In section 3.4 we wanted to analyse how the different codes were able to conserve the energy. Here what we want to know is how the physical viscosity affects the internal and kinetic energy, how well the total energy is conserved and how the increase of entropy changes depending on the amount of viscosity.

The effect of viscosity is to add friction between particles: as a consequence, the particle velocity decreases and so does the kinetic energy. The kinetic energy that is dissipated is turned into internal energy and the total energy is conserved. This effect can be seen in figure 4.7.

For higher values of viscosity, there is more friction between particles and more kinetic energy is turned into internal energy. We now sum up the two values in order to get the total energy. As we have already explained, since we are in a close system, the kinetic energy is transformed into internal energy, but the total energy should be conserved. We can see this in figure 4.8.

Surprisingly, the runs with the physical viscosity implemented conserve much better the total energy compared to the run without physical viscosity. Even the case with $\hat{\eta} = 10^{-4}\,\eta$ which behaves similarly to the case without physical viscosity (see sections 4.2.1 and 4.2.2), conserves much better the total energy compared to the SPH codes where the physical viscosity is not implemented (see figure 3.7).

Finally, if we analyse how the entropy varies with time, from equation 4.31 we can

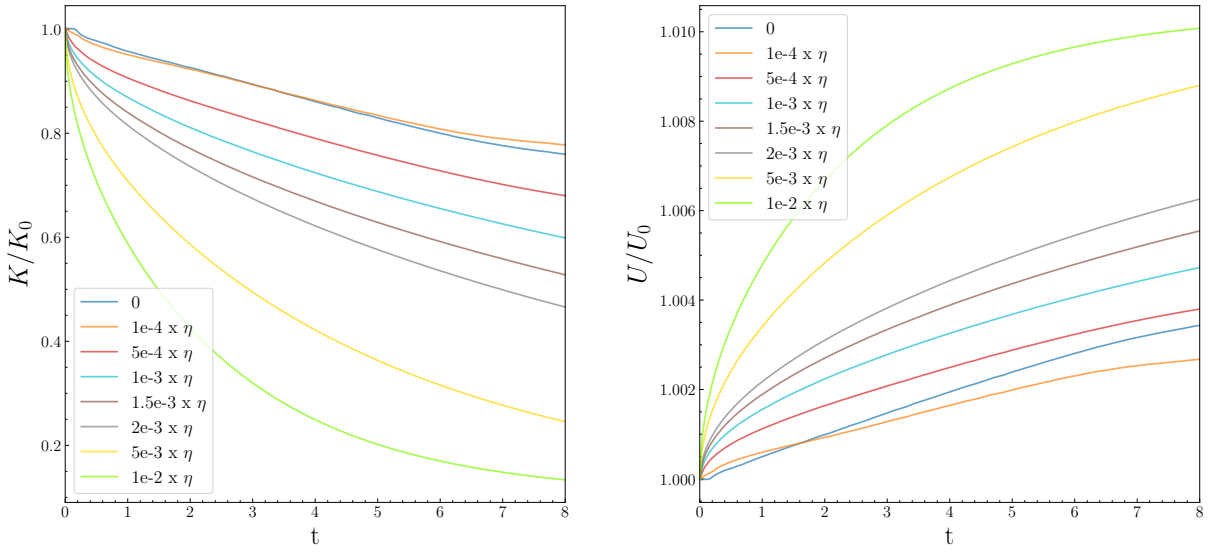

Figure 4.7: Temporal evolution of the mean kinetic energy normalized to the initial value (left) and mean internal energy normalized to the initial value (right) of the whole system for different amounts of viscosity.

see that the rate of change of the entropy increases for higher values of the viscosity. To validate this, figure 4.9 shows the variation of entropy with time for different models with different viscosities.

We can observe that there is a higher entropy growth rate for simulations with larger viscosity. For the most viscous runs, the slope is steeper at the beginning and it flattens with time, but this is not the case for the least viscous runs. In the latter cases, we can see how the slope is rather steeper at later times than at early times. This can be due to the fact that viscous fluids cannot develop a roll and, therefore, all the mixture is done by diffusion. The more time passes by, the less difference of concentration we have between the two fluids and, therefore, the rate of mixing decreases (the slope becomes less steep). For the runs that develop the roll, at first the mixing is not so important because the viscosity is lower and they are less diffusive, but then the fluids mix due to the billow of the KHI and the entropy of the system increases faster, leading to a steeper slope.

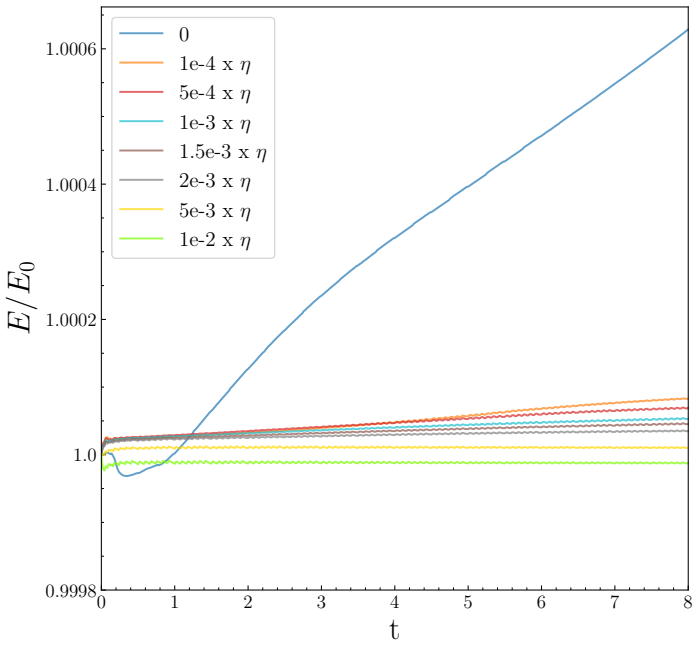

Figure 4.8: Conservation of energy per unit mass with time for different amounts of viscosity.

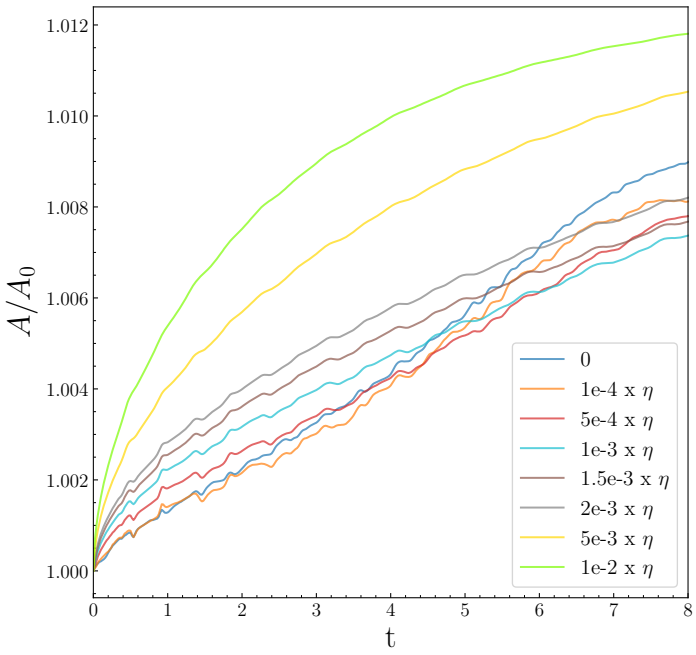

Figure 4.9: Variation of entropy per unit volume with time for different amounts of viscosity.

Chapter 5

New Initial Conditions

Throughout this project we have used the KHI to test different codes, parameters and viscosities, but in order to trigger the instability we have always used the same initial conditions. What if the results are highly dependent on the initial conditions we use? In this last chapter we are going to trigger the KHI from different initial conditions and study the dependence of the code on the initial conditions employed.

We are going to use the initial conditions suggested in Read et al. (2010), which consist of periodic boundary conditions defined by $\Delta x = 1$, $\Delta y = 1$ and $\Delta z = 1/32$ and the domain satisfies:

$$\rho, T, v_x = \begin{cases} \rho_1, T_1, v_1 & |y| < 0.25 \\ \rho_2, T_2, v_2 & |y| > 0.25 \end{cases}. \tag{5.1}$$

The densities and temperatures ratio $R_\rho = \rho_1/\rho_2 = T_2/T_1$ is equals to two, same as in the initial conditions given by Beck et al. (2015). Since no particular density or temperature is specified, we use the same densities and temperatures we were using ($\rho_1 = 6.26 \cdot 10^{-8}$ and $\rho_2 = 3.13 \cdot 10^{-8}$; $T_1 = 2.5 \cdot 10^6$ and $T_2 = 5 \cdot 10^6$). No x-velocity is specified either, but they set a mach number of $M_2 = -v_2/c_2 \approx 0.11$ and $M_1 = M_2\sqrt{R_\rho} \approx 0.15$. Due to the fact that the mach number is given by the x-velocity and the speed of sound, but the speed of sound is given by the temperature, we set the x-velocities to $v_1 = -26$ and $v_2 = 26$ in order to fulfil the value of the mach numbers.

The perturbation that triggers the instability is produced at the interface between the

two fluids, at $y_{\text{Int}} = \pm 0.25$ and is given also by equation 2.4. In this case the wavelength of the perturbation is $\lambda = 0.5$, the scale parameter σ is the same as in Beck et al. (2015), $\sigma = 0.2\lambda$ and the initial amplitude of the perturbation is $\delta v_y = |v_x|/8 = 3.25$. A summary of all the differences can be seen in table 5.1.

	Box Size	v_x	v_y (eq. 2.4)	Mach Number
Beck et al. (2015)	$256 \times 256 \times 8$	$v_{x_1} = -40$ $v_{x_2} = +40$	$\lambda = 128$ $\sigma = 0.2\lambda$ $\delta v_y = \lvert v_x \rvert/10 = 4$ $y_{\text{Int}} = \pm 64$	$M_1 \approx 0.23$ $M_2 \approx 0.17$
Read et al. (2010)	$1 \times 1 \times 1/32$	$v_{x_1} = -26$ $v_{x_2} = +26$	$\lambda = 0.5$ $\sigma = 0.2\lambda$ $\delta v_y = \lvert v_x \rvert/8 = 3.25$ $y_{\text{Int}} = \pm 0.25$	$M_1 \approx 0.15$ $M_2 \approx 0.11$

Table 5.1: Differences between the initial conditions from Beck et al. (2015) and the ones from Read et al. (2010).

Note that, since we have modified the wavelength and the x-velocity, the dynamical timescale of the perturbation has also been affected (see equation 2.5). The new growth time of the KHI is $\tau_{KH} = 0.0204$. In order to compare the results with the ones obtained in the previous sections, we are going to express the times as function of τ_{KH}.

We are going to follow the same procedure as in previous chapters, doing first a qualitative comparison between the results using the different initial conditions and then a more quantitative comparison.

5.1 Differences on the shape

For this first qualitative comparison, we show the density colormaps at different times obtained with the new initial conditions in order to compare the shape of the roll with

the old ones. In this case we simulated only the OpenGadget-SPH code with 150 and 295 neighbours to study the difference in using two different number of neighbours. Figure 5.1 shows the results of simulating the KHI with the Read et al. (2010) initial conditions with 150 and 295 neighbours.

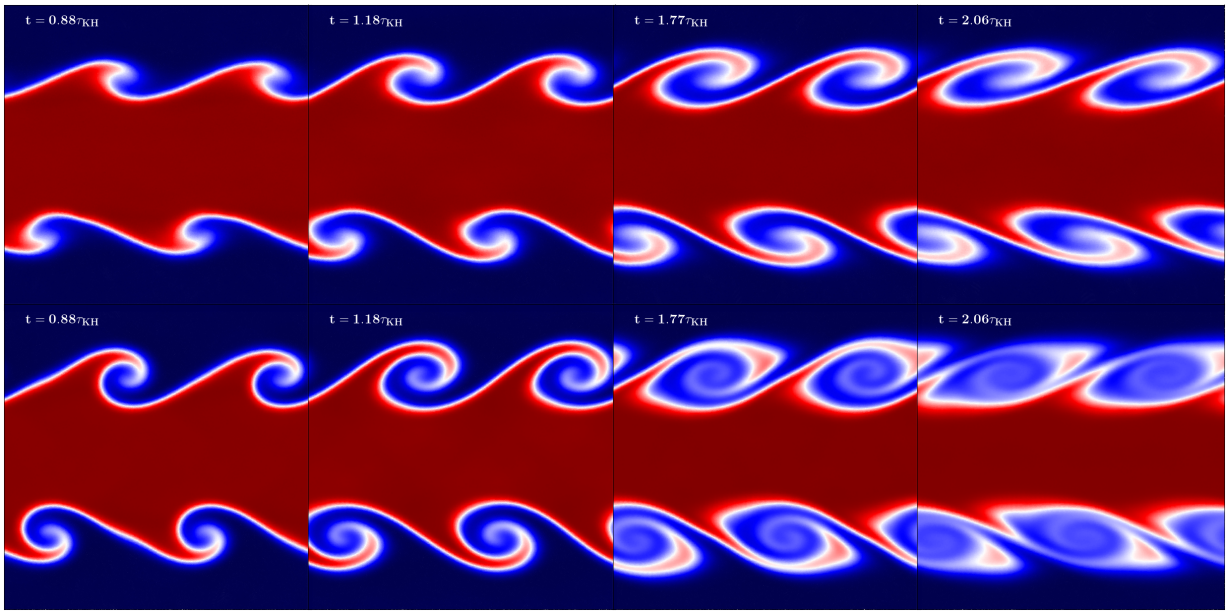

Figure 5.1: Results with the Read et al. (2010) initial conditions with 150 neighbours (top row) and with 295 neighbours (bottom row) at different times.

For a reliable comparison, the times of the different snaps of figure 5.1 correspond to the same times in terms of τ_{KH} of figure 2.7 ($t = 3$, $t = 4$, $t = 6$ and $t = 7$). Note that there is a shift between the top and the bottom rolls due to the way the initial conditions are built.

Whereas the results with 295 neighbours are similar to the ones shown in chapter 2, where the roll develops properly, the results with 150 neighbours show an improvement compared to the ones obtained with the Beck et al. (2015) initial conditions (see figure 2.8). This time the roll can partially develop in contrast to the results shown in chapter 2, where the roll was nonexistent.

5.2 Numerical differences

Figure 5.1 shows some discrepancies in the results obtained with the two initial conditions, specially in the case of 150 neighbours, so in this section we are going to analyse this in more detail.

The way we are going to study these differences is comparing the amplitudes and the growth rate using the methods explained in sections 3.1 and 3.2. Figure 5.2 shows the results for the amplitude (left) and the growth rate (right) for the Beck et al. (2015) (solid lines) and the Read et al. (2010) (dashed lines) initial conditions.

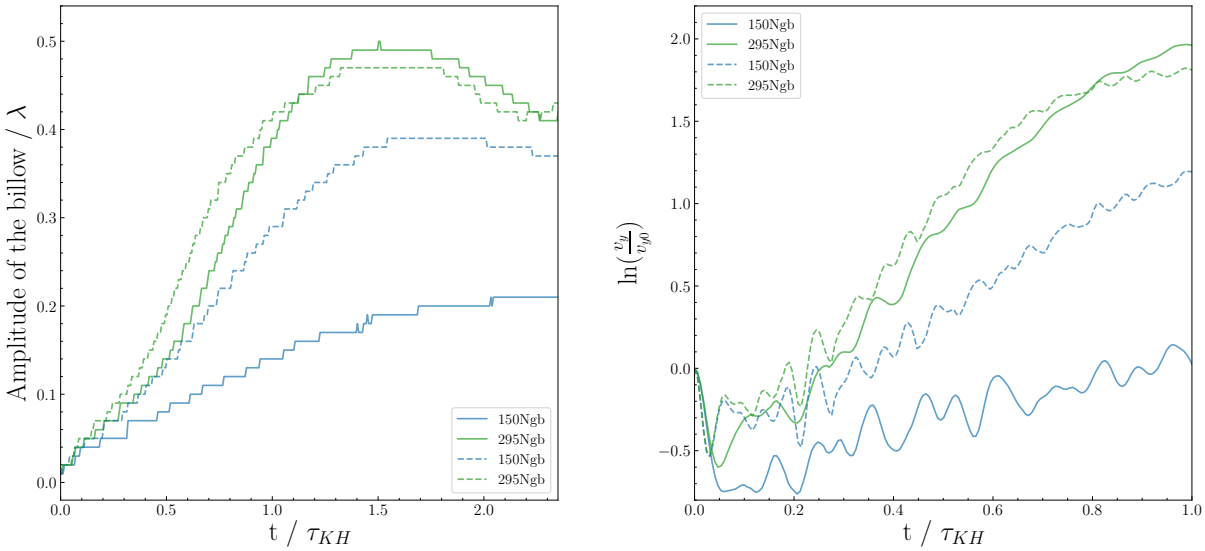

Figure 5.2: Temporal evolution of the height of the rolls (left) and evolution of the amplitude of the y-velocity (right) for the Beck et al. (2015) initial conditions (solid lines) and the Read et al. (2010) initial conditions (dashed lines) with 150 and 295 neighbours.

In agreement with the qualitative analysis, the instability triggered with the Read et al. (2010) initial conditions can grow faster and reach a higher amplitude compared to the instability triggered with the Beck et al. (2015) initial conditions in the case of 150 neighbours. The maximum amplitude reached is twice the amplitude reached with the old initial conditions. With 295 neighbours, the height of the billow with the old

initial conditions gets slightly larger, but it grows slower than the one with the new initial conditions.

The initial decrease of the amplitude of the velocity (see section 3.2) occurs slightly before with the Read et al. (2010) than with the Beck et al. (2015) initial conditions, probably provoked by the shift of the perturbation that leads in the shift of the rolls shown in figure 5.1.

5.2.1 Initial y-velocity amplitude

The different initial conditions lead to differences in the results for the growth and shape of the KHI. In order to see the main difference that leads to the discrepancy in the results, we are going to modify some of the Read et al. (2010) initial conditions to see how the results change.

First we are going to keep the same initial conditions, but modifying only the amplitude of the perturbation. We are going to set the initial amplitude of the y-velocity to $\delta v_y = |v_x|/10 = 2.6$. By reducing the initial y-velocity, we would expect a slower growth of the roll and a smaller maximum height.

Figure 5.3 shows a comparison between the results obtained modifying the initial y-velocity and the ones obtained with the original set-up. The results don't vary much compared to the initial y-velocity of $v_x/8$, but we can note that the velocity grows slightly faster, which is surprising considering that the initial amplitude of the velocity is smaller. The amplitude reached with 150 neighbours is the same in both cases, while with 295, the maximum amplitude is larger with a higher initial velocity of the perturbation.

5.2.2 Mach number

Another difference in the initial conditions is the different Mach numbers. In this section we are going to set the same Mach number in both initial conditions but we keep the rest of the things untouched. In order to set the same Mach number, we are using the Read et al. (2010) initial conditions but with $v_x = \pm 40$. As a result of this change in the x-velocity,

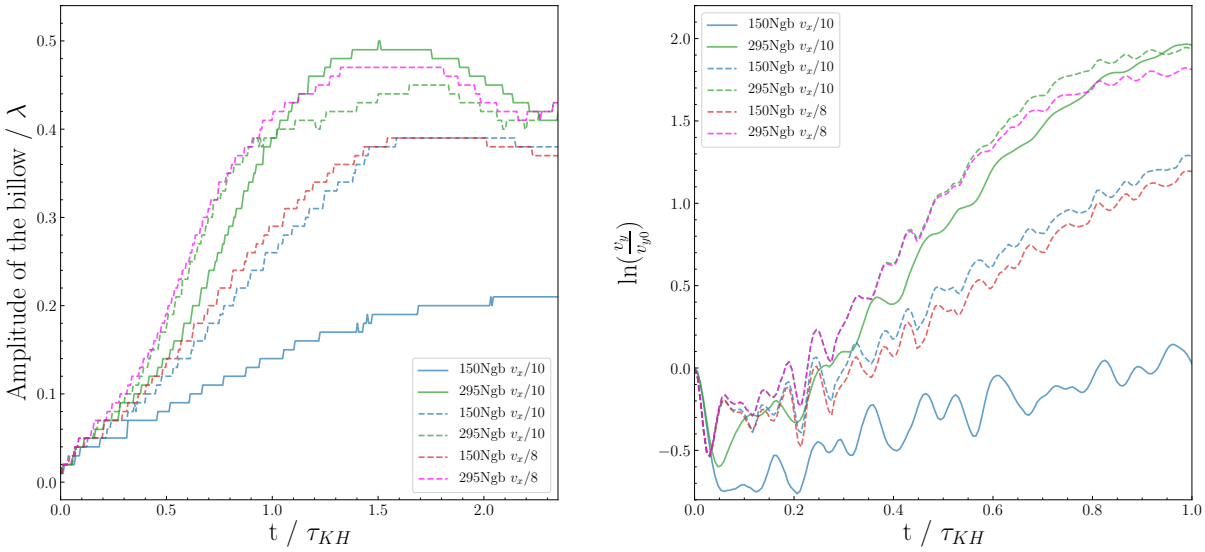

Figure 5.3: Temporal evolution of the height of the rolls (left) and evolution of the amplitude of the y-velocity (right) for the Beck et al. (2015) initial conditions (solid lines) and the Read et al. (2010) initial conditions (dashed lines) with a modified initial y-velocity (blue and green lines) with 150 and 295 neighbours.

the KHI growth time is reduced to $\tau_{KH} = 0.01326$, but since the results are normalized to the growth time, we have no problem in comparing the results (figure 5.4).

With a higher Mach number, figure 5.4 shows that the case with 150 neighbours can successfully reproduce a roll with a height close to $\lambda/2$, while for the case of 295 neighbours, the amplitude reached is even higher than using the Beck et al. (2015) initial conditions. Nevertheless, the amplitude with 295 neighbours doesn't grow linearly, indicating that there are some secondary instabilities going on. If we focus now on the growth of the y-velocity, with a higher Mach number the run with 150 neighbours grows faster, while in the case of 295 neighbours there is not a big difference with the original Read et al. (2010) initial conditions.

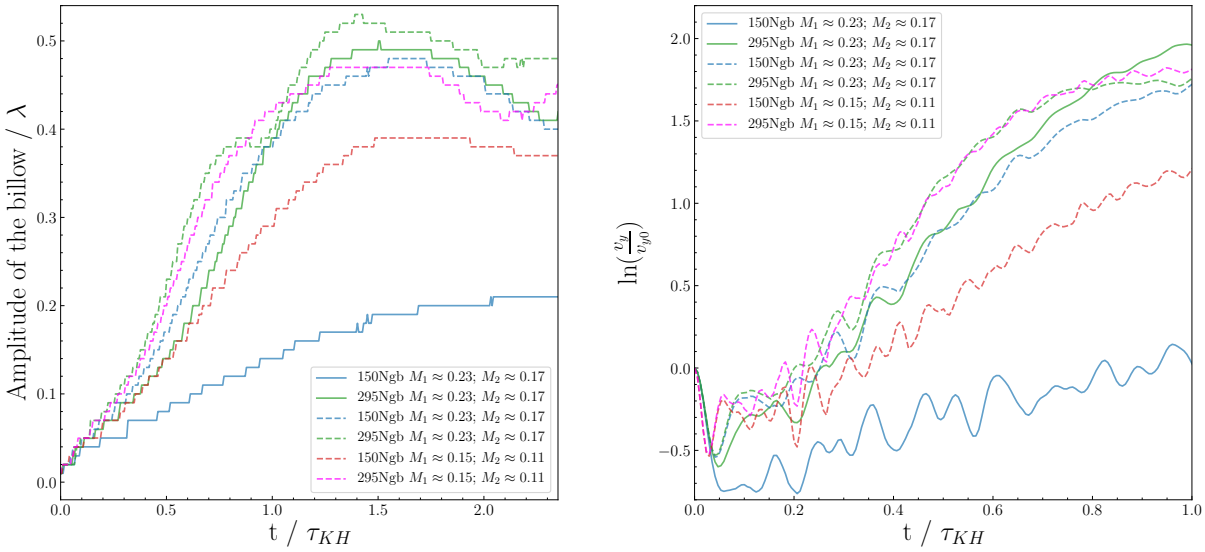

Figure 5.4: Temporal evolution of the height of the rolls (left) and evolution of the amplitude of the y-velocity (right) for the Beck et al. (2015) initial conditions (solid lines) and the Read et al. (2010) initial conditions (dashed lines) with a modified initial Mach number (blue and green lines) with 150 and 295 neighbours.

5.2.3 Initial y-velocity amplitude and Mach number

Finally, we run the simulations with the two modifications together in order to compare the results with the ones we obtained with the actual Read et al. (2010) and Beck et al. (2015) initial conditions. In this case, since the x-velocity is the same as in the previous section, the KH time-scale is $\tau_{KH} = 0.01326$ again. The results with these initial conditions are shown in figure 5.5 together with the results with the actual Read et al. (2010) and Beck et al. (2015) initial conditions.

By adding the two modifications to the Read et al. (2010) initial conditions, the results in the amplitude (left plot in figure 5.5) are similar compared to the ones obtained modifying only the Mach number, meaning that the modifications produced by the change of the Mach number are more relevant than the ones by the initial y-velocity. The rate of growth in the case of 150 neighbours is also similar to the previous result, while in the case of 295 neighbours the y-velocity grows faster.

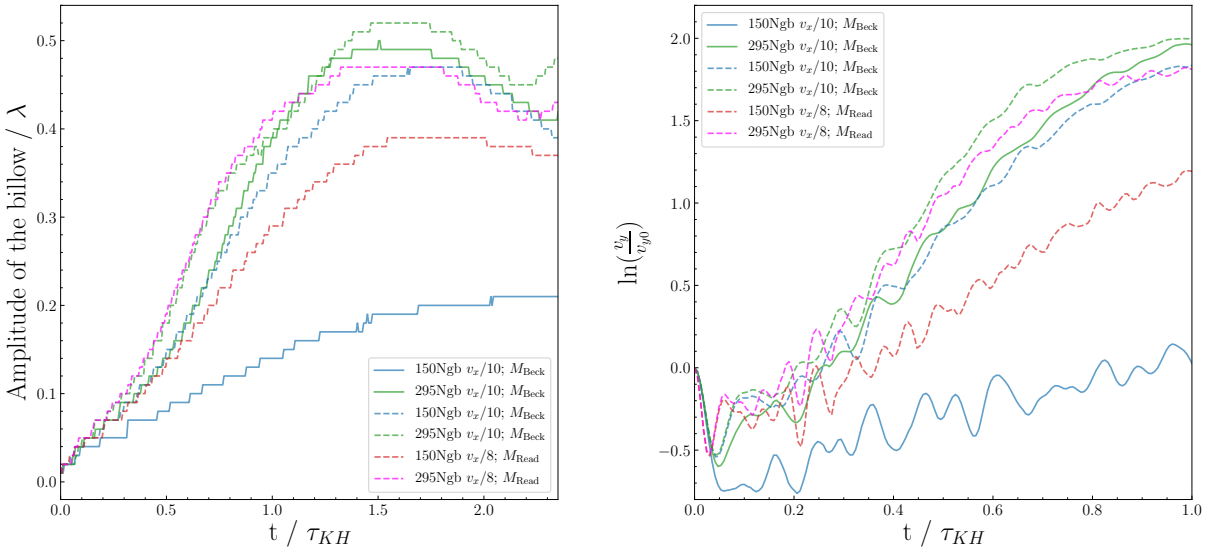

Figure 5.5: Temporal evolution of the height of the rolls (left) and evolution of the amplitude of the y-velocity (right) for the Beck et al. (2015) initial conditions (solid lines) and the Read et al. (2010) initial conditions (dashed lines) with a modified initial Mach number and y-velocity (blue and green lines) with 150 and 295 neighbours. The notation M_{Beck} corresponds to $M_1 \approx 0.23$; $M_2 \approx 0.17$ and M_{Read} to $M_1 \approx 0.15$; $M_2 \approx 0.11$.

Conclusions

In this work we have analysed the performance of both SPH and MFM codes in reproducing the KHI varying the parameters used to compute the physical quantities. Additionally, we performed different tests in order to study the properties of each code and how they affect the results obtained with the KHI. Once we had compared the codes with different parameters, we studied the effect of physical viscosity in the KHI simulations with the SPH code for different amounts of viscosity. Finally, we analysed how the initial conditions used for triggering the KHI influence the final results by comparing the original initial conditions we were using with new initial conditions.

In the SPH code we found that a correct treatment of the discontinuities in shear flows is fundamental for mixing processes and, therefore, the implementation of an artificial conductivity is needed. In order to reproduce the KHI, we had to increase the number of neighbours used for the computation and, to do so, a switch to a Wendland C^6 kernel was required to avoid the pairing instability and get satisfactory results. We also tested the addition of an artificial viscosity, which should be kept to zero in shear flows, with very positive results, as well as the addition of wake-up to the simulations. We found that a low number of neighbours (< 250) can affect negatively our results in terms of the shape of the roll, so choosing the correct number of neighbours can determine whether you reproduce the KHI successfully or not. In the case of MFM we used two different codes: OpenGadget-MFM and P-Gadget3-MFM. We tested a cubic spline kernel with 32 and 50 neighbours, finding that we could get the roll of the KHI but the secondary instabilities produce the breakdown of the billow in both cases. Nevertheless, using a

Wendland C^6 kernel the secondary instabilities were suppressed and the shape of the roll was satisfactorily achieved with both 150 and 295 neighbours using OpenGadget-MFM, while we still got some secondary instabilities with P-Gadget3-MFM. We also tested the performance of OpenGadget-MFM with two different slope limiters, finding that with the GIZMO slope limiter the roll evolves faster than with the Springel slope limiter, but with satisfactory results in both cases.

After a first qualitative analysis of the performance, we did a deeper quantitative comparison of the codes. We found that the numerical viscosity produced intrinsically by each code can determine their performance in the simulations. The SPH codes with a low number of neighbours have a high intrinsic viscosity, while above 250 neighbours, a minimum intrinsic viscosity is reached. The OpenGadget-MFM code has more intrinsic viscosity than the SPH codes with a high number of neighbours, but less than the SPH codes with a low number of neighbours, while the P-Gadget3-MFM code happens to have the lowest intrinsic viscosity. The effect of the intrinsic viscosity is noticeable when measuring the amplitude of the rolls, leading to small amplitudes in the SPH codes below 250 neighbours and with a maximum amplitude reached with 295 neighbours. The performance of the MFM codes is very similar with 150 and 295 neighbours. By analysing the growth of the y-velocity, we could see that the intrinsic viscosity produces that the growth in the runs with a low number of neighbours in SPH codes is slower than the cases with a high number of neighbours. In MFM, although the amplitude is not affected by the intrinsic viscosity, the higher viscosity in OpenGadget-MFM produces that the rate of growth is slower than in the simulations performed by P-Gadget3-MFM. Despite a low intrinsic viscosity allows the instability to grow more easily, it also allows the numerical noise to trigger spurious instabilities. This doesn't happen in the cases of SPH below 250 neighbours and OpenGadget-MFM with 295 neighbours, which are able to keep the numerical noise low. We also measured the degree of mixing due to the KHI in each code, concluding that the overall mixing is higher in SPH due to the artificial diffusion. This artificial diffusion takes place mainly at early times, where the mixing is significantly higher in SPH than in MFM, but at later times, where the mixing is produced primarily

by the KHI billow, the mixing rate is similar in both schemes. By studying the energy conservation of the codes we could see that, in general, all the codes are able to conserve the energy of the system, specially the P-Gadget3-MFM codes.

Overall, the simulations with SPH are very sensitive to the number of neighbours employed. A low number of neighbours could lead to unphysical results, while a simulation with a number of neighbours above 250 (specially 295) is able to reproduce the KHI satisfactorily. The results with MFM are not dependent on the number of neighbours employed above 150, but the case with OpenGadget-MFM and 295 neighbours is able to keep the numerical noise low while at the same time reproduce the KHI successfully.

After the analysis of the different codes, we run several simulations with physical viscosity in the SPH codes with 295 neighbours. We added different amounts of viscosity in order to see how the results varied depending if the fluids were more or less viscous. We found that the viscosity can affect significantly the results, leading from the total suppression of the instability in very viscous fluids, to no apparent effect in the cases with a low viscosity. By varying the amount of viscosity employed in the simulations, we found a threshold below which the instability is fully suppressed. This threshold, calculated numerically, was in agreement with three different estimates we made based on very general assumptions. We also measured the effective viscosity of the fluids, finding that the actual viscosity of the system is not only the physical viscosity implemented in the code, but it is slightly higher. This probably means that the effective viscosity of the simulation is an addition of the physical viscosity plus some effect due to the intrinsic viscosity of the code. We also found that the runs with physical viscosity tend to conserve better the energy than the runs without physical viscosity.

Finally, we performed some analysis with SPH varying the initial conditions in order to see how this could affect the results. The runs with 150 neighbours are very sensitive to modifications of the initial conditions, while with a higher number of neighbours (295 in this case) there are still some changes, but the overall behaviour remains similar. From these simulations we could also conclude that the runs with a higher Mach number tend to produce higher rolls even with a low number of neighbours. The consequences of varying

the Mach number happened to be more relevant than changing in the initial y-velocity of the perturbation. Finally, despite we run the simulations with the same initial perturbation and same Mach number than our original initial conditions, we could still see big differences in the behaviour of the runs, meaning that the size of the domain can play an important role in the results.

Bibliography

Agertz, O., B. Moore, J. Stadel, D. Potter, F. Miniati, J. Read, L. Mayer, A. Gawryszczak, A. Kravtsov, A. Nordlund, and et al.

2007. Fundamental differences between sph and grid methods. *Monthly Notices of the Royal Astronomical Society*, 380(3):963–978.

Agertz, O., R. Teyssier, and B. Moore

2009. Disc formation and the origin of clumpy galaxies at high redshift. *Monthly Notices of the Royal Astronomical Society: Letters*, 397(1):L64–L68.

Balsara, D. S.

1995. von neumann stability analysis of smoothed particle hydrodynamics—suggestions for optimal algorithms. *Journal of Computational Physics*, 121(2):357–372.

Barth, T. J. and D. C. Jespersen

1989. The design and application of upwind schemes on unstructured meshes.

Beck, A. M., G. Murante, A. Arth, R.-S. Remus, A. F. Teklu, J. M. F. Donnert, S. Planelles, M. C. Beck, P. Förster, M. Imgrund, K. Dolag, and S. Borgani

2015. An improved SPH scheme for cosmological simulations. *Monthly Notices of the Royal Astronomical Society*, 455(2):2110–2130.

Braginskii, S. I.

1958. Transport phenomena in a completely ionized two-temperature plasma. *Soviet Phys. JETP*, 6.

Braginskii, S. I.

1965. Transport Processes in a Plasma. *Reviews of Plasma Physics*, 1:205.

Breuer, J. P., N. Werner, F. Mernier, T. Mroczkowski, A. Simionescu, T. E. Clarke, J. A. ZuHone, and L. Di Mascolo

2020. The mergers in abell 2256: displaced gas and its connection to the radio-emitting plasma. *Monthly Notices of the Royal Astronomical Society*, 495(4):5014–5026.

Burkert, A.

2006. The turbulent interstellar medium. *Comptes Rendus Physique*, 7(3-4):433–441.

Chandrasekhar, S.

1961. *Hydrodynamic and hydromagnetic stability*.

Cullen, L. and W. Dehnen

2010. Inviscid smoothed particle hydrodynamics. *Monthly Notices of the Royal Astronomical Society*, 408(2):669–683.

Davé, R., R. Thompson, and P. F. Hopkins

2016. mufasa: galaxy formation simulations with meshless hydrodynamics. *Monthly Notices of the Royal Astronomical Society*, 462(3):3265–3284.

Dehnen, W. and H. Aly

2012. Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. *Monthly Notices of the Royal Astronomical Society*, 425(2):1068–1082.

Dong, R. and J. M. Stone

2009. Buoyant bubbles in intracluster gas: Effects of magnetic fields and anisotropic viscosity. *The Astrophysical Journal*, 704(2):1309–1320.

Dullemond, C. and H. Wang

2009. Lecture notes in numerical fluid mechanics.

Eckart, C.

1960. Variation principles of hydrodynamics. *Physics of Fluids*, 3:421–427.

Gaburov, E. and K. Nitadori

2011. Astrophysical weighted particle magnetohydrodynamics. *Monthly Notices of the Royal Astronomical Society*, 414(1):129–154.

Ge, C., R.-Y. Liu, M. Sun, H. Yu, L. Rudnick, J. Eilek, F. Owen, S. Dasadia, M. Rossetti, M. Markevitch, and et al.

2020. Chandra and xmm–newton observations of a2256: cold fronts, merger shocks, and constraint on the ic emission. *Monthly Notices of the Royal Astronomical Society*, 497(4):4704–4717.

Gendron-Marsolais, M., J. Hlavacek-Larrondo, R. J. van Weeren, L. Rudnick, T. E. Clarke, B. Sebastian, T. Mroczkowski, A. C. Fabian, K. M. Blundell, E. Sheldahl, K. Nyland, J. S. Sanders, W. M. Peters, and H. T. Intema

2020. High-resolution VLA low radio frequency observations of the Perseus cluster: radio lobes, mini-halo, and bent-jet radio galaxies. *Monthly Notices of the Royal Astronomical Society*, 499(4):5791–5805.

Gingold, R. A. and J. J. Monaghan

1977. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389.

Hopkins, P. F.

2015. A new class of accurate, mesh-free hydrodynamic simulation methods. *Monthly Notices of the Royal Astronomical Society*, 450(1):53–110.

Hopkins, P. F., A. Wetzel, D. Kereš, C.-A. Faucher-Giguère, E. Quataert, M. Boylan-Kolchin, N. Murray, C. C. Hayward, S. Garrison-Kimmel, C. Hummels, and

et al.

2018. Fire-2 simulations: physics versus numerics in galaxy formation. *Monthly Notices of the Royal Astronomical Society*, 480(1):800–863.

Hu, C.-Y., T. Naab, S. Walch, B. P. Moster, and L. Oser

2014. Sphgal: smoothed particle hydrodynamics with improved accuracy for galaxy simulations. *Monthly Notices of the Royal Astronomical Society*, 443(2):1173–1191.

Junk, V., S. Walch, F. Heitsch, A. Burkert, M. Wetzstein, M. Schartmann, and D. Price

2010. Modelling shear flows with smoothed particle hydrodynamics and grid-based methods. *Monthly Notices of the Royal Astronomical Society*, 407(3):1933–1945.

Landau, L. D. and E. M. Lifshitz

1987. *Fluid Mechanics*, volume 6 of *Course of Theoretical Physics*, second edition. Pergamon.

Lucy, L. B.

1977. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024.

Luo, H., J. Baum, and R. Lohner

2008. A discontinuous galerkin method using taylor basis for compressible flows on arbitrary grids. *Journal of Computational Physics*, 227:8875–8893.

McNally, C. P., W. Lyra, and J.-C. Passy

2012. A well-posed kelvin-helmholtz instability test and comparison. *The Astrophysical Journal Supplement Series*, 201(2):18.

Monaghan, J.

1997. Sph and riemann solvers. *Journal of Computational Physics*, 136(2):298–307.

Monaghan, J. and R. Gingold

1983. Shock simulation by the particle method sph. *Journal of Computational Physics*, 52(2):374–389.

Monaghan, J. J.

1992. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30:543–574.

Monaghan, J. J. and J. C. Lattanzio

1985. A refined particle method for astrophysical problems. *Astronomy and Astrophysics*, 149(1):135–143.

Obergaulinger, M. and M. Aloy

2020. Numerical viscosity in simulations of the two-dimensional kelvin-helmholtz instability. *Journal of Physics: Conference Series*, 1623:012018.

Pakmor, R., P. Edelmann, F. K. Röpke, and W. Hillebrandt

2012. Stellar GADGET: a smoothed particle hydrodynamics code for stellar astrophysics and its application to Type Ia supernovae from white dwarf mergers. *Monthly Notices of the Royal Astronomical Society*, 424(3):2222–2231.

Price, D. J.

2008. Modelling discontinuities and kelvin–helmholtz instabilities in sph. *Journal of Computational Physics*, 227(24):10040–10057.

Price, D. J.

2011. Smoothed particle hydrodynamics: Things i wish my mother taught me.

Price, D. J.

2012. Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics*, 231(3):759–794.

Price, D. J., J. Wurster, T. S. Tricco, C. Nixon, S. Toupin, A. Pettitt, C. Chan, D. Mentiplay, G. Laibe, S. Glover, and et al.

2018. Phantom: A smoothed particle hydrodynamics and magnetohydrodynamics code for astrophysics. *Publications of the Astronomical Society of Australia*, 35.

Rahmani, M., B. Seymour, and G. Lawrence

2014. The evolution of large and small-scale structures in kelvin–helmholtz instabilities. *Environmental Fluid Mechanics*, 14:1275–1301.

Rahmani, M., B. Seymour, and G. Lawrence

2016. The effect of prandtl number on mixing in low reynolds number kelvin-helmholtz billows. *Physics of Fluids*, 28:054107.

Read, J. I., T. Hayfield, and O. Agertz

2010. Resolving mixing in smoothed particle hydrodynamics. *Monthly Notices of the Royal Astronomical Society*, P. no–no.

Roediger, E., R. P. Kraft, W. R. Forman, P. E. J. Nulsen, and E. Churazov

2013a. Kelvin-helmholtz instabilities at the sloshing cold fronts in the virgo cluster as a measure for the effective intracluster medium viscosity. *The Astrophysical Journal*, 764(1):60.

Roediger, E., R. P. Kraft, P. Nulsen, E. Churazov, W. Forman, M. Brüggen, and R. Kokotanekova

2013b. Viscous Kelvin–Helmholtz instabilities in highly ionized plasmas. *Monthly Notices of the Royal Astronomical Society*, 436(2):1721–1740.

Rosswog, S.

2015. Boosting the accuracy of sph techniques: Newtonian and special-relativistic tests. *Monthly Notices of the Royal Astronomical Society*, 448(4):3628–3664.

Saitoh, T. R. and J. Makino

2009. A Necessary Condition for Individual Time Steps in SPH Simulations. *The Astrophysical Journal Letters*, 697(2):L99–L102.

Sander, B. and G. Hensler

2021. Physical effects on compact high-velocity clouds in the circumgalactic medium. *mnras*, 501(4):5330–5349.

Schaal, K., A. Bauer, P. Chandrashekar, R. Pakmor, C. Klingenberg, and V. Springel

2015. Astrophysical hydrodynamics with a high-order discontinuous galerkin scheme and adaptive mesh refinement. *Monthly Notices of the Royal Astronomical Society*, 453(4):4279–4301.

Schoenberg, I.

1946. Contributions to the problem of approximation of equidistant data by analytic functions. Part A. On the problem of smoothing or graduation. A first class of analytic approximation formulae. *Quart. Appl. Math.*, 4:45–99.

Schuessler, I. and D. Schmitt

1981. Comments on smoothed particle hydrodynamics. *Astronomy and Astrophysics*, 97(2):373–379.

Sijacki, D. and V. Springel

2006. Physical viscosity in smoothed particle hydrodynamics simulations of galaxy clusters. *Monthly Notices of the Royal Astronomical Society*, 371(3):1025–1046.

Springel, V.

2005. The cosmological simulation code gadget-2. *Monthly Notices of the Royal Astronomical Society*, 364(4):1105–1134.

Springel, V.

2010. E pur si muove:galilean-invariant cosmological hydrodynamical simulations on a moving mesh. *Monthly Notices of the Royal Astronomical Society*, 401(2):791–851.

Springel, V.

2014. High performance computing and numerical modelling.

Steinwandel, U. P., B. P. Moster, T. Naab, C.-Y. Hu, and S. Walch

2020. Hot phase generation by supernovae in ism simulations: resolution, chemistry, and thermal conduction. *Monthly Notices of the Royal Astronomical Society*, 495(1):1035–1060.

Tricco, T. and D. Price

2013. A switch for artificial resistivity and other dissipation terms.

Tricco, T. S.

2019. The kelvin-helmholtz instability and smoothed particle hydrodynamics. *Journal of Physics: Conference Series*, 1225:012019.

Wendland, H.

1995. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4:389–396.

ZuHone, J. A., M. W. Kunz, M. Markevitch, J. M. Stone, and V. Biffi

2015. The Effect of Anisotropic Viscosity on Cold Fronts in Galaxy Clusters. *Astrophysical Journal*, 798(2):90.

List of Figures

List of Tables

Appendix A

Derivation of the dynamical timescale for the KHI

The growth time of the KHI is defined as

$$\tau_{KH} = \frac{2\pi}{\omega} = \frac{2\pi}{k\,\Delta v_x}\frac{(\rho_1+\rho_2)}{(\rho_1\cdot\rho_2)^{1/2}} = \frac{\lambda}{\Delta v_x}\frac{(\rho_1+\rho_2)}{(\rho_1\cdot\rho_2)^{1/2}}\,, \tag{A.1}$$

where ω is the frequency, k the wavenumber, Δv_x the difference in velocities of both fluids and ρ the densities of both fluids (Agertz et al., 2007).

To derive equation A.1 we are going to get the dispersion relation following Landau and Lifshitz (1987). To do so, we assume two layers of an ideal fluid moving one with respect to the other in the x-axis. For simplicity, we choose the reference system where one of them has zero velocity ($v_{x_2} = 0$, $v_{x_1} = v$). Now we consider a small perturbation (v') in the y-velocity (perpendicular to the surface) proportional to $\mathrm{e}^{i(kx-\omega t)}$. This perturbation satisfies the following system of equations for ideal fluids:

$$\nabla \cdot v' = 0\,, \tag{A.2}$$

$$\frac{\partial v'}{\partial t} + (v\cdot\nabla)v' = -\frac{\nabla P'}{\rho}\,. \tag{A.3}$$

Since v only has x-component, we can rewrite equation A.3 as

$$\frac{\partial v'}{\partial t} + v\frac{\partial v'}{\partial x} = -\frac{\nabla P'}{\rho}\,. \tag{A.4}$$

If we take the divergence in both sides, the left side is zero because of equation A.2, which means that P' satisfies the Laplace's equation:

$$\nabla^2 P' = 0\,. \tag{A.5}$$

Due to this perturbation, the pressure will change following a similar formula to the velocity, also proportional to $\mathrm{e}^{i(kx-\omega t)}$, so we define it as $P' = f(y)\,\mathrm{e}^{i(kx-\omega t)}$. Using equation A.5, we get that the function $f(y)$ satisfies $\mathrm{d}^2 f/\mathrm{d}y^2 - k^2 f = 0$ and if we solve this differential equation we get that $f(y) = C\,\mathrm{e}^{\pm ky}$, where C is a constant. If we consider the upper part of the system (positive values of y), we get that the pressure in the upper part follows:

$$P_1' = C\,\mathrm{e}^{i(kx-\omega t)}\,\mathrm{e}^{-ky}\,. \tag{A.6}$$

If we now substitute this expression in the y-component of equation A.4 we get[1][2].

$$v_y' = \frac{k\,P_1'}{i\,\rho_1(kv-\omega)}\,. \tag{A.7}$$

We now introduce the displacement in the y-axis due to the perturbation as $\zeta = \zeta(x,t)$, which derivative is the rate of change of the displacement for a given x. So, the derivative can be written as

$$v_y' = \frac{\mathrm{d}\zeta}{\mathrm{d}t} = \frac{\partial\zeta}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial\zeta}{\partial t} = v\,\frac{\partial\zeta}{\partial x} + \frac{\partial\zeta}{\partial t}\,, \tag{A.8}$$

where the total derivative of the displacement with time corresponds to the y-velocity of the particles of the fluid. Since the displacement is also proportional to $\mathrm{e}^{i(kx-\omega t)}$, equation A.8 becomes $v_y' = i\,\zeta\,(kv-\omega)$. If we now equalize equations A.7 and A.8, we get

$$P_1' = -\frac{\zeta\,\rho_1\,(kv-\omega)^2}{k}\,. \tag{A.9}$$

On the lower part (negative values of y), we get a similar formula, but this time $v = 0$ and a different sign due to the fact that we have e^{+ky} instead of e^{-ky}, so the pressure for the lower layer is given by

$$P_2' = \frac{\zeta\,\rho_2\,\omega^2}{k}\,. \tag{A.10}$$

[1] Note that v_y' is also proportional to $\mathrm{e}^{i(kx-\omega t)}$, so $\partial v_y'/\partial t = -i\,\omega\,v_y'$ and $\partial v_y'/\partial x = i\,k\,v_y'$.

[2] Although the case where $\omega = kv$ can occur, it's not of interest here due to the fact that the instability can only grow for complex values of ω.

The system must be in pressure equilibrium, which means that $P_1' = P_2'$, therefore we get that $\rho_1 \, (kv - \omega)^2 = -\rho_2 \, \omega^2$. This gives us the dispersion relation we were looking for

$$\omega = kv \, \frac{\rho_1 \pm i\sqrt{\rho_1 \, \rho_2}}{\rho_1 + \rho_2} \, . \tag{A.11}$$

We get a complex ω, where we can distinguish between the real and the imaginary part

$$\omega_{\mathrm{Re}} = \frac{kv \, \rho_1}{\rho_1 + \rho_2}; \qquad \omega_{\mathrm{Im}} = \pm \frac{kv \, \sqrt{\rho_1 \, \rho_2}}{\rho_1 + \rho_2} \, . \tag{A.12}$$

Taking into account that the growth of the perturbation is given by $\mathrm{e}^{i(kx - \omega t)}$, it will only grow exponentially with a positive imaginary part (Springel, 2014). Therefore, the growth time of the instability is

$$\tau_{KH} = \frac{2\pi}{\omega_{\mathrm{Im}}} = \frac{2\pi}{kv} \frac{(\rho_1 + \rho_2)}{\sqrt{\rho_1 \, \rho_2}} \, , \tag{A.13}$$

which is the expression we wanted to derive.

Appendix B

Pairing instability

In figure B.1 we can see the effect of the pairing instability. In this run we used a cubic spline kernel with 295 neighbours.

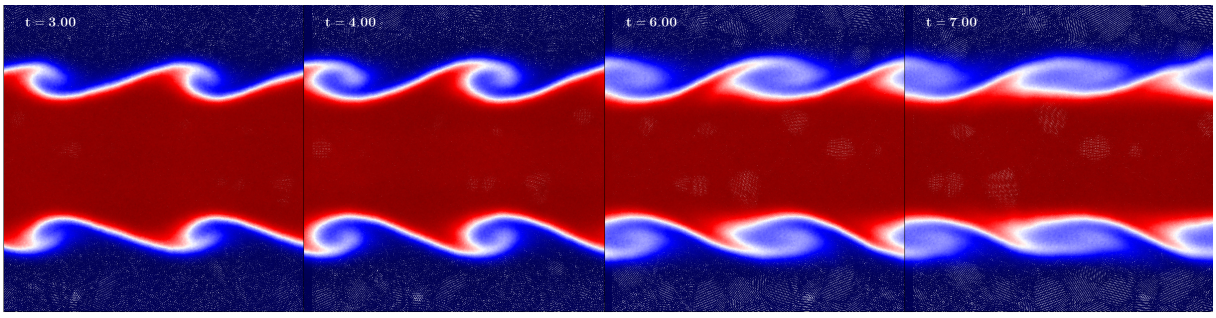

Figure B.1: Effect of the pairing instability in our simulations at $t = 3$, $t = 4$, $t = 6$ and $t = 7$.

Appendix C

Wendland C^6 kernel with 160 and 175 neighbours

Here we can see the results for running the code with a Wendland C^6 kernel and 160 and 175 neighbours. As mentioned in section 2.3.6, we can see a difference in the results when we change the number of neighbours. In this case, neither 160 nor 175 neighbours can develop the roll.

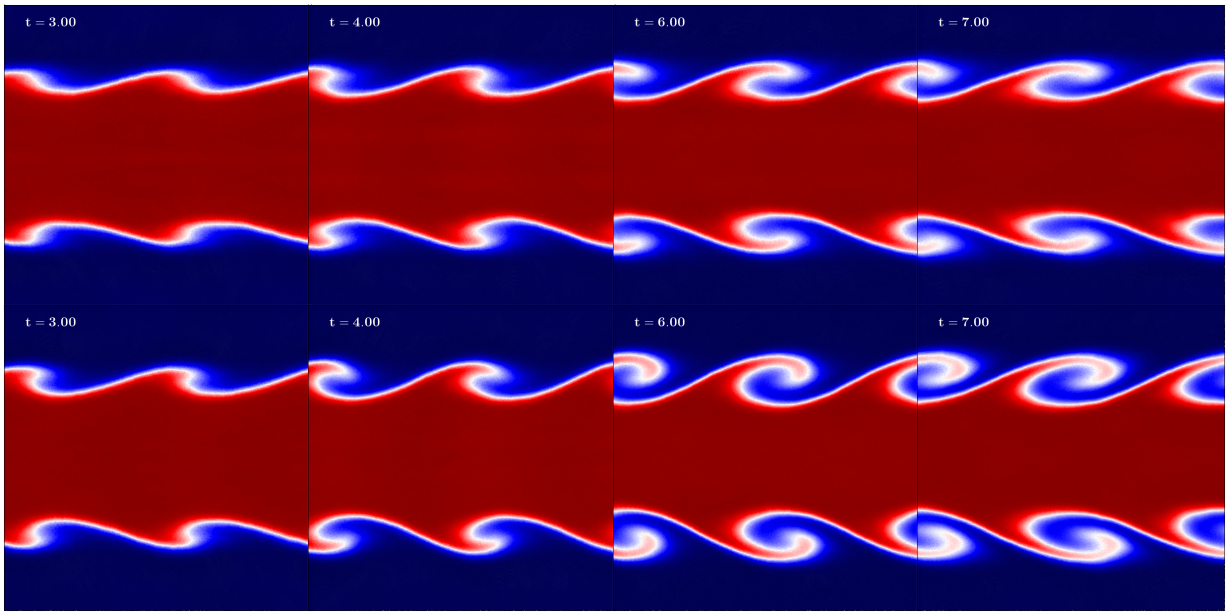

Figure C.1: Wendland C^6 kernel with 160 and 175 neighbours at $t = 3$, $t = 4$, $t = 6$ and $t = 7$.

Appendix D

MFM code with a Wendland C^6 kernel and 150 neighbours

Here we can see the performance of the MFM and a Wendland C^6 kernel with 150 neighbours for both cases: OpenGadget-MFM and P-Gadget3-MFM.

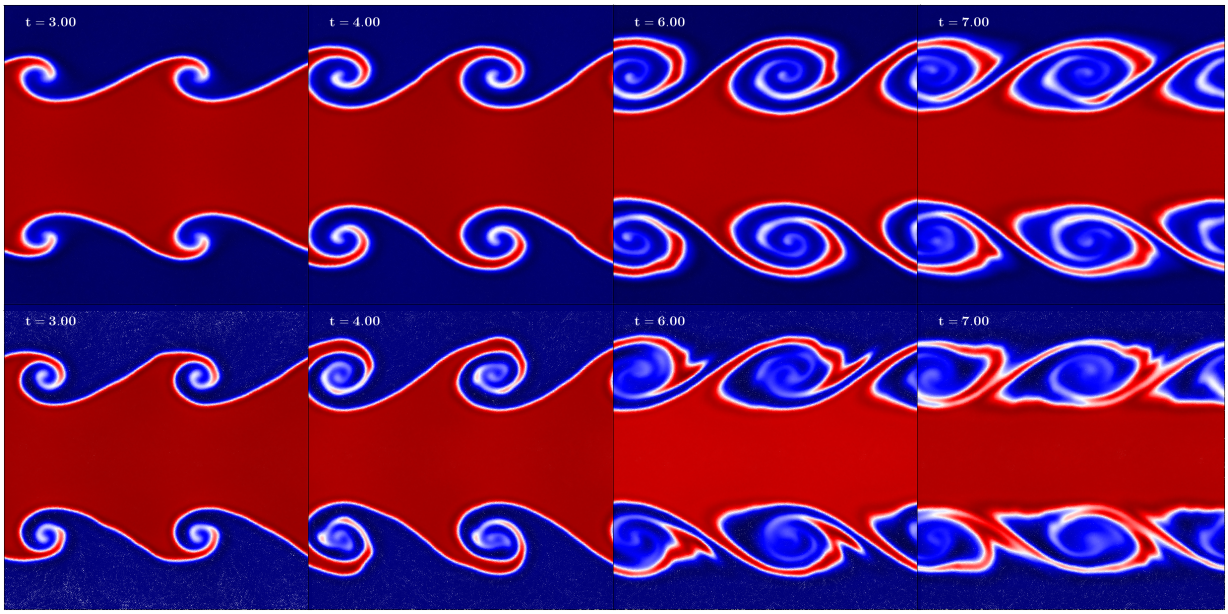

Figure D.1: Comparison of the results of OpenGadget-MFM and P-Gadget3-MFM both with a Wendland C^6 kernel and 150 neighbours.

Appendix E

Kinematic viscosity

Here we have the rest of the plots of the fit to our data using function 3.5 in order to compute the value of ν for every simulation.

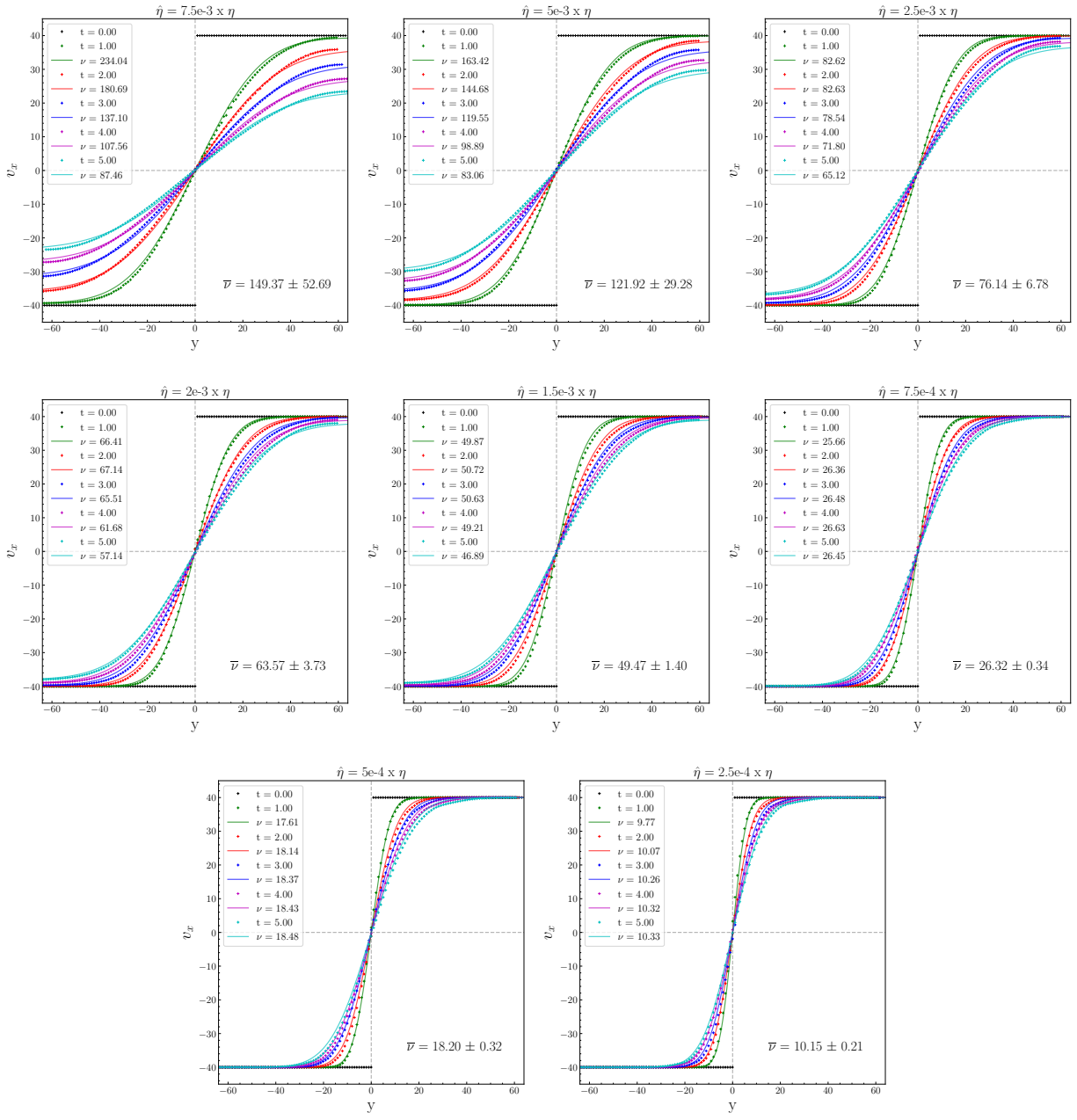

Figure E.1: Fit to the data from our simulations with the analytic formula 3.5 to calculate the physical kinematic viscosity for the different simulations.

Acknowledgements

After all this journey I am really grateful to many people that has helped me along this year. First of all, to my supervisors Klaus Dolag, Milena Valentini and Ulrich Steinwandel for your support and your help. It has been a weird year in which we were only able to meet in person a few times but, despite that, you have always been willing to make everything easier for me and I really appreciate that. Thank you so much for giving me this opportunity.

Thank you also to all the CAST group for making me feel comfortable here. The online meetings are very impersonal, but with you I always felt part of the group and welcome from the first meeting. I must say that I am such a shy person (specially with zoom meetings) but you did the meetings so much easier. Of course, special thanks to Elena for insisting that this group was the best choice I could do for my master thesis and you were totally right.

I cannot forget all my friends here in Munich. These two years hasn't been easy with the Corona, lockdown, restrictions, etc. but you were there to take a beer and forget about it; or to go for a trip and don't think of Corona for a while; or meet for dinner and enjoy having fun with you guys. Thank you so much for those little moments that really made me enjoy despite the tough days.

Quiero agradecer también a todos esos amigos tanto en Alicante como en Valencia, con los que, cada vez que vuelvo a España, es como si no me hubiera ido nunca. Gracias en especial a Luis, muchas gracias hermano por estar ahí durante tantos años.

Por último, pero no menos importante, muchas gracias a mi familia. Sabéis que este

siempre ha sido mi sueño desde muy pequeño y en todo momento habéis hecho todo lo posible porque luchara por ello. Hoy es el día con el que había soñado tanto tiempo, el día en el que acaba mi etapa como estudiante y empieza mi carrera como astrofísico. No ha sido un camino fácil, pero habéis estado ahí en cada decisión que he tomado y sin vuestro apoyo estoy seguro de que hoy no podría estar escribiendo estas palabras. Muchas gracias, os quiero mucho.

Declaration:

I hereby declare that this thesis is my own work, and that I have not used any sources and aids other than those stated in the thesis.

München, 26.10.2021

_____

Tirso Marín Gilabert