Facultad de Ciencias. Sección de Física
Máster Universitario en Astrofísica

TRABAJO DE FIN DE MÁSTER

Mesh-free hydrodynamics: theory and validation of the method

Isaac Alonso Asensio

Tutor:

Dr. Claudio Dalla Vecchia

Cotutor:

Dr. Andrés Balaguera Antolínez

Contents

Abstract

In this *Trabajo de Fin de Máster* a state-of-the-art hydrodynamic solver has been studied and implemented, namely the Meshless Finite Volume scheme (MFV). We have heuristically derived the latter scheme after a brief review of the most used scheme in hydrodynamic, cosmological simulations: Smoothed Particle Hydrodynamics (SPH).

All the implementation has been carried out inside a state-of-the-art cosmological N-body code, `PKDGRAV3`. To take advantage of the parallelism strategy of the latter code, it has been studied in depth. Its structure and general workflow has been understood to allow for an easy and non-intrusive implementation of the hydrodynamics. In this way, we can successfully deploy our code in a laptop or in a high-performance, parallel computer, achieving good scaling properties.

The hydrodynamics has been implemented independently from other codes to provide another test of the scheme, as there are few codes using MFV. In some cases, they provide significantly different results, even when running standard hydrodynamic tests.

We have tested our implementation with a set of cases with known (analytical) solution. With them, we checked that our code: is second-order in space; allow for individual time steps to decrease the time-to-solution; it does not artificially transport angular momentum; it provides adaptive resolution thanks to its Lagrangian nature and conserves energy and mass to machine accuracy. We find that another code implementing the MFV scheme, namely, the public version of GIZMO, is not fully conservative as stated in the literature.

In the future, we expect to add the gravity forces to the scheme and all the physics required to provide Large Scale Structure simulations. Knowing that this is a difficult task due to the demanding computational requirements, we have implemented the scheme having in mind a future porting to Graphics Processing Units (GPUs) to further decrease the time-to-solution.

Resumen

En este *Trabajo de Fin de Máster* hemos implementado un esquema numérico de vanguardia para la resolver las ecuaciones de la hidrodinámica: Meshless Finite Volume (MFV). Hemos derivado este esquema numérico tras una breve descripción de Smoothed Particle Hydrodynamics (SPH), el esquema más usado hoy en día en el contexto de las simulaciones cosmológicas.

Toda la implementación se ha realizado dentro de un código de última generación para resolver el problema de N-cuerpos en contextos cosmológicos, `PKDGRAV3`. Para poder aprovechar al máximo la estrategia de paralelización de este, lo hemos estudiado en detalle. Hemos examinado tanto su estructura como su funcionamiento para poder implementar de manera sencilla y no intrusiva la hidrodinámica. De esta manera, podemos ejecutar nuestro código en un portátil o un nodo de altas prestaciones sin mayores problemas.

El código se ha desarrollado de la manera más independiente posible para probar que MFV funciona como se espera, ya que hay pocos códigos que usen dicho esquema. Incluso, en algunos casos, hay diferentes códigos que dan diferentes resultados usando el mismo esquema.

Hemos puesto a prueba nuestra implementacion con diferentes casos con solución conocida. Gracias a ellos, hemos comprobado que nuestro código: es de segundo orden espacial; permite el uso de pasos de tiempo individuales para disminuir el tiempo de cómputo; no transporta momento angular de manera artificial; adapta su resolución espacial gracias a la naturaleza Lagrangiana del método y conserva masa y energía hasta errores de redondeo numérico. Por otra parte, hemos encontrado que GIZMO, que también usa MFV, no es totalmente conservativo, tal y como se afirma en el artículo que describe el código.

En el futuro queremos añadir la gravedad al esquema numérico, además de toda la física que se requiere para realizar simulaciones de Estructura a Gran Escala. Teniendo en mente que esta es una tarea difícil debido a su alto coste computacional, hemos implementado el esquema pensando en una futura versión capaz de usar Tarjetas Gráficas (GPUs) para acelerar los cálculos.

Chapter 1

Introduction

Throughout this chapter, a brief justification of why we need to take into account gas dynamics in cosmological simulations is presented (section 1.1). Therein, the most used formalism, Smooth Particles Hydrodynamics, will be briefly explained (section 1.1.1). The state-of-the-art Meshless Finite Volume formalism, which is the core of the present work, will be derived (section 1.1.2).

Afterwards, the base code within which all the development has been carried out, `PKDGRAV3`, will be detailed in section 1.2.

1.1 Gas dynamics

Approximately only 15% of the matter content of the universe is formed by baryons (Planck Collaboration et al., 2018), being this the only component that we can detect through direct observations. Therefore, correctly simulating the baryonic component is crucial to compare the results of simulations with observations. The primordial baryonic matter is mainly formed by a mix of Hydrogen and Helium. Matter self-interact through gravity whilst baryons also self-interact via pressure forces.

At small scales, the pressure slows down the collapse of baryons inside dark matter halos. Indeed, unless cooling effects are taken into account, the gas is not compressed enough to form stars (Schaye et al., 2010). Furthermore, the gas dynamics have been proven to be able to modify the dark matter density profiles of virialized halos thanks to feedback due to exploding supernovae. This latter effect has been studied extensively during recent years with different simulations (Benitez-Llambay et al., 2018; Di Cintio et al., 2013).

In summary, nowadays taking into account the gas dynamics in cosmological simulations is of paramount relevance in order to accurately describe the dynamics of dark matter tracers and to derive cosmological parameters (Balaguera-Antolínez and Porciani, 2013).[1] From a numerical point of view, there are two main ways to simulate the evolution of

[1] Otherwise, there are semi-analytical and empirical models to populate with galaxies dark matter halos from N-body simulations, but these statistical methods are out of the scope of this work.

the gas throughout cosmic time. The difference between both approaches lies on the discretization of the gas elements.

In the first ones, gas elements are represented by particles that in general move with the fluid velocity, thus these schemes are of Lagrangian nature. These schemes have the great advantage that they naturally adapt the resolution where there is a higher density of particles. This is essential for cosmological simulations, where the matter density ranges over ~ 10 orders of magnitude (i.e. ,we can have enormous voids with almost no particles, and extremely dense regions in the centre of halos, all within the same computational volume). It has been used successfully in plenty of simulations (see Bahé et al., 2017; Schaye et al., 2015; Springel, 2005; Wang et al., 2015, for examples).

The second ones are extensively used in other fields of Astrophysics such as Solar Physics (e.g., Felipe et al., 2010; Gudiksen et al., 2011, among others), and engineering (e.g., Lani et al., 2013, OPENFOAM, and others). In those fields, the density contrast inside the computational volume is small enough (although it can be of a few orders of magnitude) that the space can be subdivided in a grid. In general, the nodes of grid do not move with time, and thus these methods are of Eulerian nature.

Although we will not discuss the latter for cosmological simulations, it is important to notice that there are codes that use grids. They either use adaptive ones (e.g., AREPO, Nelson et al., 2018; Springel, 2010b) or subdivide the grid where more resolution is needed (e.g., ENZO or RAMSES, Bryan et al., 2014; Teyssier, 2002, respectively). The latter codes use Adaptive Mesh Refinement (AMR). This technique is based on a Cartesian grid which can be subdivided multiple times where more resolution is needed. Thus the use of grid codes is by no means negligible.

In section 1.1.1 we briefly describe the traditional Smoothed Particle Hydrodynamics (SPH) scheme that has been used for most particle based simulations. Then, in section 1.1.2, we will describe in detail the state-of-the-art Meshless Finite Volume scheme that has been implemented in this work.

1.1.1 SPH formalism

The SPH scheme was first developed by Lucy (1977) and Gingold and Monaghan (1977) to avoid the use of Cartesian grids. In this small description, we will use the standard formulation depicted in Springel and Hernquist (2002), which is still extensively used (a more general derivation can be found in Hopkins, 2013).

SPH defines the fluid quantities from a kernel interpolation of the states of tracer particles. The particles themselves can be thought as portions of the fluid, located at $\boldsymbol{r}_i$, moving with velocity $\boldsymbol{v}_i$ and having mass m_i. Furthermore, the thermodynamic state of the particle can be defined using its specific internal energy, u_i, or specific entropy, s_i.

Critical to the SPH formalism is the density estimate

$$\rho_i = \sum_{j=1}^{N} m_j W(\boldsymbol{r}_{ij}, h_i), \tag{1.1}$$

for a set of N particles, where $\boldsymbol{r}_{ij} \equiv \boldsymbol{r}_j - \boldsymbol{r}_i$ and W is a smoothing kernel of characteristic size h, the smoothing length. The latter is defined such that the kernel volume contains constant mass.[2]

From the Lagrangian of the set of N particles, Springel and Hernquist (2002) showed that the equations of motion are:

$$\frac{d\boldsymbol{v}_i}{dt} = -\sum_{j=1}^{N} m_j \left[f_i \frac{p_i}{\rho_i^2} \nabla_i W(\boldsymbol{r}_{ij}, h_i) + f_j \frac{p_j}{\rho_j^2} \nabla_j W(\boldsymbol{r}_{ij}, h_j) \right], \tag{1.2}$$

where p denotes the pressure and ∇_i is the derivative with respect to the coordinates of the i-th particle and

$$f_i = \left(1 + \frac{h_i}{3\rho_i} \frac{\partial \rho_i}{\partial h_i} \right)^{-1}. \tag{1.3}$$

The latter term arises from the consideration that the smoothing length is allowed to smoothly change both in time and space.

This formulation is composed of anti-symmetrical pair-wise interactions, thus it conserves energy, entropy and momentum as long as the smoothing lengths contain constant mass.

Unfortunately, in many physical scenarios discontinuities can arise and eventually shocks will develop, increasing the entropy of the fluid. As SPH is inviscid and non-diffusive by construction the entropy would remain constant along the shock, thus a wrong solution would be obtained. To correctly capture shocks, an extra term to equation 1.2 must be added. This term is the artificial viscosity. It is artificial in the sense that is prescribed ad-hoc, but in general it is representative of the micro physics that take place well below the resolved scales. The viscosity will tend to smooth out the shock over a few resolution elements (particles in this case), and can be added as,

$$\left(\frac{d\boldsymbol{v}_i}{dt} \right)_{visc} = -\sum_{j=1}^{N} m_j \Pi_{ij} \nabla_i \bar{W}_{ij}, \tag{1.4}$$

where the viscosity tensor, Π_{ij}, is definite positive and different from zero when particle i and j are approaching each other. The kernel is symmetrized, for example, as $\bar{W}_{ij} = \frac{1}{2}(W(\boldsymbol{r}_{ij}, h_i) + W(\boldsymbol{r}_{ij}, h_j))$.

The addition of this term in the equations of motion creates entropy at a rate

$$\frac{dA}{dt} = \frac{1}{2} \frac{\gamma - 1}{\rho_i^{\gamma - 1}} \sum_{j=1}^{N} m_j \Pi_{ij} \boldsymbol{v}_{ij} \cdot \nabla_i \bar{W}_{ij}, \tag{1.5}$$

where the entropic function, A, is defined as $P = A(s)\rho^\gamma$.

Despite of the exceptional conservation properties, the standard SPH scheme have important drawbacks. Most of them comes from the difficulties SPH has with handling

[2] This definition is not unique, but this one conserves energy explicitly.

discontinuities. For example, the suppression of fluid mixing (Agertz et al., 2007) and artificial transport of angular momentum (Cullen and Dehnen, 2010). Nonetheless, there have been extensive research aiming to improve the standard SPH scheme during last years (e.g. Cullen and Dehnen, 2010; Hopkins, 2013; Saitoh and Makino, 2013), and modern SPH is still broadly used in cosmological and galaxy formation codes (e.g., Barnes et al., 2017; Schaye et al., 2015; Wadsley et al., 2017, to name a few).

1.1.2 Meshless Finite Volume formalism

In recent years a new formalism has been proposed by Lanson and Vila (2008a,b) for the solution of conservation equations without relying on building a mesh. In the following, we will present a heuristic derivation of the scheme based on Gaburov and Nitadori (2011) and Hopkins (2015).

First, we look for a weak solution (LeVeque, 1992) for the system of hyperbolic partial differential equations moving in a frame with velocity $\boldsymbol{v}_{\text{frame}}$,

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot (\boldsymbol{F} + \boldsymbol{v}_{\text{frame}} \otimes \boldsymbol{U}) = \boldsymbol{S} \tag{1.6}$$

where the conserved quantities, their fluxes and the source term[3] are, for the fluid equations, respectively

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho \boldsymbol{v} \\ \rho e \end{pmatrix}; \quad \boldsymbol{F} = \begin{pmatrix} \rho \boldsymbol{v} \\ \rho \boldsymbol{v} \otimes \boldsymbol{v} + p\mathcal{I} \\ (\rho e + p)\, \boldsymbol{v} \end{pmatrix}; \quad \boldsymbol{S} = \begin{pmatrix} 0 \\ \boldsymbol{0} \\ 0 \end{pmatrix} \tag{1.7}$$

and ρ, $\boldsymbol{v}$, $e = u + \frac{1}{2}v^2$ are the mass density, fluid velocity and specific total energy. We denote the outer product as $\otimes$ and the identity matrix as $\mathcal{I}$. The density, pressure and internal energy, u, of the fluid are linked through the equation of state (EOS) $u(p, \rho)$. In this work, we will always use the ideal gas EOS,

$$u = \frac{p}{\rho(\gamma - 1)} \tag{1.8}$$

where γ is the adiabatic index. To obtain the weak solution, we multiply equation (1.6) by a differentiable test function $\phi(\boldsymbol{x}, t)$ and then integrate over the spatial ($\mathbb{R}^\nu$) and temporal domains ($\mathbb{R}^+$):

$$0 = \int_{\mathbb{R}^\nu \times \mathbb{R}^+} \phi \left[\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot (\boldsymbol{F} + \boldsymbol{v}_{frame} \otimes \boldsymbol{U}) - \boldsymbol{S} \right] d\boldsymbol{x}dt = \tag{1.9}$$

$$- \int_{\mathbb{R}^\nu \times \mathbb{R}^+} \left[\boldsymbol{U} \frac{\partial \phi}{\partial t} + \nabla\phi \cdot \boldsymbol{F} + \nabla\phi \cdot \boldsymbol{v}_{frame} \otimes \boldsymbol{U} + \phi \boldsymbol{S} \right] d\boldsymbol{x}dt = \tag{1.10}$$

$$- \int_{\mathbb{R}^\nu \times \mathbb{R}^+} \left[\boldsymbol{U} \frac{d\phi}{dt} + \nabla\phi \cdot \boldsymbol{F} + \phi S \right] d\boldsymbol{x}dt \tag{1.11}$$

[3] Although the source terms are zero for the ideal gas, we have kept them in the derivation as they can include gravity, cooling, and other effects not taken into account yet in our implementation.

For the first step, we have integrated by parts both the $\boldsymbol{U}$ and $\boldsymbol{F}$ terms, and the boundary terms arising from those integrations are zero because we assume that ϕ vanishes at the boundaries. To obtain the final weak formulation we have defined the comoving (Lagrangian) derivative of a generic function $f(\boldsymbol{x}, t)$ as $df/dt = \partial f/\partial t + \boldsymbol{v}_{\text{frame}} \cdot \nabla f$.

The spatial part of this integral is discretized in a set of N points arbitrarily distributed in $\mathbb{R}^\nu$. To that end, we use a partition of unity such as:

$$\psi_i(\boldsymbol{x}) = \omega^{-1}(\boldsymbol{x}) W(\boldsymbol{x} - \boldsymbol{x}_i, h(\boldsymbol{x})) \qquad \omega(\boldsymbol{x}) = \sum_{j=1}^{N} W(\boldsymbol{x} - \boldsymbol{x}_j, h(\boldsymbol{x})) \tag{1.12}$$

The kernel function, W, must have compact support of size h, the smoothing length, so in practice the sum is only performed over those particles that lie inside the support of W, $\mathbb{S}$. The normalization factor, ω, can be understood as an estimate of the local particle number density, if we assume that the kernel is normalized to unity such that $\int_{\mathbb{R}^\nu} W(\boldsymbol{x}, h(\boldsymbol{x}))d\boldsymbol{x} = 1$.

By definition, $1 = \sum_{i=1}^{N} \psi_i(\boldsymbol{x})$ for any $\boldsymbol{x}$. In a physical sense, we are partitioning each point of space into contributions of different particles. Those contributions are proportional to the "effective" volume of the i-th particle at $\boldsymbol{x}$. Having this in mind, we can use this to approximate the integral of an arbitrary function,

$$\int_{\mathbb{R}^\nu} f(\boldsymbol{x})d\boldsymbol{x} = \sum_{i=1}^{N} \int f(\boldsymbol{x})\psi_i(\boldsymbol{x})d\boldsymbol{x} \approx \sum_{i=1}^{N} f_i \int \psi_i(\boldsymbol{x})d\boldsymbol{x} \equiv \sum_{i=1}^{N} f_i V_i, \tag{1.13}$$

where we have adopted the notation $f_i \equiv f(\boldsymbol{x}_i)$. As we anticipated, $V_i \equiv \int \psi_i(\boldsymbol{x})d\boldsymbol{x} \approx \omega_i^{-1} + \mathcal{O}(h^2)$ is the effective volume of particle i. For the third step, we have assumed that $f(\boldsymbol{x})$ changes smoothly over the domain dominated by the contribution of the i-th particle. If desired, higher order expansions can be done for this term. However, this is enough to achieve the desired second-order accuracy. We can directly apply the discretization scheme (1.13) to the weak solution (1.11):

$$\sum_{i=1}^{N} \int_{\mathbb{R}^+} \left[V_i \boldsymbol{U}_i \left(\frac{d\phi}{dt} \right)_i + V_i \left(\nabla \phi \right)_i \cdot \boldsymbol{F}_i + \phi_i S_i \right] dt = 0 \tag{1.14}$$

For further development of this solution, we need an estimate for $\nabla\phi$. Lanson and Vila (2008a) proposed a second-order accurate meshless partial derivative:

$$(\nabla f)_i^\alpha \approx (D^\alpha f)_i \equiv \sum_{j \in \mathbb{S}_i} (f_j - f_i)\tilde{\psi}_j^\alpha(\boldsymbol{x}_i), \text{ where} \tag{1.15}$$

$$\tilde{\psi}_j^\alpha(\boldsymbol{x}_i) \equiv B_i^{\alpha\beta}(\boldsymbol{x}_j - \boldsymbol{x}_i)^\beta \psi_j(\boldsymbol{x}_i), \tag{1.16}$$

$$\bar{\bar{B}}_i \equiv \bar{\bar{E}}_i^{-1}, \text{ and} \tag{1.17}$$

$$E_i^{\alpha\beta} \equiv \sum_{j \in \mathbb{S}_i} (x_j^\beta - x_i^\beta)(x_j^\alpha - x_i^\alpha)\psi_j(\boldsymbol{x}_i). \tag{1.18}$$

For the definition of $\tilde{\psi}_j^\alpha$ we have used the Einstein summation notation, and $\mathbb{S}_i$ denotes the set of particles lying inside the support of the i-th particle, i.e., its neighbours. Inserting (1.15) into the weak solution (1.14),

$$\int_{\mathbb{R}^+} dt \sum_{i=1}^{N} \phi_i \left[-\frac{d}{dt}\left(V_i \boldsymbol{U}_i\right) - \sum_{j\in\mathbb{S}_i} \left[V_i F_i^\alpha \tilde{\psi}_j^\alpha(\boldsymbol{x}_i) - V_j F_j^\alpha \tilde{\psi}_i^\alpha(\boldsymbol{x}_j) \right] + V_i \boldsymbol{S}_i \right] = 0, \tag{1.19}$$

where we have integrated by parts the first term and rearranged the second term as[4]

$$\sum_{i=1}^{N} V_i (\nabla\phi)_i \cdot \boldsymbol{F}_i = \sum_{i=1}^{N} V_i \sum_{j\in\mathbb{S}_i} (\phi_j - \phi_i) \tilde{\psi}_j^\alpha(\boldsymbol{x}_i) F_i^\alpha = \tag{1.20}$$

$$- \sum_{i=1}^{N} \sum_{j\in\mathbb{S}_i} V_i \phi_i \tilde{\psi}_j^\alpha(\boldsymbol{x}_i) F_i^\alpha + \sum_{i=1}^{N} \sum_{j\in\mathbb{S}_i} V_j \phi_i \tilde{\psi}_i^\alpha(\boldsymbol{x}_j) F_j^\alpha = \tag{1.21}$$

$$- \sum_{i=1}^{N} \phi_i \sum_{j\in\mathbb{S}_i} \left[V_i F_i^\alpha \tilde{\psi}_j^\alpha(\boldsymbol{x}_i) - V_j F_j^\alpha \tilde{\psi}_i^\alpha(\boldsymbol{x}_j) \right]. \tag{1.22}$$

Now following the general idea of a Godunov scheme (i.e., the fluid variables are represented as piece-wise functions; see chapter 13 of LeVeque, 1992, for details), we can set $\boldsymbol{F}_i = \boldsymbol{F}_j \equiv \boldsymbol{F}_{ij}$, where the latter is the solution of a Riemann problem with left and right primitive variables, $\boldsymbol{W}_i$ and $\boldsymbol{W}_j$, respectively. Assuming this, we can define the "face" vector as $A_{ij}^\alpha = V_i \tilde{\psi}_j(\boldsymbol{x}_i) - V_j \tilde{\psi}_i(\boldsymbol{x}_j)$ such that the final form of the scheme reads:

$$\frac{d}{dt}\left(V_i \boldsymbol{U}_i\right) + \sum_{j\in\mathbb{S}_i} \boldsymbol{F}_{ij} \cdot \boldsymbol{A}_{ij} = V_i \boldsymbol{S}_i. \tag{1.23}$$

The flux projected onto the face can be computed using a standard 1D Riemann solver. We have to take into account that the face is moving with velocity $\boldsymbol{v}_{\rm frame} = \frac{1}{2}(\boldsymbol{v}_i + \boldsymbol{v}_j)$, so the final fluxes must be de-boosted. Following Hopkins (2015),

$$\boldsymbol{F}_{ij} = \boldsymbol{F}'_{ij} + \begin{pmatrix} 0 \\ \boldsymbol{v}_{\rm frame} F^\rho \\ \frac{1}{2} v_{\rm frame}^2 F^\rho + \boldsymbol{v}_{\rm frame} \cdot \boldsymbol{F}^v \end{pmatrix}, \tag{1.24}$$

where F'_{ij} is the solution of the 1D Riemann problem in the frame of reference of the moving face.

For being consistent and having a spatial second order scheme we must extrapolate the particle's states to the faces where the Riemann problem will be solved. As we already have an approximation to the derivative $(D^\alpha f)$, we can extrapolate the values in a straightforward way, as

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_i) + (D^\alpha f)(x^\alpha - x^\alpha), \tag{1.25}$$

[4]This rearrangement can be done because $\psi_i(\boldsymbol{x}_j) = 0$ if $j \notin \mathbb{S}_i$

which will be only accurate if $(\boldsymbol{x} - \boldsymbol{x}_i) \sim h_i$. Even in those cases, we can still produce non-physical states such as negative densities and/or pressures. To avoid this, we use the Barth and Jespersen (1989) limiter which only allows extrapolated values to be between the maximum and minimum values of the neighbouring particles.

In order to provide a second order scheme, we discretize the equation (1.23) as (Hopkins, 2015; Springel, 2010b)

$$\boldsymbol{Q}_i^{(n+1)} = \boldsymbol{Q}_i^{(n)} + \Delta t \left[V_i \boldsymbol{S}_i^{(n+1/2)} - \sum_{j \in \mathbb{S}_i} \boldsymbol{F}_{ij}^{(n+1/2)} \cdot \boldsymbol{A}_{ij} \right] \tag{1.26}$$

where $\boldsymbol{Q}_i \equiv V_i \boldsymbol{U}_i$ is the vector of conserved quantities and both the source term and fluxes have been extrapolated in time for half a time step. This is straightforward for an ideal gas, as we can easily convert spatial gradients to temporal ones using the Euler equations (Hopkins, 2015; Springel, 2010b),

$$\frac{\partial \boldsymbol{W}}{\partial t} = - \begin{pmatrix} \boldsymbol{v} & \rho & 0 \\ 0 & \boldsymbol{v} & \rho^{-1} \\ 0 & \gamma p & \boldsymbol{v} \end{pmatrix} \nabla \boldsymbol{W}, \tag{1.27}$$

and then extrapolate in time as $\boldsymbol{W}^{(n+1/2)} = \boldsymbol{W}^{(n)} + \frac{1}{2}\Delta t (\partial \boldsymbol{W}/\partial t)$.

In the same way as with the SPH formulation exposed in section 1.1.1, this scheme also *must* conserve energy, mass and angular momentum as long as we are sure that all the interactions are symmetric. Thus, it is of key importance in this scheme to guarantee that all interactions are done in pairs. In contrast with SPH, we are solving the Riemann problem between each particle, thus we are safely taking into account the possible discontinuities in the solution, and there is no need for a special treatment of discontinuities, such as artificial viscosity or conduction.

It must be noted that the preceding conservation properties are not expected to be satisfied to the leading order of the scheme, but to *machine accuracy*, so they make this scheme a powerful tool for particle based simulations.

1.2 The code: `PKDGRAV3`

The framework within which all the development has been carried out is the public code `PKDGRAV3` (Potter et al., 2017). The code is state-of-the-art regarding N-body simulations, having performed one of the biggest (in number of particles) simulation up to date (8 trillion particles), and having very low time-to-solution for large scale structure simulations. It can take full advantage of heterogeneous architectures as it provides support for `pthreads`, `MPI` and `CUDA`. Due to this versatility, it can efficiently make the most of the new trend on High Performance Computing (HPC) facilities: hybrid CPU+GPU nodes. Under the hood, the code is based on `PKDGRAV` (Stadel, 2001), which has been extensively used throughout recent years (for example, Diemand et al., 2004; Power et al., 2003). Furthermore, it is also the base for the hydro-SPH code `GASOLINE`

(Wadsley et al., 2004) (and its newer version GASOLINE2, Wadsley et al., 2017), that does not provide GPU support.

In the following, we will briefly describe the code `PKDGRAV3` For more details the reader is referred to Stadel (2001).

1.2.1 Code structure

`PKDGRAV3` is divided in four abstract layers[5]. This design separates the communication from the actual computations in such a way that a typical developer (e.g., someone who implements a physics module) does not need to understand in detail how the communication is done, but only how to interact with the layer associated with that task. It also provides high adaptability to system architectures, cluster topologies (i.e., how many cores are per node, and how the nodes are connected between them) and physical problems.

We now briefly explain each one of those layers:

1. The base layer, in charge of communication, task managing and scheduling is the MDL (Machine Dependent Layer). This layer is also in charge of memory management, and it creates local caches for fast accesses to already fetched remote data.

2. The master layer, MSR, is executed serially. It represents the workflow of the program. It starts reading the parameters of the simulation, and then starts to dispatch computations to the other processors via the next layer.

3. The PST (Processor Set Tree) organize all the processors in a binary tree, being the master the root node. When some computation is dispatched to the PST, each node pass the information to its leaves recursively, and then starts its own part of the computation, calling functions in the next layer.

4. The shallower layer, PKD, is the only one that has actual access to particle data. It is executed in all nodes and encloses all the physics. It is essentially serial code, and may have calls to the MDL layer to access data in remote processors.

An example workflow is shown in figure 1.1. The master process calls three different functions serially, which then call the corresponding PST functions to distribute the workload to the different available processors. Each one of them calls the PKD functions that can modify the particles' data. The naming convention for the functions is also depicted in this figure. Each function name should have a suffix indicating in which layer it is defined. For example, a function that computes the time-step would need the definition of `msrComputeTimeStep`, `pstComputeTimeStep` and `pkdComputeTimeStep`.

In a general case, the definition of the function in the Master layer is not needed, as the PST layer can be directly invoked from the `main` function, but it helps to make the code more readable. Furthermore, as we want the function to be called by all processors,

[5]Its name comes from using a k-D tree for the Particles.

all the PST functions will have very similar definitions, only changing the data structure which is sent (and/or received) to(from) other processes.

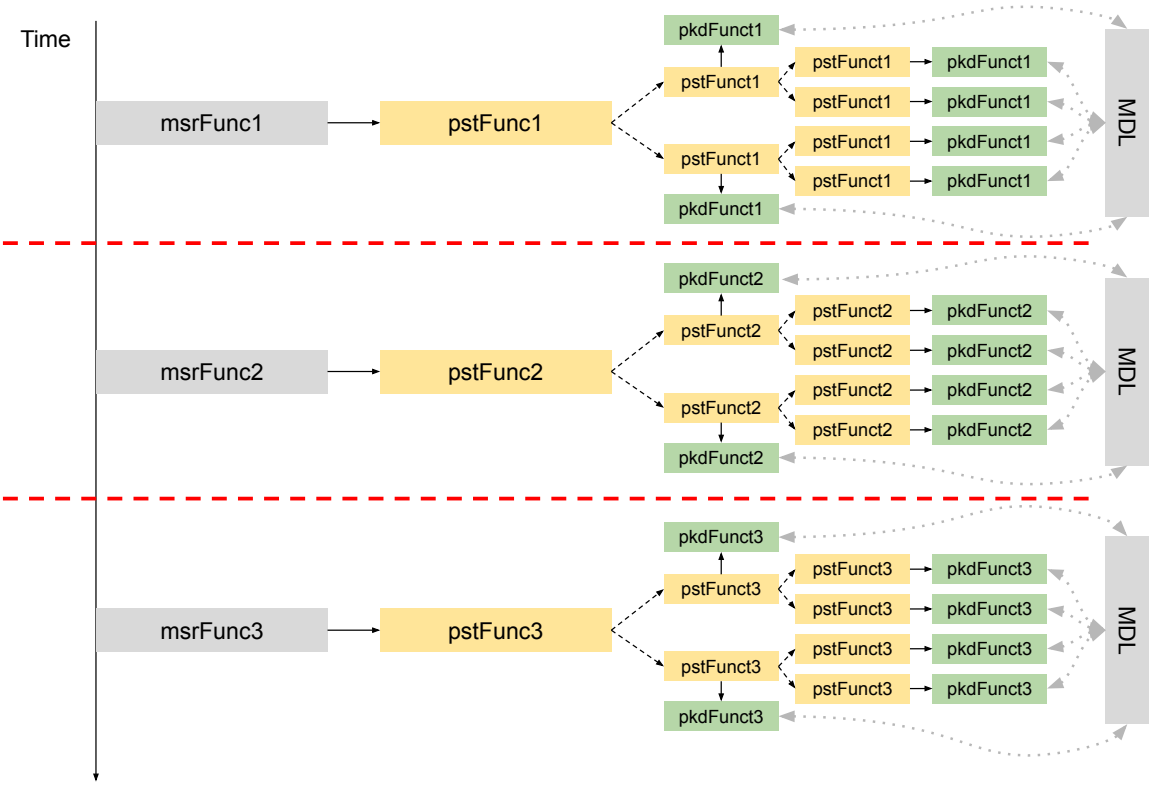

Figure 1.1: Example workflow showing how the different layers of `PKDGRAV3` interact. The master process can invoke PST functions, which then can send work to deeper nodes of the binary tree (dashed lines). Once the processor has finished sending information, it starts its own part of the work at the PKD layer. In this layer, information stored in other processors can be requested via the MDL (dotted gray lines). Although not shown, information can be distributed in both directions, so gather operations can be easily implemented using this layering scheme. After all the work has been finished, the master process call the next PST function. This creates an implicit barrier, marked as a red dashed line.

1.2.2 Gravity solver

`PKDGRAV3` relies on the Fast Multipole Method (FMM) for computing the gravitational interaction among particles. Although gravity has not been yet added to our new solver, it will be briefly described as it forms the core of `PKDGRAV`. Rigorous formulations of the FMM can be found in Greengard (1988) and Greengard and Rokhlin (1997); application to astrophysical N-body problems in Dehnen (2002), Cheng et al. (1999) and Stadel (2001); and reviews in Greengard (2013) and Beatson and Greengard (1997).

The FMM expands the notion of hierarchical tree codes based on the work of Barnes and Hut (1986), known as BH tree. They recursively decompose the domain in an oct-tree and add the contribution of far regions using their centre of mass or higher order multipole expansion. The BH algorithm scales as $\mathcal{O}(N \log N)$, as for each particle $\mathcal{O}(\log N)$ cells-particle[6] interactions are computed.

In `PKDGRAV3`, the domain is decomposed instead in a binary tree where *parent* cells are divided along the longest axis into two equal volume *child* cells. This can be easily mapped to the PST, improving load-balancing and communication efficiency. Once the particles are organized in the tree, three types of gravitational interactions can happen, depending on the distance between particles/cells:

1. **Particle-Particle**: These interactions take place when particles are close. It uses the standard force evaluation without any approximation, i.e., the force between two particles is $\boldsymbol{F} = -Gm_i m_j \boldsymbol{r}_{ij}/\boldsymbol{r}_{ij}^3$
2. **Particle-Cell**: These interactions are equivalent to those of Barnes and Hut (1986), where the contribution of a distant region is added using its multipoles.
3. **Cell-Cell**: These inherent interactions of FMM are computed for very distant cells, and consist of adding the contribution of the multipoles of a distant cell directly to a whole other cell. Then the multipoles are added to its *child* cells/particles.

Once the tree is built, the multipole expansions are computed (using all the available mathematical machinery as multipole additions and translations) for each cell, and then for each particle the interactions are added depending upon how far the source is located. Given that for very distant cells the contribution is computed as CC interactions, the total number of interactions of each particle is constant[7] and thus the FMM scales simply as $\mathcal{O}(N)$.

Due to this scaling behavior, FMM is effectively a *structure free algorithm*, as its performance does not depend directly on the tree structure (the $\mathcal{O}(\log N)$ term). This means that, in principle, the gravity computation should scale the same way at the beginning of a cosmological simulation (when the universe is highly homogeneous) and at the end, when collapsed regions (halos) have been formed (Potter et al., 2017). In contrast, codes using the BH oct-tree, generate deep trees in regions of high clustering of particles. Then, walking the tree becomes more computationally demanding and the performance can be deteriorated.

1.2.3 Time integration

`PKDGRAV` uses a standard, second-order Kick-Drift-Kick scheme for the time integration. This has proven to be very successful for cosmological simulations and orbital dynamics thanks to its symplectic nature (see Springel, 2005, for details).

[6] Cell or leaf is one of the nodes of the spatial tree that divides the domain and contains more than one particle

[7] Or at least does not directly depend on N.

The "Kick" consists of updating the particle's velocities with the acceleration, whereas the "Drift" updates only their positions. This is done with half a time step difference. Simple visualization of this algorithm is shown in figure 1.2. It is clear that only at times $t = n\Delta t$ both velocities and positions are synchronized, so that would be the ideal time to, for example, save the particle data to a file.

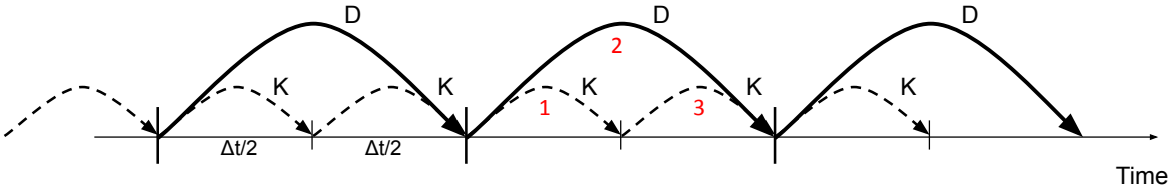

Figure 1.2: Visualization of the Kick-Drift-Kick scheme. In red, the order of the operations for a single time step. Notice that the Kick phase is divided in two just to have synchronized positions and velocities at $t = n\Delta t$. For the Drift step (2), the velocities computed at the Kick (1) are used.

This scheme can be easily modified in order to allow individual time steps for the particles. This way, more "active" particles will have their velocities updated more often, whereas less active particle will be updated less often. The allowed time steps of the particles must be $(\Delta t)_i = (\Delta t)_0/2^{k_i}$, with k_i a positive integer.[8] Then, all particles will be synchronized $N_{\rm step}$ times, at multiples of $(\Delta t)_0$, given that the latter is defined as $(\Delta t)_0 = (t_f - t_0)/N_{\rm step}$.

To obtain k_i, different time steps criteria can be used (they will be explained in section 3.2.5) to obtain a guess (Δt) for the particle. That being known, the smallest k_i that fulfill $(\Delta t)_0/2^{k_i} < (\Delta t)$ is chosen.

In general, a slight modification to the KDK scheme is done when allowing individual time steps. The "Kick" is certainly done for the "active" particles, but as the "Drift" is computationally inexpensive, it is done for all particles at all time steps. For example, the least active particle[9] instead of doing one drift of $(\Delta t)_0$, does 2^K small drifts adding always the same displacement, being $K = \max_i(k_i)$.

1.2.4 Smoothing operator

Key to both SPH and MFV algorithms is the definition of the smoothing length, h. Traditionally, there have been three approaches to compute its value:

1. The number of neighbours, $N_{\rm NGB}$, is a *fixed* parameter of the simulation. Then h is the distance to the $N_{\rm NGB}$-th nearest neighbour. This is in disuse in favor of the next approaches.

[8] In `PKDGRAV` jargon, k_i is called the *rung* of the particle

[9] We thus assume that for this particle $k_i = 0$.

2. The *approximate* number of neighbours is a parameter of the simulation, and the smoothing length is computed iteratively either fixing:

   (a) an estimation of the volume, such as $V_i = n_i^{-1} \approx \omega_i^{-1}$,

   $$N_{\mathrm{NGB}} = \frac{\frac{4\pi}{3} h_i^3}{\omega_i^{-1}}, \tag{1.28}$$

   (b) or an estimation of the density

   $$N_{\mathrm{NGB}} = \frac{\frac{4\pi}{3} \rho_i h_i^3}{\sum_{j \in \mathcal{S}_i} m_j}. \tag{1.29}$$

The first one is used in `PKDGRAV`, hence in its SPH version, GASOLINE (Wadsley et al., 2004). It is still implemented in `PKDGRAV3`, but we have found that although it correctly returns N_{NGB} neighbours, those are not necessarily the closest ones. We are not aware if this has been modified for GASOLINE. To avoid using a function that we do not think it is correctly implemented, we have discarded this approach.

The last two approaches are common in most SPH codes (e.g. Gonnet, 2014; Schaye et al., 2015; Springel, 2005). Most of those codes also assume constant mass for the particles, and in those cases (2a) and (2b) are equivalent. The difference arises when the mass of the particles can change, as in the MFV scheme. In those cases, the last approach is generally used (Gaburov and Nitadori, 2011; Hopkins, 2015) and we have implemented it as well(details to be discussed in section 3.2.4).

Chapter 2

Objectives

In the present work we aim to accomplish the following objectives:

1. To get acquainted with the Meshless Finite Volume (MFV) scheme. Not only about its mathematical structure, but also from a physical and technical point of view.

2. To get acquainted with the state-of-the-art N-body `PKDGRAV3`. To understand its underlying structure and its approach to High Performance Computing. To get used to the Fast Multipole Method for the future integration of the gravity solver with the hydrodynamic solver.

3. To implement the MFV scheme within the `PKDGRAV3` code. This must be done taking full advantage of the underlying code structure of `PKDGRAV3` and having in mind future improvements, such as porting part of the code to be executed on Graphics Processing Units (GPU's).

4. This implementation must be *unique* and *new*. Currently, there are few codes using the MFV scheme (only Gaburov and Nitadori, 2011; Hopkins, 2015; Hubber et al., 2018, with their codes named: WPMHD, GIZMO and GANDALF, respectively),[1] thus it is of key importance to provide an independent test of the scheme. To that end, all the implementation must be original, based mostly on the main equations of the scheme, rather than previously developed code.

5. To test the new implementation with a series of test cases with known solution. These tests check the correctness of the implementation, and are idealized cases of more complex configurations found in production runs.

The first and second objectives are partially discussed in the previous chapter, namely in sections 1.1.2 and 1.2, respectively. More importantly, their accomplishment is the base for the third and fourth objectives. How these were achieved will be discussed in the next chapter. The last objective will be described in chapter 4.

[1] Actually, both GIZMO and GANDALF typically use the Meshless Finite Mass, a variant of MFV with fixed mass for the particles.

Chapter 3

Code development

The scheme described in section 1.1.2 has been implemented into the N-body code PKDGRAV3 (section 1.2). We have maintained the inherent code structure, taking full advantage of all the already implemented functions, such as the smoothing operator, or the Kick-Drift-Kick time integration. Doing so, we can deploy our code in desktop computers or clusters achieving good scaling.

In section 3.1, the overall workflow of the code will be described. Then, in section 3.2, the implementation of the Meshless Finite Volume method introduced in section 1.1.2 will be laid out. Remarks about performance and other minor modifications to the code will be done in section 3.3 and section 3.4, respectively.

3.1 Workflow

The workflow of the master process can be summarized in figure 3.1. Although PKDGRAV3 is able to use individual time steps as described in section 1.2.3, in this diagram we show a simplified workflow assuming that all particles have the same time step. The phases labeled in red are those newly implemented in this work, while the green ones were already at PKDGRAV3 but have been modified. To directly check the densities that have been computed for the initial conditions, a new call for creating an snapshot have been included just before the main loop starts, shown in blue. The workflow can be divided in three phases:

1. Setup of the simulation and the processes involved. In this phase, density, primitive variables, gradients and fluxes are computed, but the conserved variables of the particles are not modified. Due to this initialization, we can safely compute the first time step (details in section 3.2.5). This, together with creating a snapshot just at the end of the setup phase, has been proven very useful for debugging the initial conditions, as we can check that the computed density is the same that we expect.[1]

[1] As density depends on the geometry and number of neighbours, it can not be set as a normal variable at the initial conditions (such as, e.g., the velocity).

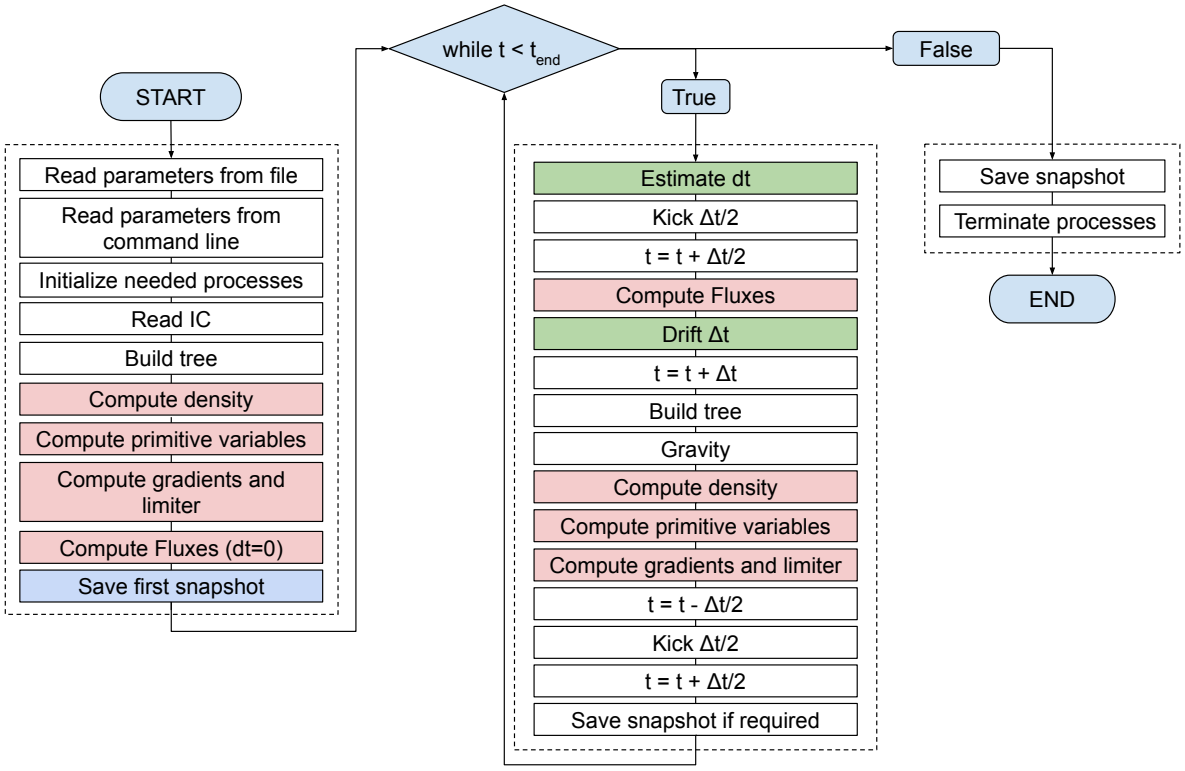

Figure 3.1: Example workflow where all the particles have the same time step. In red, those new functions implemented in this work. In green, those that has been modified with respect to the base `PKDGRAV3`.

2. Main loop, which starts with a "Kick" to the particles. In this work it will not be relevant as this only involves collisionless particles. Next, the hydrodynamic fluxes will be computed, and their contributions will be added to the conserved quantities. Then, all the particles are drifted to their new positions, according to their velocity, which was last updated when "Compute primitive variables" was called.

   Knowing the new positions, the tree is built (which is needed for the neighbours finding algorithm) and the densities are updated. After that, the primitive variables are computed from the conserved variables, and the gradients are calculated. If needed, they are limited for stability of the solution. Those gradients and primitive variables are the ones to be used the next iteration.

3. After the desired time is reached, the results are dumped to a final snapshot, and all the worker processes (i.e., those invoked by the PST layer) are finalized.

It must be stated that unless specified, all the implementation carried out and explained throughout this section is *new*, in other words, there has been no "copy-pasting" from previous existing works (to accomplish the fourth objective of section 2). We find this highly relevant as there is little amount of published work regarding this numerical scheme

(only Gaburov and Nitadori, 2011; Hopkins, 2015; Hubber et al., 2018; Lanson and Vila, 2008a, and other works using the same codes, with minor modifications). Thus it is interesting to *independently* check that the scheme works as expected and reported.

3.2 Hydrodynamics solver

In this section a thorough technical overview of the implementation is presented. We have divided this section in the description of: the particle variables that has been added to the code (section 3.2.1), the so-called neighbours loops (section 3.2.2), the Riemann solver (section 3.2.3), the iterative computation of the smoothing length (section 3.2.4) and the different time step criteria implemented (section 3.2.5).

3.2.1 Memory footprint

`PKDGRAV3` save the particle information creating structures on-the-fly, saving the offset of each variable, and then allocating the hole chunk of memory needed at once. Compared with using the standard C `struct`, this increases memory efficiency as no padding bytes are added.

Furthermore, this approach is very versatile, as for example we can choose not to save the potential of the particle without modifying the code, by just using the appropriate run-time flag. Nevertheless, this is more complex than using a C `struct`, and was not used for our implementation due to time limitations (it can be changed in the future).

For the hydrodynamics, an old `struct` of an unfinished SPH implementation was reused, as well as all the helper functions associated with it. As today, the `struct` holding all this information contains: the old SPH variables (which can be removed); the $\bar{\bar{B}}$ matrix (equation (1.17)); the time at which the last update of the primitive variables was performed; the approximated local number density, ω; the primitive variables, p, (both density and velocity are already in the native `PKDGRAV3` particle data); their gradients; the fluxes of the conserved variables and the conserved variables ($m\boldsymbol{v}$, me).

Thus this implementation requires to store 33 doubles (i.e., 264 bytes[2]) per particle. We have chosen to use double precision, but in the future the use of single precision can be studied for some of these variables to reduce the memory footprint.

3.2.2 Neighbours loops

The majority of the new code is included in the four phases denoted "Compute Fluxes", "Compute density", "Compute gradients and limiter" and "Compute primitive variables". In each one of these, there is a main loop over all the gas particles, and for the first three, also a nested loop over the neighbours of the i-th particle. Before the start of the inner loop, the neighbours of the i-th particle are identified. If needed, the information is fetched from other processors. For the last one, there is no need to look for the neighbours as all the operations involve only i-th particle's variables.

[2]Without accounting for padding bytes.

We now describe each of the neighbours loops:

- **Compute density**: this loop is in charge of computing the local number density, ω_i, and assuming that $V_i \approx \omega_i^{-1}$ (Gaburov and Nitadori, 2011; Hopkins, 2015), it computes the density of the particle. In this neighbour loop, the smoothing length of the particle is also updated (details at section 3.2.4). This must be called after the particles are drifted.

- **Compute gradients and limiter**: this loop computes first the gradients using the partial derivative approximation by Lanson and Vila (2008a), equation (1.15). Then, it gathers the maximum and minimum values of each variable across its neighbours. Following the standard limiter of Barth and Jespersen (1989), the gradients are multiplied by a factor $\alpha \in [0, 1]$ such that the extrapolated variables inside the support space are not above (below) the maximum (minimum) value of the neighbours.

- **Compute Fluxes**: this is the core of the implementation. It starts extrapolating to the faces ($\boldsymbol{A}_{ij}$ in equation (1.23)) the boosted primitive variables using equation (1.25), and then do a temporal extrapolation following equation (1.27). Once the states are computed, they are passed to the Riemann solver (section 3.2.3). Afterward, the fluxes are de-boosted from the face system of reference using equation (1.24) and added to the conserved variables of the i-th particle using the smallest of the time steps between the i-th particle and its neighbour.

Additionally, a neighbour loop is done when estimating Δt, which computes the maximum signal velocity of a given particle with respect to its neighbours. This will be discussed in more detail in section 3.2.5.

3.2.3 The Riemann solver

The Riemann solver is the core of almost all Godunov schemes. Because of this, there are plenty of methods and codes to solve the Riemann problem. To reduce the development time, we have used the Riemann solver implemented within GIZMO (Hopkins, 2015), which is heavily based on that of AREPO (Springel, 2010b). Some minor modifications where required to match the input and output structures, as well as reading parameters (such as γ) from the core of `PKDGRAV3`.

From the GIZMO code, we have kept two Riemann solvers:[3] an exact, iterative, one; and an approximate one (HLLC). We will not describe them further in this work, and the reader is referred to Toro (2009) for more information (the HLLC solver is described in chapter 10 and the exact solution developed in chapter 4).

[3] Actually, we kept all of them but the wrapper function now only calls those two.

3.2.4 Iterative computation of h

The actual value of the smoothing length is not crucial for the convergence of the MFV scheme as long as it has a continuous first derivative. However, h to some extent represent the size of each particle, and thus the resolution of the simulation. Because of this, it is logical to define it in such a way that the "size" of a particle is smaller(bigger) where the number density of particles is higher (lower). This correspond to the approach (2a) explained in section 1.2.4.

This implies solving N implicit equations,[4]

$$(2h_i)^3 = \frac{3N_{\mathrm{NGB}}}{4\pi n_i}, \tag{3.1}$$

where $n_i \approx \omega_i$. For the iterative procedure to work, two parameters have to be defined: the desired N_{NGB}, and ΔN_{NGB} which is the allowed deviation from this expected value. The latter is usually set equal to a fraction of unity, which gives accurate results without expending a lot of time in the iterative solver.

The iterative procedure consist on a hybrid approach between a fixed-point iteration and a bisection algorithm. This approach have been found consistent for all the cases studied in this work.

Although iterative methods are avoided due to their impact in performance, in our case we have found that the h computation always takes less time than the flux computation. Furthermore, as the smoothing length is only updated for active particles, typically the neighbour search is only performed for a small set of particles.

3.2.5 Time step

In all explicit schemes (such as equation 1.26), the time step should be limited in such a way that information can not travel more than one resolution element each iteration. This simple imposition leads to the condition known as the Courant-Friedrichs-Levy (CFL) criteria:

$$\Delta t < \frac{\Delta x}{c}, \tag{3.2}$$

where Δx is the spacing between points and c is the speed at which information moves (in general, the speed of sound). However this criteria was devised for a static distribution of particles and thus must be revisited for schemes which have moving particles. We follow the criteria of Hopkins (2015):

$$(\Delta t)_{\mathrm{CFL},i} = 2C_{\mathrm{CFL}} \frac{h_i}{|\max_j[c_{s,i} + c_{s,j} - \min(0, v_{ij}]|}. \tag{3.3}$$

In this criteria, both sound speeds ($c_{s,i}$ and $c_{s,j}$) are taken into account, but also the velocity at which the particles are approximating each other, $v_{ij} = (\boldsymbol{v}_i - \boldsymbol{v}_j) \cdot \boldsymbol{r}_{ij}/|\boldsymbol{r}_{ij}|$.

[4] In our implementation, the radius of the sphere where the neighbour search is carried out is $2h$, because our kernel is zero when $h \geq 2$

Throughout this work we have used $C_{\rm CFL} = 0.2$ unless otherwise stated.

Besides this criteria, we also have an acceleration criteria (Durier and Dalla Vecchia, 2012) expressed as:

$$(\Delta t)_{{\rm acc},i} = C_{\rm acc}\sqrt{\frac{2h_i}{a_i}}, \tag{3.4}$$

where $C_{\rm acc} \sim 10^{-3}$. From these criteria, the one that provides a tighter constraint to the time step is chosen. Furthermore, we have implemented as well a limiter following Saitoh and Makino (2009). This limiter constraint the time step of a particle to be at most 2^p times that of the most active particle surrounding it. Typically $p = 2, 3$ is a safe value to avoid that very active particles can overtake less active ones when energy is suddenly injected to the system from, for example, supernovae feedback.

When particles with different time steps are interacting, the flux is added to the conserved quantities using $\min[(\Delta t)_j, (\Delta t)_i]$. This idea was first employed in the AREPO code Springel (2010b) to have explicit conservation even with particles with individual time steps.

3.3 Remarks about performance

The code that has been described throughout this chapter is aimed to High Performance Computing (HPC). It has been thought as the base for a GPU-enabled implementation of the MFV scheme, where the flux computation would be carried out on GPU. Furthermore, as it takes advantage of the underlying MDL layer (see section 1.2.1), it is able to run on laptops or high-end clusters without modifications.

However, due to the limited time to carry out this work, the current CPU version is not fully optimized. Not only for the optimal time-to-solution, but also memory-wise. These two points will be addressed in future versions of the code, before porting it to GPU.

3.4 Other modifications to PKDGRAV3

Together with all the developments introduced in previous sections, there are minor modifications to the code which are not unique to the MFV scheme, namely:

- Generation of a first snapshot before the first step. This is useful in order to check whether the initial conditions have been read successfully.
- Implementation of a gather/scatter pair to set the time step of all particles to the minimum Δt in the whole domain, even when running in parallel. This was a needed tool for debugging the implementation, and can be used to check the correctness of the multiple time stepping scheme (section 1.2.3) when applied to the hydrodynamic solver.

Chapter 4

Results

In this section, the results obtained with the newly developed solver will be laid out. In the first place, the solver will be tested using a set of problems with known solution (section 4.1). They will ensure the correctness of the code in a wide variety of problems, some of them with strong astrophysical implications. Secondly, a brief analysis of the performance will be carried out in section 4.2.

4.1 Test cases

In the following, a set of test cases will be performed. In section 4.1.1 a simple traveling sound wave will be solved and checked against its analytical solution. This test can also give an idea about the convergence of the solver under different parameters, such as N_{NGB} and N. Then, in section 4.1.2, the core of the solver will be tested solving a typical Riemann problem. Next, more astrophysically relevant cases will be studied. In section 4.1.3, the ability to conserve the angular momentum will be tested, as well as to check if the solver artificially transports angular momentum. Afterwards, the Sedov-Taylor's explosion will be simulated, where the ability to correctly capture strong shocks will be tested (section 4.1.4). To finalize, in section 4.1.5 a fluid-mixing test will be performed, aimed to check the adaptive resolution of the scheme.

4.1.1 Sound waves

One of the most used problems to check the correctness and convergence properties of a given hydrodynamic solver is the propagation of sound waves in a isothermal medium at rest. If the amplitude of the wave is small enough, the fluid's equations can be linearised and an analytical solution can be obtained. This solution can be expressed, for a periodic

one dimensional domain of size L, as

$$\varphi = \frac{2\pi x}{L} N_w - \omega t, \tag{4.1}$$

$$\rho = \rho_0 \left(1 + A \cos(\varphi)\right), \tag{4.2}$$

$$p = p_0 \left(1 + A\ \gamma \cos(\varphi)\right) \text{ and} \tag{4.3}$$

$$v = A\ c_s \cos(\varphi), \tag{4.4}$$

where $A \ll 1$ is the amplitude of the perturbation, and N_w the number of wavelengths that fit into the domain. The whole solution moves as time advances at the speed of sound, c_s, without changing its shape.

This test is ideal for checking the spatial order of a hydrodynamical solver. The MFV scheme is second order in space, thus we expect the difference between the numerical solution and the analytical one (measured with the L_1 norm) to decrease as N^{-2}, with N the number of particles in the x-axis

For these tests we initialize a Cartesian grid with N nodes along the x-axis and we place a particle in each node. The size of the domain along the x-axis is set to unity. For the y and z axes, we set them in such a way that they contain N_2 nodes, and with the same separation among them than in the x-axis. The number of nodes in these axes is of no importance if the sound wave travels along the x-axis and thus can be taken small and constant without affecting the convergence of the method.[1]

We set for this cases $\rho_0 = 1$, $p_0 = 1$, $\gamma = 1.4$, $A = 10^{-3}$, $C_{\rm CFL} = 0.05$, $N_{\rm NGB} = 32$ and $\Delta N_{\rm NGB} = 0.05$. We run a set of simulations with $N = 16, 32, 64, 128$ with fixed $N_2 = 4$ for one crossing time.[2] In figure 4.1 the density perturbation is shown for all N (colored points), and also the expected analytical solution (black line).

It can be seen that the limiter flattens the peaks because it does not allow values of density (and other variables) to be higher (lower) than the maximum (minimum) value among its neighbours. This behavior could degrade the spatial order of the method and is more evident when using a higher $N_{\rm NGB}$, because the affected particles will be further away from the maximum (or minimum).

To check this hypothesis, we also run the same set of simulations, this time turning the limiter off. As there are no discontinuities, it is not explicitly needed and can be safely disabled. We also re-run the simulations with different $N_{\rm NGB}$. Then, for all the simulations we compute the L_1 norm of the density at one crossing time.

The convergence results are shown in figure 4.2. To guide the eye, we also show the theoretical convergence of a second order scheme ($L_1 \propto N^2$) and for a less accurate $L_1 \propto N^{-1.5}$ scheme. When the limiter is turned off, the convergence of the scheme reaches $L_1 \propto N^{-1.9}$, very close to the expected second order accuracy. Also, this convergence is not affected by the effective number of neighbours.

Comparing with convergence properties of GIZMO (figure 2 in Hopkins, 2015), we

[1] Even in this adapted cases we allow for changes in the variables along the z-direction, as well as $v_z \neq 0$. If the scheme behaves as expected, those changes will be negligible.

[2] This is the time that it takes the wave to travel the whole domain, i.e., L/c_s.

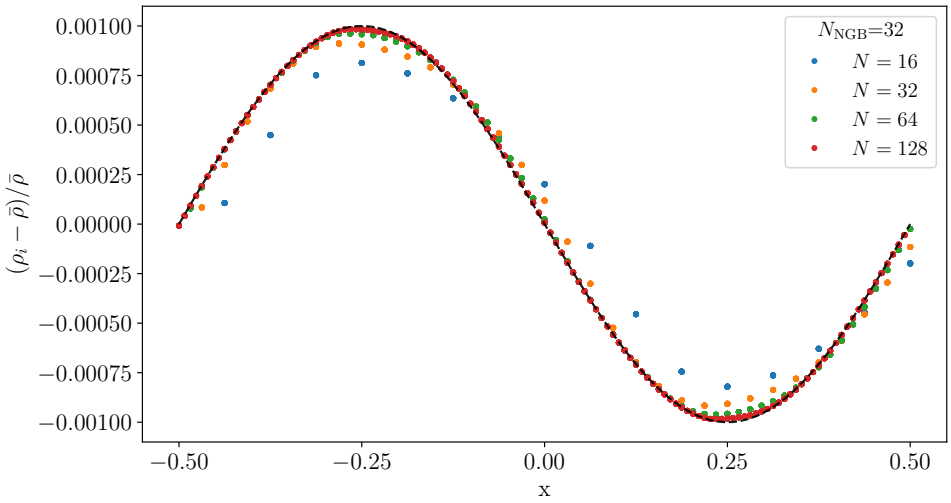

Figure 4.1: Density perturbation of the sound wave test case (section 4.1.1) after one crossing time. Marked as dots, the numerical solution of the problem with different resolution (and constant $N_{\rm NGB} = 32$). The expected solution is shown in black lines (equation 4.2). Notice that although we are solving the problem in a 3D domain, all the points with the same x coordinate have the same density, thus the symmetry of the problem is well conserved.

found slightly smaller values. They report almost perfect convergence ($L_1 \propto N^{-2}$), independently of the choice of $N_{\rm NGB}$. The difference (albeit small) can be attributed to the fact they they run the simulation in a strict 1D setup, while in our case we solve a 3D configuration.

On the other hand, when using the limiter, the convergence ratio is degraded down to ~ 1.5. Furthermore, we found that when using $N_{\rm NGB} = 32$ it increases up to 1.6 because less particles are affected by the limiter.

Not using the limiter is a risky decision, as negatives densities or pressures can appear. Therefore, in the following all simulations will be carried out with the limiter turned on. We are aware of the convergence problem, and although it is not critical, the limiter can be further improved to avoid this behavior.[3] In the following we will use, in general, $N_{\rm NGB} = 32$ because we have found in this number a good compromise between an accurate derivative approximation and over limiting.

4.1.2 The Riemann problem

Another extensively used test case consist of solving the Riemann problem, i.e., the time evolution of two regions with different states ($\boldsymbol{W}_L$, $\boldsymbol{W}_R$) joined by a discontinuity at $t = 0$ (Toro, 2009). In general, at $t > 0$, four different regions can appear, each one having its own density, pressure and velocity. Between them, either a shock wave, contact

[3]We are not aware if for GIZMO this was the approach, or if they turned off the limiter in the convergence test.

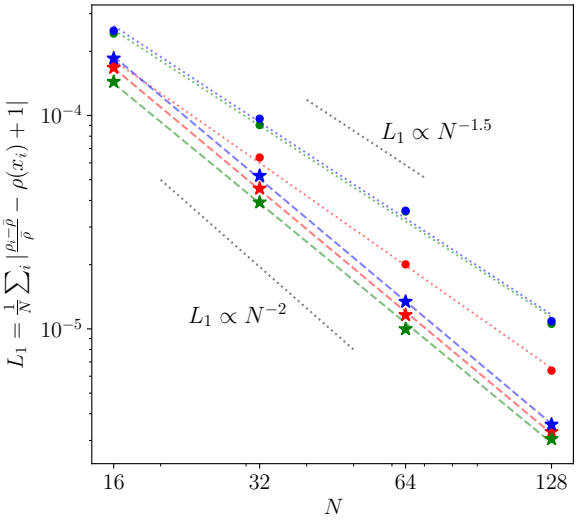

Figure 4.2: Convergence of the MFV scheme implemented in this work for the sound waves test cases (section 4.1.1). The dots represent the L_1 norm computed in the case where the limiter is active. For the star symbols the limiter was de-activated. To guide the eye, two theoretical convergence ratios were plotted. The color encodes the number of neighbours in each run: $N_{\mathrm{NGB}} = 32, 48, 64$ are red, green and blue respectively.

discontinuity or a rarefaction wave can appear. The core of the developed scheme consist in solving the Riemann problem between particles, hence we expect that the scheme will successfully pass this test.

The Riemann problem is defined in 1D. Our code, however, can only be run in three dimensions, thus we have slightly adapted the problem by expanding the one dimensional formulation maintaining constant values along the y and z axes, as in the previous test case (section 4.1.1).

We have set for this problem the following states:

$$\boldsymbol{W}_L = \begin{pmatrix} \rho_L \\ \boldsymbol{v}_L \\ p_L \end{pmatrix} = \begin{pmatrix} 1 \\ \mathbf{0} \\ 1 \end{pmatrix}; \qquad \boldsymbol{W}_R = \begin{pmatrix} \rho_R \\ \boldsymbol{v}_R \\ p_R \end{pmatrix} = \begin{pmatrix} 0.25 \\ \mathbf{0} \\ 0.1795 \end{pmatrix}. \tag{4.5}$$

and the discontinuity is placed at $x = 0$. This initial configuration will develop producing the three possible transitions between fluid states. From left to right, a rarefaction wave, a contact discontinuity and a shock wave will appear.

Due to the nature of the solver described in this work, in which the density is defined using both the mass of a particle and the local number density, there are multiple ways to set the particle's positions and masses resulting in a density obeying equation (4.5). For this case, we have studied two of them:[4]

[4]It is important to remark that regardless of the initial conditions (ICs) used, the mass of each particle is allowed to change freely during the integration.

- All particles have **constant mass** (CM) at $t = 0$. The distance between particles determines the density.
- All particles are equally separated. To have a density consistent with equation (4.5), we use **variable masses** (VM).

We have set for this case $C_{\rm CFL} = 0.2$, $N_{\rm NGB} = 64$, $\Delta N_{\rm NGB} = 0.05$, $\gamma = 1.4$, and $N = 200$ particles along the x-axis. We have performed tests with smaller number of neighbours, but we have found that unwanted oscillatory solutions appear. This may happen because the MFV is supposed to work as long as the smoothing length, h, is a continuous function. However, in these cases we can have a discontinuity in the local number density, which is translated into a discontinuity in h. If the number of neighbours is increased, the changes both in the local number density and h would be smooth enough that the MFV can successfully converge to the correct solution.

In figure 4.3, the solutions for both initial conditions are shown and compared against the analytical solution[5] at $t = 0.13$. Although with both IC the expected result is achieved, there are small differences between both runs.

Namely, the contact discontinuity (located at $x = 0$) is better resolved with the CM initial conditions. For the VM, a bump can be seen in the pressure, where it should be constant. This behavior was also observed in the GIZMO code, and is attributed to the limiter (Hopkins, 2015). At the shock wave ($x \sim 0.2$), the CM presents oscillations, which are likely caused by the choice of limiter. This is also present in GIZMO.

To better understand these differences, we show in the lower right panel of figure 4.3 the smoothing length of the particles. In this scheme, if h increases, the effective spatial resolution is lower. This becomes evident, for example, in the rarefaction wave ($x < 0$), where the run using the CM initial conditions is clearly closer to the analytical solution because the smoothing length of the particles is smaller. On the other hand, at the shock wave, the CM has worse resolution and thus the shock is smoothed over a higher number of particles.

In summary, both IC successfully converge to the expected solution with minor discrepancies. Due to its simplicity, we will take the VM approach in the following problems, but we expect no major difference if the initial conditions were set using equal masses for all particles.

4.1.3 The Gresho Vortex

For cosmological simulations it is of fundamental importance that the angular momentum is conserved, and that it is not transported by artificial viscous forces. The transport of angular momentum towards the inner(outer) parts of a galaxy would result in a decrease(increase) of the galaxy size, which is something well-constrained by observations.

One simple 2D test that allow to check whether angular momentum is being conserved or transported is the set up of triangular vortex embedded in a non-rotating background

[5] We have used the solution given by https://gitlab.com/fantaz/simple_shock_tube_calculator/tree/master/

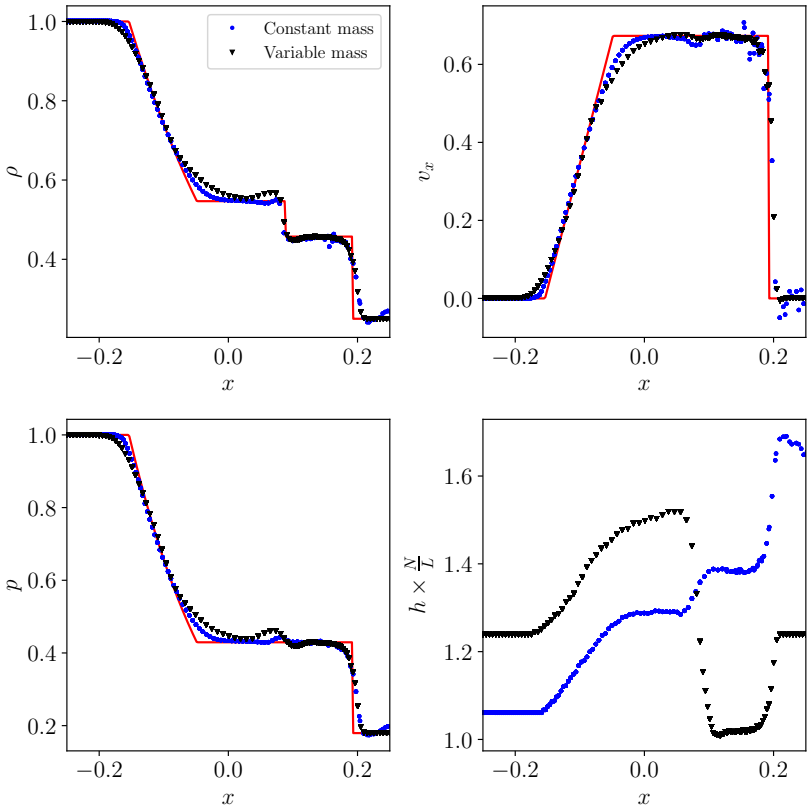

Figure 4.3: Solution of the Riemann problem (section 4.1.2) at $t = 0.13$, with the analytical solution shown in red. Two different initial conditions have been used: *Blue*, all particles started with the same mass; *black*, all particles were equally spaced and thus mass was not constant in the domain. *Top left*, density profile; *top right*, velocity along the x direction; *bottom left*, pressure profile; and *bottom right*, the smoothing length. Notice how the latter is substantially different in both runs. A higher value implicates a lower resolution in the area.

(Gresho and Chan, 1990). The inner region rotates as a rigid body up to a radius $R_1 = 0.2$, after that point, tangential velocity decreases linearly down to zero, at $R_2 = 0.4$. Assuming that the fluid has constant density, a pressure profile can be obtained such that this configuration is stationary. In summary, the initial conditions are:

$$v_\varphi = \begin{cases} 5r & r \leq 0.2 \\ 2 - 5r & 0.2 \leq r \leq 0.4 \\ 0 & 0.4 \leq r \end{cases} \quad ; \quad p = \begin{cases} 5 + \frac{25}{2} r^2 & r \leq 0.2 \\ 9 + \frac{25}{2} r^2 - 20r + 4\log(5r) & 0.2 \leq r \leq 0.4 \\ 3 + 4\log 2 & 0.4 \leq r \end{cases} \tag{4.6}$$

The size of the domain is $L = 1$ and we set 64^2 particles in the nodes of a Cartesian grid to provide $\rho = 1$. We extend the case to 3D using the methodology explained in previous cases. To test the Galilean invariance (i.e., invariant against rotations and translations) of the implemented scheme, we add a bulk velocity to all particles, $v_x = 1$. We run this

simulation with $C_{\mathrm{CFL}} = 0.2$ and $N_{\mathrm{NGB}} = 32$ ($\Delta N_{\mathrm{NGB}} = 0.05$) until $t = 3$, when the inner region has completed ~ 2.4 orbits.

In figure 4.4 we show the solution at the end of the simulation.[6] For comparison, we plot the expected solution (equation (4.6)) in red. Even when there is a bulk velocity the cylindrical symmetry is well conserved, as expected due to the Lagrangian nature of the method. Furthermore, the peak of v_φ is not displaced outwards or inwards, thus there is no noticeable artificial transport of angular momentum. In the inner region, the particles correctly rotate as a rigid body with almost no disturbance.

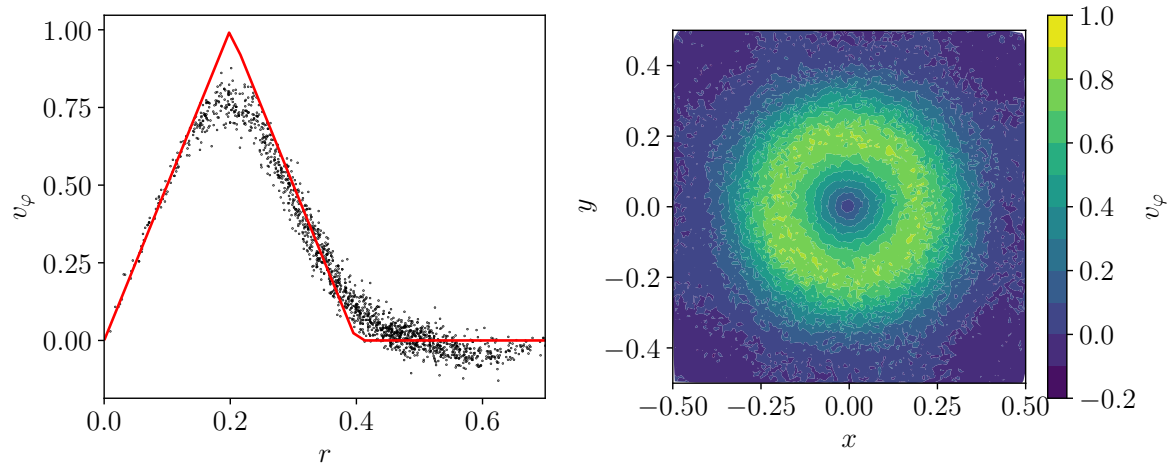

Figure 4.4: Solution of the Gresho vortex (section 4.1.3) at $t = 3$, when the inner region has completed more than two orbits. *Left*, tangential velocity profile, with the stationary solution showed as a red line. For clarity, only a small random subset of points is shown. *Right*, tangential velocity in the x-y plane. The cylindrical symmetry is well conserved as there are no preferred axes.

Comparing this results with those obtained with GIZMO (figure 4 in Hopkins, 2015), the main difference is that in our code the tangential velocity's peak is at $v_\varphi \approx 0.75$, and in their simulations it was at $v_\varphi \approx 0.85$. We found this difference not critical, as in our run we use less favorable initial conditions: whereas they started from a distribution of particles in concentric circles, we placed the particles in a Cartesian grid. Furthermore, they can solve this problem strictly in 2D, and that may enhance the convergence properties of the scheme.

4.1.4 Sedov's explosion

In cosmological simulations the feedback induced by supernovae is key to the evolution of galaxies: if too strong, gas is ejected outside of the galaxy and no more stars are formed until it collapses again; on the other hand, if too weak, too much gas collapses and too

[6] After subtracting the bulk velocity

many star forming galaxies are created, compared with observational constraints (Schaye et al., 2010).

Technically, the feedback is implemented as a sudden increase of thermal (or kinetic) energy into one (or few) particles. This causes a strong shock that expands radially outwards, leaving behind a bubble of low density, hot gas.

Hence checking the correctness of the implemented scheme when strong shocks are present is key if we plan to use it for cosmological simulations with feedback. One of the standard tests for the hydrodynamic solvers implementing feedback is the Sedov explosion (Sedov, 1993). This test starts with a homogeneous medium at rest. Then, for one (or a few) particle(s), the internal energy is suddenly increased by a few orders of magnitude with respect to the background.

A self-similar solution can be found for this case using dimensional analysis and assuming that the background energy is negligible compared to the "explosion" energy.[7] The solution gives that the shock position, after the injection of an energy E_0, is placed at

$$r_s(t) = \beta \left[\frac{E_0 t^2}{\rho_0} \right]^{1/5}, \tag{4.7}$$

and thus the shock expands with a velocity $v_s \propto t^{-3/5}$. The constant β is of order unity and depends on the adiabatic index. In our case, $\gamma = 5/3$ and $\beta = 1.1527$. For the simulations done in this section, we have set $E_0 = 1$ and $\rho_0 = 1$. To provide a background with negligible internal energy, we set the particles temperature such that (without taking into account the exploding particle) $u_b = \sum_i u_i = 10^{-5}$.

The expected density contrast between the background and the shock front is

$$\frac{\rho_s}{\rho_0} = \frac{\gamma + 1}{\gamma - 1}, \tag{4.8}$$

and is equal to 4 if we use $\gamma = 5/3$.

We have set a box of $L = 1$ with 64^3 particles arranged in a Cartesian grid. After setting the total energy of the background, we add to the central particle all the energy of the explosion.

We set $C_{\rm CFL} = 0.2$, $N_{\rm NGB} = 32$ and $\Delta N_{\rm NGB} = 0.05$. To directly compare this test with GIZMO, we set the exact same initial conditions and parameters described above.

We run both simulation until $t = 0.1$. The solution of both codes at $t = 0.07$ is shown in figure 4.5. The self-similar solution is shown as a red line. We have also marked the maximum density in the shock front for both runs.

With our code, we do not achieve the density contrast of GIZMO ($\rho_s = 3.52$ vs. $\rho_s = 3.15$), but is closer to that reported by Hubber et al. (2018) with GANDALF (although their test is in strict 2D). However, we find substantially less dispersion in the radial velocity in the inner region. The latter shares the same explanation with the absence of particles with $m \leq 0.19 \times 10^{-5}$ in the GIZMO run: that code can merge/split particles in certain situations to improve resolution and avoid close gravitational interactions between

[7] Indeed, the solution was firstly designed for estimating the energy of nuclear explosions from images.

particles with very different masses. The merging routine is triggered when the mass of a particle drops below $m_0 = 0.5 \min_i[m_i(t=0)]$, and merges that particle with its lightest neighbour. This effectively reduce the number of particles in the inner region, lowering the spatial resolution.

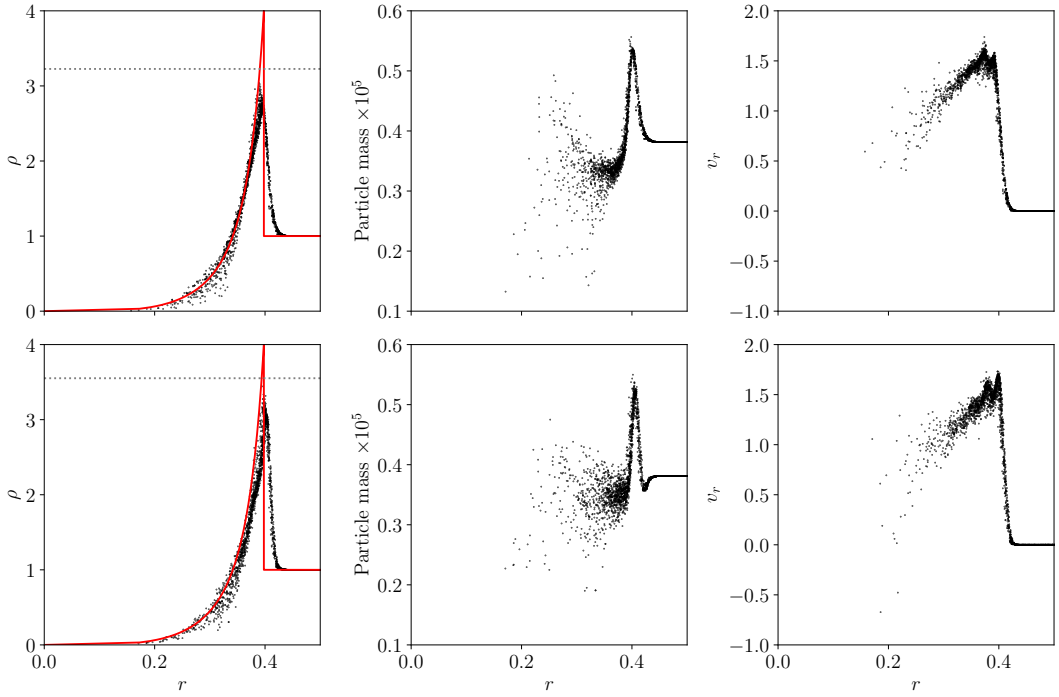

Figure 4.5: Solution at $t = 0.07$ of the Sedov explosion (section 4.1.4) for the density (*left*), particle mass (*middle*) and radial velocity (*right*). *Upper row*, results obtained with the MFV scheme developed within this work. *Bottom row*, results with the same initial conditions but solved with the MFV scheme in GIZMO. For clarity, only a small random subset of points is shown. In red, the self-similar solution.

We also show in figure 4.6 the temperature 2D distribution in a thin slice $|z| < 1/32$ for both runs. We have also marked the particle position as black dots. Our solution (left), shows better symmetry in the central region because we do not merge particles. Furthermore, GIZMO's solution has a temperature dip just before the shock (white regions), where the particle's internal energy almost drop to zero. We do not have this dip and therefore our solution is smooth.

One of the premises of the work presented at Hopkins (2015) was that the method was fully conservative, up to machine precision, provided that all particle interactions are done in pairs. We have been careful during the development and testing of our code to make sure that this condition holds, as the benefits are of great importance. Although we have checked mass and energy conservation in all our test, we only show for this test the time evolution of energy and mass, as we think this is the most challenging case. The time evolution of total mass, kinetic, internal and total energy are shown in figure 4.7.

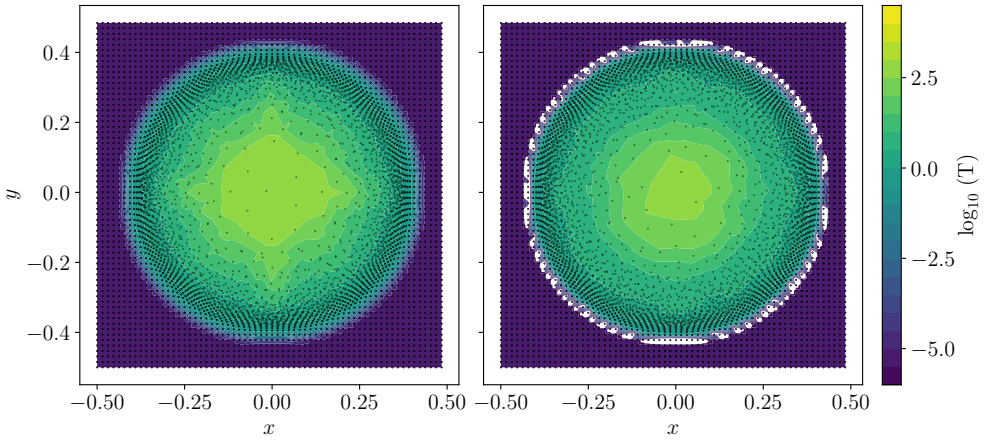

Figure 4.6: Temperature and particle distribution for a thin slice ($|z| < 1/32$) of the solution of the Sedov explosion at $t = 0.7$. *Right*, solution from the solver implemented in this work *Left*, solution using the same initial conditions but solved with the MFV scheme in GIZMO. The particle's position is represented by the black dots.

We have added for comparison the evolution of the same variables using GIZMO.

It is clear that GIZMO[8] does not conserve energy nor mass. The increase of total energy is of $\sim 5\%$, which is a noticeable amount. The mass is better conserved, with losses $\leq 0.1\%$. Comparing with GANDALF, their result is between our code and GIZMO, with a reported relative energy loss of 10^{-4}. They attribute this energy loss to the time limiter of Saitoh and Makino (2009) (see section 3.2.5), of which we have implemented a modified version.

On the other hand, our code does *conserve mass and energy up to machine accuracy*. This is an improvement of orders of magnitude with respect to GIZMO. We are not aware if this a bug, or "intentional" up to some extent.[9] Most likely the limiter or the merger/split algorithms are not fully conservatives. But if that is the case, it is not explicitly stated in Hopkins (2015).

4.1.5 Kelvin-Helmholtz instability

The last test is aimed at studying the adaptive spatial resolution and fluid mixing properties of the solver.

For cosmological simulations, fluid mixing is needed to correctly capture the interaction between multiphase flows (Agertz et al., 2007). For example, if fluid mixing is suppressed

[8]We have used the standard off-the-self code, with the indications given in the IC repository (http://www.tapir.caltech.edu/ phopkins/Site/GIZMO_files/gizmo_documentation.html#tests-shocks-sedov). The code was downloaded the 5th of October, 2018. We have not found any "obvious" parameter to "turn the conservation on".

[9]In some cases non-conservative schemes are less noisy.

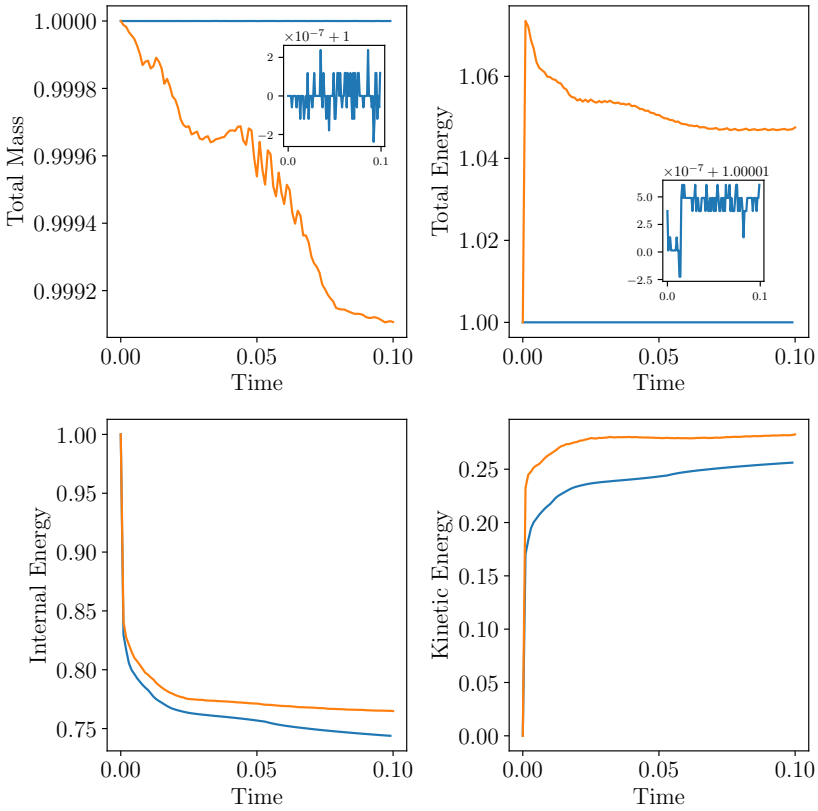

Figure 4.7: Energy and mass evolution for the Sedov explosion test case (section 4.1.4) using the code developed in this work (*blue*), and GIZMO (*orange*). For the total energy and mass a zoom-in has been added to emphasize the conservation properties down to machine accuracy of the new solver.

due to numerical effects, the ram pressure inside clusters would be too low, hindering the stripping of gas of satellite galaxies falling into the cluster.

It is known that standard SPH can not faithfully simulate contact discontinuities, such as those present in the Kelvin Helmholtz (KH) instability (Agertz et al., 2007; Springel, 2010a). This is caused by spurious pressure forces, which appear in regions of steep density gradients. Thus standard SPH solvers will suppress fluid mixing and avoid the non-linear evolution of the instability.

On the other hand, solvers using a grid correctly simulate fluid mixing (Agertz et al., 2007). But, if they are not Galilean invariant, adding a bulk velocity to the problem would completely distort the solution (Hopkins, 2015).

In the following, we will simulate the evolution of a Kelvin-Helmholtz instability. This test will give us an idea about the fluid mixing properties of the code. Also, due to the non-linear nature of this test, eddies will appear at all scales, so it is a good test for the adaptive spatial resolution.

We follow Hopkins (2015) for the setup of the initial conditions. The case is 2D, but

has been adapted to 3D following the same procedure as in previous cases. We set N^2 particles in a regular lattice and we set the mass of the particles such that

$$\rho(y) = \begin{cases} \rho_2 - \Delta\rho \exp\left[(y-0.25)/\Delta y\right] & -0.5 \leq r \leq -0.25 \\ \rho_1 + \Delta\rho \exp\left[-(y-0.25)/\Delta y\right] & -0.25 \leq r \leq 0 \\ \rho_1 + \Delta\rho \exp\left[(y-0.75)/\Delta y\right] & 0 \leq r \leq 0.25 \\ \rho_2 - \Delta\rho \exp\left[-(y-0.25)/\Delta y\right] & 0.25 \leq r \leq 0.5 \end{cases} \tag{4.9}$$

holds. We have set $\Delta y = 0.025$, $\rho_2 = 2$, $\rho_1 = 1$ and $\Delta\rho = 0.5(\rho_2 - \rho_1)$. The pressure is constant ($p = 5/2$) along the domain and $\gamma = 5/3$. The KH instability is characterized by a shear flow along the x direction,

$$v_x(y) = \begin{cases} v_{x2} - \Delta v_x \exp\left[(y-0.25)/\Delta y\right] & -0.5 \leq r \leq -0.25 \\ v_{x1} + \Delta v_x \exp\left[-(y-0.25)/\Delta y\right] & -0.25 \leq r \leq 0 \\ v_{x1} + \Delta v_x \exp\left[(y-0.75)/\Delta y\right] & 0 \leq r \leq 0.25 \\ v_{x2} - \Delta v_x \exp\left[-(y-0.25)/\Delta y\right] & 0.25 \leq r \leq 0.5 \end{cases}, \tag{4.10}$$

and $v_{x1} = 0.5$, $v_{x2} = -0.5$. To allow the growth of the instability, we seed an initial perturbation to the vertical velocity,

$$v_y(x) = \delta v_y \sin\left(4\pi x\right). \tag{4.11}$$

We have run this case with $N = 128, 256, 512$, and up to $t = 10$. In figure 4.8 a summary of the results is presented. The results at $t = 2.5$ are shown at the top row, and, in the bottom row, at $t = 10$. The resolution increases from $N = 128$ to $N = 512$ from left to right.

The solver successfully captures both the linear and non-linear growth of the instability at all resolved scales. However, the evolution is not exactly the same for the three resolutions. This is expected because as we increase the resolution we can resolve smaller scales that, due to the non-linearity of the problem, can have an impact in the evolution at larger scales.

To show a more comprehensive view of the problem, we have produced movies with the time evolution of the KH instability for the three resolutions shown in figure 4.8: `MOV/N128.avi`, `MOV/N256.avi` and `MOV/N512.avi`, respectively. It is clearly seen that the overall evolution is similar for all N, but small scale eddies are more common when the resolution is increased.

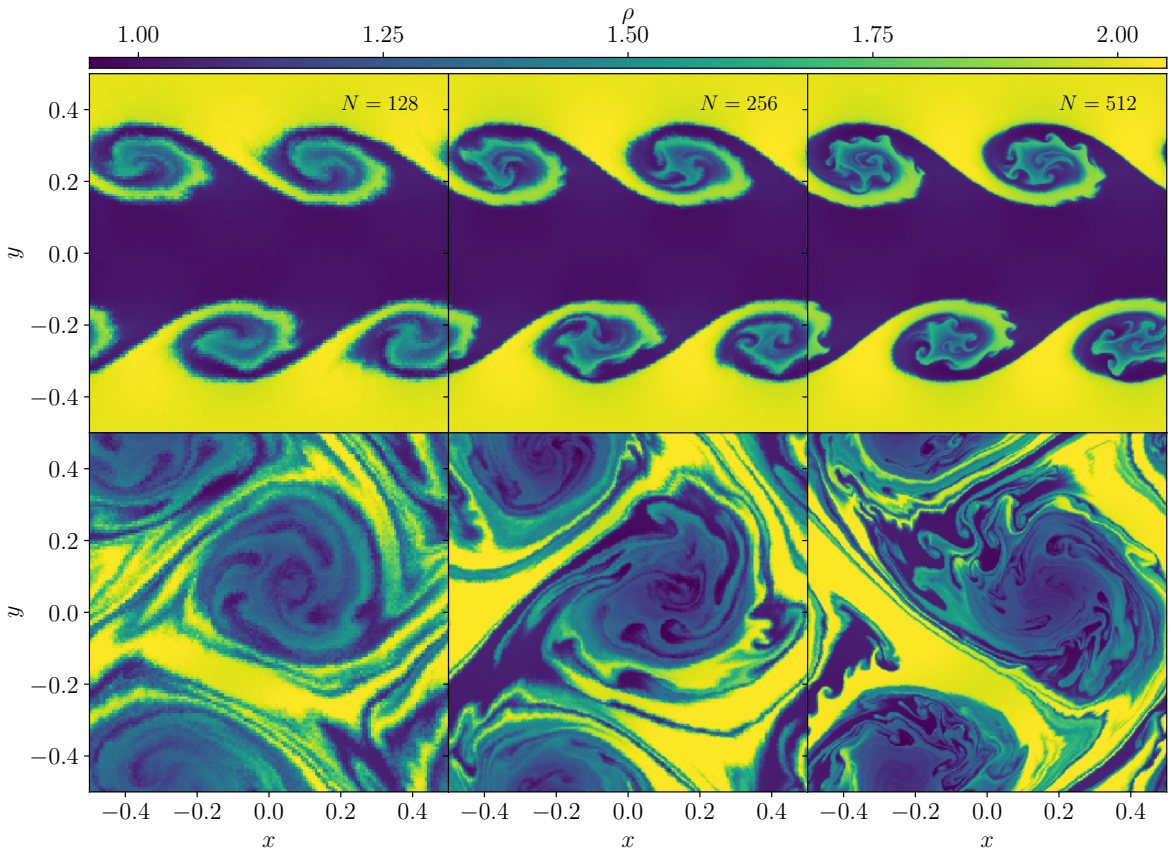

Figure 4.8: Solution of the Kelvin Helmholtz instability (section 4.1.5) using the code developed in this work. The density is shown at $t = 2.5$ in the *top row* and at $t = 7.5$ in the *bottom row*. From left to right, the number of particles per axis is $N = 128, 256, 512$.

4.2 Performance analysis

As stated in previous sections, we take full advantage of the parallelism model of `PKDGRAV3`. Doing so, we can expect good scaling in a wide variety of scenarios. However, it would be easy to degrade the parallel performance of the scheme if, say, we use too many cache request from other processes. Thus, in order to check that the code still performs as expected, we have performed a simple strong scaling test.

There are different metrics to check the scalability of a code. The weak scaling, for example, increases the problem size and number of cores used fixing the theoretical work load of each process. On the other hand, the strong scaling is obtained solving a fixed-size problem with an increasing number of cores. Whereas in the first case the expected solution is a constant time-to-solution, in the second we expect the time-to-solution to decrease with the number of cores used.

To setup this simple case, we have placed randomly $N = 256^3$ particles in a 3D box of

size $L = 1$, and we set the same temperature for all of them. Then, we run 100 iterations of the Kick-Drift-Kick scheme with global time stepping, storing the time taken for each iteration. The time is computed as the median of those timing measurements. We have run this simulation with increasing number of cores, all residing in the same node and NUMA domain.[10] The result of the strong scaling are shown in figure 4.9.

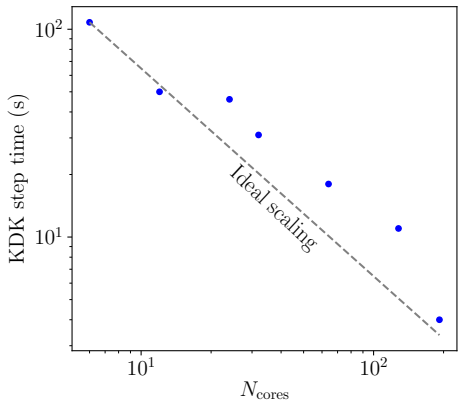

Figure 4.9: Strong scaling of the hydrodynamic solver in a single node with 192 cores in the same NUMA domain. As we use the efficient parallelism strategy of `PKDGRAV3` we can achieve ideal scaling within a node. It must be noted that points above the ideal scaling may have been caused by other processes using the same CPU, as the whole node was not requested for those benchmarks.

As expected, we successfully achieve almost perfect scaling up to 192 cores. We suspect that the points above the ideal scaling are caused by the interference with other processes running in the node, as we did not request for the whole node for each run. Although further tests need to be done to check the scalability in different nodes, we expect that no major difference with the base `PKDGRAV3` will be encountered.

[10] Simplifying, this means that all CPU's share the same RAM, but each one has its own cache.

Chapter 5

Conclusions

In the framework of this *Trabajo de Fin de Máster*, we have developed a state-of-the-art hydrodynamic solver into the efficient N-body code `PKDGRAV3`, namely, MFV. This was done after understanding the underlying mathematical structure of the solver (section 1.1.2) as well as other extensively used scheme, such as SPH (section 1.1.1). Furthermore, in order to be able to successfully implement the solver and to take advantage of the parallelism of `PKDGRAV3`, the code was studied in depth (section 1.2).

With the previous acquired knowledge, we have implemented the MFV scheme into `PKDGRAV3`, explaining in great details the different physical and algorithmic features of the implementation (section 3). This implementation is based in its majority in original code, written from scratch. This allows us to independently check that the scheme behaves as expected.

As no code is free of bugs, we have tested our code with a set of test cases (section 4) extensively used in the Literature. During that process, plenty of bugs were discovered and solved. Those cases were used to test different properties of the solver, namely: spatial second order, individual time stepping, energy and mass conservation *up to machine accuracy* and null artificial transport of angular momentum. Furthermore, we checked that the implementation conserves the inherent scaling properties of `PKDGRAV3` in section 4.2.

In summary, we have implemented and extensively tested an hydrodynamic solver embedded in a N-body code, which in the future can be the base of a cosmological-hydrodynamic code. That and other future improvements will be laid out next.

5.1 Future Work

All the new code that has been implemented during this *Trabajo de Fin de Máster* is intended to perform cosmological hydrodynamical simulations, thus the next lines of works are aimed to that goal. We can summarize the short term goals:

1. Implement **self-gravity** in the solver.

2. Add the gravitational interaction between collisionless particles (i.e., dark matter) and gas particles.

3. Add physical units to the solver and take into account the evolution of the Universe via the **scale-factor**.

4. Add **more physics** into the code, such as thin radiative cooling.

5. Optimize the **memory footprint** and fine-tune CPU performance of the hydrodynamic solver.

They are oriented to have a proper code which can evolve, say, the collapse of a dark matter halo and the gas in it to form a cluster. The next long term goals are aimed to provide a code able to faithfully reproduce star formation and feedback, and doing that in a computationally efficient way:

1. Port the existing solver to **Graphics Processing Units** (GPU's) to improve the time-to-solution.

2. Implement **star formation** from the collapse of dense, cold gas.

3. Follow the **evolution** of the formed stars and their impact in the environment, e.g, with supernovae feedback.

If all the preceding goals are achieved, the result will be a state-of-the-art cosmological hydrodynamic solver with an efficient implementation of both gravity and hydrodynamics. We expect the potential of such a code to be huge, as we could perform truly large (box size $L > 100$ Mpc) scale structure simulations with consistent stellar formation and evolution, something difficult to obtain nowadays.

Chapter 6

Bibliography

Oscar Agertz, Ben Moore, Joachim Stadel, Doug Potter, Francesco Miniati, Justin Read, Lucio Mayer, Artur Gawryszczak, Andrey Kravtsov, Åke Nordlund, Frazer Pearce, Vicent Quilis, Douglas Rudd, Volker Springel, James Stone, Elizabeth Tasker, Romain Teyssier, James Wadsley, and Rolf Walder. Fundamental differences between SPH and grid methods. *Monthly Notices of the Royal Astronomical Society*, 380(3):963–978, 2007. ISSN 00358711. doi: 10.1111/j.1365-2966.2007.12183.x.

Yannick M. Bahé, David J. Barnes, Claudio Dalla Vecchia, Scott T. Kay, Simon D. M. White, Ian G. McCarthy, Joop Schaye, Richard G. Bower, Robert A. Crain, Tom Theuns, Adrian Jenkins, Sean L. McGee, Matthieu Schaller, Peter A. Thomas, and James W. Trayford. The Hydrangea simulations: galaxy formation in and around massive clusters. *Monthly Notices of the Royal Astronomical Society*, 470(4):4186–4208, oct 2017. ISSN 0035-8711. doi: 10.1093/mnras/stx1403. URL https://academic.oup.com/mnras/article/470/4/4186/3868206.

Andrés Balaguera-Antolínez and Cristiano Porciani. Counts of galaxy clusters as cosmological probes: The impact of baryonic physics. *Journal of Cosmology and Astroparticle Physics*, 2013(4), 2013. ISSN 14757516. doi: 10.1088/1475-7516/2013/04/022.

David J Barnes, Scott T Kay, Yannick M. Bahé, Claudio Dalla Vecchia, Ian G. McCarthy, Joop Schaye, Richard G Bower, Adrian Jenkins, Peter A Thomas, Matthieu Schaller, Robert A Crain, Tom Theuns, and Simon D.M. White. The Cluster-EAGLE project: Global properties of simulated clusters with resolved galaxies. *Monthly Notices of the Royal Astronomical Society*, 471(1):1088–1106, oct 2017. ISSN 13652966. doi: 10.1093/MNRAS/STX1647. URL https://academic.oup.com/mnras/article/471/1/1088/3906605.

Josh Barnes and Piet Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324(6096):446–449, 1986. ISSN 00280836. doi: 10.1038/324446a0.

Timothy Barth and Dennis Jespersen. The design and application of upwind schemes on unstructured meshes. In *27th Aerospace Sciences Meeting*, Reston, Virigina, jan 1989.

AIAA, American Institute of Aeronautics and Astronautics. doi: 10.2514/6.1989-366. URL http://arc.aiaa.org/doi/10.2514/6.1989-366.

Rick Beatson and Leslie Greengard. A short course on fast multipole methods. In *New York*, pages 1–37. Oxford University Press, 1997. ISBN 0 19 850190. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.7826{&}rep=rep1{&}type=pdf.

Alejandro Benitez-Llambay, Carlos S. Frenk, Aaron D. Ludlow, and Julio F. Navarro. Baryon-induced dark matter cores in the EAGLE simulations. 17(October):1–17, oct 2018. URL http://arxiv.org/abs/1810.04186.

Greg L. Bryan, Michael L. Norman, Brian W. O'Shea, Tom Abel, John H. Wise, Matthew J. Turk, Daniel R. Reynolds, David C. Collins, Peng Wang, Samuel W. Skillman, Britton Smith, Robert P. Harkness, James Bordner, Ji Hoon Kim, Michael Kuhlen, Hao Xu, Nathan Goldbaum, Cameron Hummels, Alexei G. Kritsuk, Elizabeth Tasker, Stephen Skory, Christine M. Simpson, Oliver Hahn, Jeffrey S. Oishi, Geoffrey C. So, Fen Zhao, Renyue Cen, and Yuan Li. Enzo: An adaptive mesh refinement code for astrophysics. *Astrophysical Journal, Supplement Series*, 211(2), 2014. ISSN 00670049. doi: 10.1088/0067-0049/211/2/19.

H. Cheng, L. Greengard, and V. Rokhlin. A Fast Adaptive Multipole Algorithm in Three Dimensions. *Journal of Computational Physics*, 155(2):468–498, nov 1999. ISSN 00219991. doi: 10.1006/jcph.1999.6355. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999199963556.

Lee Cullen and Walter Dehnen. Inviscid smoothed particle hydrodynamics. *Monthly Notices of the Royal Astronomical Society*, 408(2):669–683, 2010. ISSN 00358711. doi: 10.1111/j.1365-2966.2010.17158.x.

Walter Dehnen. A hierarchical O(N) force calculation algorithm. *Journal of Computational Physics*, 179(1):27–42, 2002. ISSN 00219991. doi: 10.1006/jcph.2002.7026.

Arianna Di Cintio, Chris B. Brook, Andrea V. Macciò, Greg S. Stinson, Alexander Knebe, Aaron A. Dutton, and James Wadsley. The dependence of dark matter profiles on the stellar-to-halo mass ratio: A prediction for cusps versus cores. *Monthly Notices of the Royal Astronomical Society*, 437(1):415–423, 2013. ISSN 00358711. doi: 10.1093/mnras/stt1891.

Jürg Diemand, Ben Moore, and Joachim Stadel. Convergence and scatter of cluster density profiles. *Monthly Notices of the Royal Astronomical Society*, 353(2):624–632, 2004. ISSN 00358711. doi: 10.1111/j.1365-2966.2004.08094.x.

Fabrice Durier and Claudio Dalla Vecchia. Implementation of feedback in smoothed particle hydrodynamics: Towards concordance of methods. *Monthly Notices of the Royal Astronomical Society*, 419(1):465–478, 2012. ISSN 00358711. doi: 10.1111/j.1365-2966.2011.19712.x.

T. Felipe, E. Khomenko, and M. Collados. Magneto-acoustic waves in sunspots: First results from a new three-dimensional nonlinear magnetohydrodynamic code. *Astrophysical Journal*, 719(1):357–377, 2010. ISSN 15384357. doi: 10.1088/0004-637X/719/1/357.

Evghenii Gaburov and Keigo Nitadori. Astrophysical weighted particle magnetohydrodynamics. *Monthly Notices of the Royal Astronomical Society*, 414(1):129–154, 2011. ISSN 00358711. doi: 10.1111/j.1365-2966.2011.18313.x.

R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, dec 1977. ISSN 0035-8711. doi: 10.1093/mnras/181.3.375. URL https://academic.oup.com/mnras/article-lookup/doi/10.1093/mnras/181.3.375.

Pedro Gonnet. Efficient and Scalable Algorithms for Smoothed Particle Hydrodynamics on Hybrid Shared/Distributed-Memory Architectures. apr 2014. URL http://arxiv.org/abs/1404.2303.

L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 135(2):280–292, 1997. ISSN 00219991. doi: 10.1006/jcph.1997.5706.

Leslie Greengard. The rapid evaluation of potential fields in particle systems. page 106. MIT Press, 1988. ISBN 9780262071109.

Leslie Greengard. The Numerical Solution of the N-Body Problem. *Computers in Physics*, 4(2):142, 2013. ISSN 08941866. doi: 10.1063/1.4822898.

Philip M. Gresho and Stevens T. Chan. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. Part 2: Implementation. *International Journal for Numerical Methods in Fluids*, 11(5):621–659, oct 1990. ISSN 0271-2091. doi: 10.1002/fld.1650110510. URL http://doi.wiley.com/10.1002/fld.1650110510.

B. V. Gudiksen, M. Carlsson, V. H. Hansteen, W. Hayek, J. Leenaarts, and J. Martínez-Sykora. The stellar atmosphere simulation code Bifrost. *Astronomy & Astrophysics*, 531:A154, jul 2011. ISSN 0004-6361. doi: 10.1051/0004-6361/201116520. URL https://doi.org/10.1051/0004-6361/201116520http://www.aanda.org/10.1051/0004-6361/201116520.

Philip F. Hopkins. A general class of Lagrangian smoothed particle hydrodynamics methods and implications for fluid mixing problems. *Monthly Notices of the Royal Astronomical Society*, 428(4):2840–2856, 2013. ISSN 00358711. doi: 10.1093/mnras/sts210.

Philip F Hopkins. A New Class of Accurate , Mesh-Free Hydrodynamic Simulation Methods. 000(December), 2015. doi: 10.1093/mnras/stv195.

D. A. Hubber, G. P. Rosotti, and R. A. Booth. GANDALF - Graphical astrophysics code for N-body dynamics and Lagrangian fluids. *Monthly Notices of the Royal Astronomical Society*, 473(2):1603–1632, 2018. ISSN 13652966. doi: 10.1093/mnras/stx2405.

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

Nathalie Lanson and Jean-Paul Vila. Renormalized Meshfree Schemes I: Consistency, Stability, and Hybrid Methods for Conservation Laws. *SIAM Journal on Numerical Analysis*, 46(4):1912–1934, jan 2008a. ISSN 0036-1429. doi: 10.1137/S0036142903427718. URL http://epubs.siam.org/doi/10.1137/S0036142903427718.

Nathalie Lanson and Jean-Paul Vila. Renormalized Meshfree Schemes II: Convergence for Scalar Conservation Laws. *SIAM Journal on Numerical Analysis*, 46(4):1935–1964, jan 2008b. ISSN 0036-1429. doi: 10.1137/S003614290444739X. URL http://epubs.siam.org/doi/10.1137/S003614290444739X.

Randall J. LeVeque. *Numerical Methods for Conservation Laws*, volume 57. Birkhäuser Basel, Basel, 1992. ISBN 978-3-7643-2723-1. doi: 10.1007/978-3-0348-8629-1. URL http://link.springer.com/10.1007/978-3-0348-8629-1.

L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82(12):1013, dec 1977. ISSN 00046256. doi: 10.1086/112164. URL http://adsabs.harvard.edu/cgi-bin/bib{_}query?1977AJ.....82.1013L.

Dylan Nelson, Volker Springel, Annalisa Pillepich, Vicente Rodriguez-Gomez, Paul Torrey, Shy Genel, Mark Vogelsberger, Ruediger Pakmor, Federico Marinacci, Rainer Weinberger, Luke Kelley, Mark Lovell, Benedikt Diemer, and Lars Hernquist. The IllustrisTNG Simulations: Public Data Release. pages 1–30, 2018. URL http://arxiv.org/abs/1812.05609.

Planck Collaboration, N. Aghanim, Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. J. Banday, R. B. Barreiro, N. Bartolo, S. Basak, R. Battye, K. Benabed, J. P. Bernard, M. Bersanelli, P. Bielewicz, J. J. Bock, J. R. Bond, J. Borrill, F. R. Bouchet, F. Boulanger, M. Bucher, C. Burigana, R. C. Butler, E. Calabrese, J. F. Cardoso, J. Carron, A. Challinor, H. C. Chiang, J. Chluba, L. P. L. Colombo, C. Combet, D. Contreras, B. P. Crill, F. Cuttaia, P. de Bernardis, G. de Zotti, J. Delabrouille, J. M. Delouis, E. Di Valentino, J. M. Diego, O. Doré, M. Douspis, A. Ducout, X. Dupac, S. Dusini, G. Efstathiou, F. Elsner, T. A. Enßlin, H. K. Eriksen, Y. Fantaye, M. Farhang, J. Fergusson, R. Fernandez-Cobos, F. Finelli, F. Forastieri, M. Frailis, E. Franceschi, A. Frolov, S. Galeotta, S. Galli, K. Ganga, R. T. Génova-Santos, M. Gerbino, T. Ghosh, J. González-Nuevo, K. M. Górski, S. Gratton, A. Gruppuso, J. E. Gudmundsson, J. Hamann, W. Handley, D. Herranz, E. Hivon, Z. Huang, A. H. Jaffe, W. C. Jones, A. Karakci, E. Keihänen, R. Keskitalo, K. Kiiveri, J. Kim, T. S. Kisner, L. Knox, N. Krachmalnicoff, M. Kunz, H. Kurki-Suonio, G. Lagache, J. M. Lamarre, A. Lasenby,

M. Lattanzi, C. R. Lawrence, M. Le Jeune, P. Lemos, J. Lesgourgues, F. Levrier, A. Lewis, M. Liguori, P. B. Lilje, M. Lilley, V. Lindholm, M. López-Caniego, P. M. Lubin, Y. Z. Ma, J. F. Macías-Pérez, G. Maggio, D. Maino, N. Mandolesi, A. Mangilli, A. Marcos-Caballero, M. Maris, P. G. Martin, M. Martinelli, E. Martínez-González, S. Matarrese, N. Mauri, J. D. McEwen, P. R. Meinhold, A. Melchiorri, A. Mennella, M. Migliaccio, M. Millea, S. Mitra, M. A. Miville-Deschênes, D. Molinari, L. Montier, G. Morgante, A. Moss, P. Natoli, H. U. Nørgaard-Nielsen, L. Pagano, D. Paoletti, B. Partridge, G. Patanchon, H. V. Peiris, F. Perrotta, V. Pettorino, F. Piacentini, L. Polastri, G. Polenta, J. L. Puget, J. P. Rachen, M. Reinecke, M. Remazeilles, A. Renzi, G. Rocha, C. Rosset, G. Roudier, J. A. Rubiño-Martín, B. Ruiz-Granados, L. Salvati, M. Sandri, M. Savelainen, D. Scott, E. P. S. Shellard, C. Sirignano, G. Sirri, L. D. Spencer, R. Sunyaev, A. S. Suur-Uski, J. A. Tauber, D. Tavagnacco, M. Tenti, L. Toffolatti, M. Tomasi, T. Trombetti, L. Valenziano, J. Valiviita, B. Van Tent, L. Vibert, P. Vielva, F. Villa, N. Vittorio, B. D. Wandelt, I. K. Wehus, M. White, S. D. M. White, A. Zacchei, and A. Zonca. Planck 2018 results. VI. Cosmological parameters. pages 1–71, jul 2018. URL http://arxiv.org/abs/1807.06209.

Douglas Potter, Joachim Stadel, and Romain Teyssier. PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys. *Computational Astrophysics and Cosmology*, 4(1):2, may 2017. ISSN 2197-7909. doi: 10.1186/s40668-017-0021-1. URL https://doi.org/10.1186/s40668-017-0021-1.

C. Power, Julio F. Navarro, Adrian Jenkins, Carlos S. Frenk, Simon D. M. White, Volker Springel, Joachim Stadel, and Thomas Quinn. The inner structure of CDM haloes – I. A numerical convergence study. *Monthly Notices of the Royal Astronomical Society*, 338(1):14–34, jan 2003. ISSN 0035-8711. doi: 10.1046/j.1365-8711.2003.05925.x. URL http://arxiv.org/abs/astro-ph/0201544{%}5Cnhttp://mnras.oxfordjournals.org/cgi/doi/10.1046/j.1365-8711.2003.05925.x{%}5Cnhttp://dx.doi.org/10.1046/j.1365-8711.2003.05925.xhttps://academic.oup.com/mnras/article-lookup/doi/10.1046/j.1365-8711.2003.05925.x.

Takayuki R. Saitoh and Junichiro Makino. A necessary condition for individual time steps in SPH simulations. *Astrophysical Journal*, 697(2 PART 2):99–102, 2009. ISSN 15384357. doi: 10.1088/0004-637X/697/2/L99.

Takayuki R. Saitoh and Junichiro Makino. A density-independent formulation of smoothed particle hydrodynamics. *Astrophysical Journal*, 768(1), 2013. ISSN 15384357. doi: 10.1088/0004-637X/768/1/44.

Joop Schaye, Claudio Dalla Vecchia, C. M. Booth, Robert P.C. Wiersma, Tom Theuns, Marcel R. Haas, Serena Bertone, Alan R. Duffy, I. G. McCarthy, and Freeke van de Voort. The physics driving the cosmic star formation history. *Monthly Notices of the Royal Astronomical Society*, 402(3):1536–1560, 2010. ISSN 00358711. doi: 10.1111/j.1365-2966.2009.16029.x.

Joop Schaye, Robert A. Crain, Richard G. Bower, Michelle Furlong, Matthieu Schaller, Tom Theuns, Claudio Dalla Vecchia, Carlos S. Frenk, I. G. McCarthy, John C. Helly, Adrian Jenkins, Y. M. Rosas-Guevara, Simon D. M. White, Maarten Baes, C. M. Booth, Peter Camps, Julio F. Navarro, Yan Qu, Alireza Rahmati, Till Sawala, Peter A. Thomas, and James Trayford. The EAGLE project: simulating the evolution and assembly of galaxies and their environments. *Monthly Notices of the Royal Astronomical Society*, 446(1):521–554, jan 2015. ISSN 1365-2966. doi: 10.1093/mnras/stu2058. URL http://academic.oup.com/mnras/article/446/1/521/1316115/The-EAGLE-project-simulating-the-evolution-and.

L. I. Sedov. Similarity and Dimensional Methods in Mechanics, Tenth Edition. 1993. URL http://books.google.com.hk/books/about/Similarity{_}and{_}Dimensional{_}Methods{_}in{_}Me.html?id=xXSg388Su38C{&}pgis=1.

Volker Springel. The cosmological simulation code GADGET-2. *Monthly Notices of the Royal Astronomical Society*, 364(4):1105–1134, 2005. ISSN 00358711. doi: 10.1111/j.1365-2966.2005.09655.x.

Volker Springel. Smoothed Particle Hydrodynamics in Astrophysics. *Annual Review of Astronomy and Astrophysics*, 48(1):391–430, aug 2010a. ISSN 0066-4146. doi: 10.1146/annurev-astro-081309-130914. URL http://arxiv.org/abs/1109.2219{%}0Ahttp://dx.doi.org/10.1146/annurev-astro-081309-130914http://www.annualreviews.org/doi/10.1146/annurev-astro-081309-130914.

Volker Springel. E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh. *Monthly Notices of the Royal Astronomical Society*, 401(2):791–851, 2010b. ISSN 00358711. doi: 10.1111/j.1365-2966.2009.15715.x.

Volker Springel and Lars Hernquist. Cosmological smoothed particle hydrodynamics simulations: The entropy equation. *Monthly Notices of the Royal Astronomical Society*, 333(3):649–664, 2002. ISSN 00358711. doi: 10.1046/j.1365-8711.2002.05445.x.

Joachim Stadel. *Cosmological N-body simulations and their analysis*. PhD thesis, University of Washington, 2001. URL http://adsabs.harvard.edu/abs/2001PhDT.....21S.

R. Teyssier. Cosmological hydrodynamics with adaptive mesh refinement. *Astronomy & Astrophysics*, 385(1):337–364, apr 2002. ISSN 0004-6361. doi: 10.1051/0004-6361:20011817. URL http://www.aanda.org/10.1051/0004-6361:20011817.

Eleuterio F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-25202-3. doi: 10.1007/b79761. URL http://link.springer.com/10.1007/b79761.

James W. Wadsley, Benjamin W. Keller, and Thomas R. Quinn. GASOLINE2: A modern smoothed particle hydrodynamics code. *Monthly Notices of the Royal Astronomical Society*, 471(2):2357–2369, 2017. ISSN 13652966. doi: 10.1093/mnras/stx1643.

J.W. Wadsley, Joachim Stadel, and Thomas Quinn. Gasoline: a flexible, parallel implementation of TreeSPH. *New Astronomy*, 9(2):137–158, feb 2004. ISSN 13841076. doi: 10.1016/j.newast.2003.08.004. URL http://arxiv.org/abs/astro-ph/0303521{%}0Ahttp://dx.doi.org/10.1016/j.newast.2003.08.004https://linkinghub.elsevier.com/retrieve/pii/S138410760300143X.

Liang Wang, Aaron A. Dutton, Gregory S. Stinson, Andrea V. Macciò, Camilla Penzo, Xi Kang, Ben W. Keller, and James Wadsley. NIHAO project - I. Reproducing the inefficiency of galaxy formation across cosmic time with a large sample of cosmological hydrodynamical simulations. *Monthly Notices of the Royal Astronomical Society*, 454 (1):83–94, 2015. ISSN 13652966. doi: 10.1093/mnras/stv1937.