Alma Mater Studiorum – Università di Bologna

Dipartimento di Fisica e Astronomia
Corso di Laurea Magistrale in Astrofisica e Cosmologia

Dynamic Zoom Simulations in AREPO

Tesi di laurea

Presentata da:
Riccardo Zangarelli

Relatore:
Chiar.mo Prof. Marco Baldi

Correlatori:
Prof. Federico Marinacci
Dott. Enrico Garaldi

Anno Accademico 2021-2022

S A T O R

A R E P O

T E N E T

O P E R A

R O T A S

Sator square (roman puzzle)

Contents

Sommario **v**

Abstract **vi**

1 Introduction **1**
1.1 Fundamentals of cosmology 1
1.1.1 The Friedmann-Lemaître-Robertson-Walker metric 2
1.1.2 The Friedmann equations and cosmological parameters . . . 4
1.2 Structure formation in the early Universe 7
1.2.1 Jeans perturbation theory 8
1.2.2 Hot and cold dark matter 11
1.3 Processes of galaxy formation 12
1.3.1 Gas cooling and star formation 13
1.3.2 Feedback processes 15
1.4 The numerical approach 17

2 Numerical techniques **20**
2.1 N-body solvers 20
2.1.1 Particle-mesh method 21
2.1.2 Hierarchical multipole method 24
2.1.3 Tree-PM method 27
2.2 Hydrodynamics 29
2.2.1 Smoothed particle hydrodynamics 30
2.2.2 Moving mesh schemes 32

3 The Dynamic Zoom Simulations algorithm **36**
3.1 Algorithm outline 37
3.1.1 Tree-based DZS 39
3.2 The AREPO implementation 42
3.2.1 Initial setup 43
3.2.2 Tree walk 46

3.2.3 Node derefinement 54
3.2.4 Particle elimination and insertion point 55
3.2.5 Additional considerations 59
3.2.6 Baryon derefinement 60

4 Algorithm validation **64**
4.1 Output analysis 64
4.1.1 Lightcone Halo Mass Function 67
4.1.2 Sky-projected lightcone 68
4.1.3 3D lightcone 69
4.1.4 Particle displacements 77
4.2 Performance analysis 79
4.2.1 Run time performance 80
4.2.2 Work-load balance 82

5 Summary, conclusions and future prospects **85**
5.1 Work-load balance and DZS special stop 86
5.2 Further validation and additional physics 87
5.3 Conclusions 88

Bibliography **89**

Sommario

Nell'astrofisica moderna, le simulazioni cosmologiche rappresentano lo strumento di analisi principale per modellizzare e interpretare la sempre crescente quantità di dati forniti dalle campagne osservative. La prossima generazione di survey, tuttavia, richiederà simulazioni con risoluzione e volume coperto senza precedenti, con un impatto estremamente elevato sulle risorse computazionali. Per affrontare questo problema e ridurre il costo computazionale delle future simulazioni, ho implementato nel codice d'avanguardia `AREPO` un nuovo metodo denominato "Dynamic Zoom Simulations" (DZS), proposto originariamente in un contesto dark-matter-only nel più datato codice `PGADGET-3` da Garaldi et al. (2020). Il metodo sfrutta il fatto che il confronto significativo fra dati osservativi e output simulati richiede che questi ultimi siano in una forma cosiddetta "lightcone-like", la quale include nei file di output soltanto una frazione del volume totale simulato. Per questo motivo, il metodo DZS punta a concentrare gli sforzi computazionali di una simulazione all'interno del cono di luce, riducendo dinamicamente la risoluzione al di fuori di esso. Ciò riduce di molto le risorse computazionali richieste con modifiche minime rispetto a una run standard. In particolare, nel contesto dark-matter-only testato in questo elaborato, le simulazioni eseguite con il metodo DZS sono in grado di riprodurre accuratamente delle quantità lightcone-like come la Lightcone Halo Mass Function, il cono di luce proiettato sulla volta celeste e il cono di luce tridimensionale, nonché di generare dislocazioni molto contenute nella posizione finale delle singole particelle. Dopo aver implementato e testato accuratamente l'algoritmo originale in un codice più moderno, ho iniziato ad occuparmi del suo principale difetto, vale a dire la mancanza di supporto per la fisica barionica. In questo senso, l'implementazione è ancora in corso, ma ci si aspetta che abbia un impatto ancora maggiore sulle risorse computazionali risparmiate. Anche nel caso dark-matter-only, comunque, la mia implementazione consente di eseguire la simulazione con la box più grande tra quelle testate in quasi metà del tempo richiesto per una simulazione standard. Questo risultato sarà probabilmente ancora migliore in simulazioni con risoluzioni più elevate e volumi più grandi, così da rendere il metodo DZS uno strumento estremamente promettente per ridurre significativamente l'impatto computazionale della prossima generazione di simulazioni cosmologiche.

Abstract

In modern astrophysics, cosmological simulations represent the primary analysis tool to model and interpret the ever-growing amount of data provided by observational campaigns. However, the next generation of surveys will require simulations with unprecedented resolutions and volume coverages, whose impact on computational resources will be extremely high. To address this issue and reduce the computational cost of these upcoming simulations, I implemented in the state of the art code `AREPO` a new method dubbed "Dynamic Zoom Simulations" (DZS), originally proposed in a dark-matter-only-fashion by Garaldi et al. (2020) in the older code `PGADGET-3`. This method exploits the fact that a meaningful comparison of real data and simulated output requires the latter to be in a lightcone-like form, which includes only a fraction of the total simulated volume in the output files. Therefore, the DZS methods aims to focus the computational efforts of a simulation inside the lightcone by dynamically reducing the resolution outside of it. This largely reduces the computational resources employed with only minimal modifications with respect to a standard run. Specifically, in the dark-matter-only scenario tested in this work, simulations performed with the DZS method are capable of accurately reproducing lightcone-like quantities such as the Lightcone Halo Mass Function, the sky-projected lightcone and the $3D$ lightcone with deviations mostly below the percent level, as well as generating very contained displacements in the final position of individual particles. After implementing and thoroughly testing the original DZS algorithm in a more modern code, I also started to address its main drawback, namely the lack of support for baryonic physics. The implementation is still ongoing, but is expected to have an even-bigger impact on the computational resources saved. Nevertheless, even in the dark-matter-only case, my implementation is capable of running the simulation with the biggest box among those tested in nearly half of the original run time. This result is likely to get even better in simulations with higher resolutions and larger volumes, making the DZS method a very promising tool to ease the computational burden of the next generation of cosmological simulations.

1 | Introduction

Cosmology, which is the study of the Universe as a whole, represents a relatively young branch of astrophysics: the concept of an evolving Universe was unconceivable until the early 20^{th} century, when Einstein's generaly relativity (Einstein 1915) paved the way to new cosmological models, later formalised by the Friedmann equations (Friedmann 1922). These showed that our Universe is unlikely to be stationary, as later confirmed by the Hubble-Lemaître law (Hubble 1929), which highlighted a correlation between the distance of an object and its recession velocity from the observer, showing that the Universe is in fact expanding. The newfound dynamical nature of the large-scale Universe gave rise to the topic of the formation and evolution of objects such as stars and galaxies, topic whose understanding requires to consider other fundamental results: for example, the first observational evidence of "missing" matter (Zwicky 1933) and of the accelerated expansion of the Universe (Riess et al. 1998) imply the existence of two components, respectively dark matter and dark energy, whose nature is still unknown.

This chapter will present the cosmological framework and main physical processes behind the formation of cosmic structures and galaxies, and explain how in order to better understand these processes, it is of paramount importance to reproduce observational data through numerical simulations; it will also highlight how it is crucial to analyse and possibly extend the capabilities of numerical methods in modeling observational data.

1.1 Fundamentals of cosmology

The standard cosmological model in use nowadays is based on a fundamental principle, which serves as guidance in the construction of a theoretical framework. This so-called Cosmological Principle states that on sufficiently large scales the Universe is homogeneous and isotropic. The former property implies that there are no privileged positions in the Universe, the latter that there are no privileged directions. The cosmic microwave background (CMB, e.g. Penzias and Wilson 1965) represents an

observational proof of the isotropy of the Universe, because the temperature map of the CMB radiation (i.e. its main observable property) is nearly independent of direction. In order for the assumption of homogeneity to also hold, the observer must not be in a privileged location (e.g. a spherically simmetric density distribution looks isotropic only if observed from its center, but it can be inhomogeneous). If the assumption of not being in such a location (i.e. of the Universe not having privileged positions) holds, and if there are observational proofs of isotropy, the Cosmological Principle represents a good starting assumption to model our Universe.

In order to elaborate a cosmological model, it is also necessary to select an appropriate theory of gravity.[1] The most accurate theory available in this regard is general relativity (Einstein 1915), which uses a geometrical framework to describe space-time and the relationship between changes in its curvature and the matter inside it. In fact, this framework is employed in a cosmological model through the definition of a so-called metric tensor.

1.1.1 The Friedmann-Lemaître-Robertson-Walker metric

The definition of a general metric tensor can be obtained through the concept of distance s between two events in the $1D + 3D$ space-time (i.e. a three-dimensional space with an additional coordinate for time). Specifically, an event is a point of space-time with coordinates (x_0, x_1, x_2, x_3), where the first is the time coordinate and the other three are the cartesian space coordinates, i.e. the position vector $\mathbf{x}$. When considering events separated by infinitesimal displacements $\mathrm{d}x_i$, $i = 0,1,2,3$ in a space-time whose geometry is modified by a gravitational potential $\Phi(\mathbf{x})$, the square of the generic distance $\mathrm{d}s$ can be written as

$$\mathrm{d}s^2 = \sum_{i,j=0}^{3} g_{ij}\mathrm{d}x_i\mathrm{d}x_j = g_{ij}\mathrm{d}x^i\mathrm{d}x^j \,, \tag{1.1}$$

where the last equality follows from Einstein summmation notation[2] and g_{ij} is the metric tensor. The components of g_{ij} are a function of the gravitational potential and determine how the distance between events and the geometry of space-time are affected by the matter inside it. The reduction of this concept to empty space ($\Phi(\mathbf{x}) = 0\ \forall\, \mathbf{x}$) yields the so-called Minkowski metric tensor η_{ij}, whose matrix representation

[1] From a cosmological point of view, gravity is often considered the most important force, because it is the strongest on large distances among all forces of nature (Coles and Lucchin 2002).

[2] This notation implies a sum over pairs of repeated subscript and superscript indices, without the need to explicitly write the $\sum$ operator. It was introduced in Einstein (1916).

can be written as

$$\eta = \begin{pmatrix} c^2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} . \tag{1.2}$$

The expression for ds^2 is readily obtained through the use of matrix multiplication:

$$ds^2 = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} c^2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = c^2 dx_0^2 - (dx_1^2 + dx_2^2 + dx_3^2) . \tag{1.3}$$

It is easy to show that for events connected by a beam of light ds^2 is always zero, for example by considering a spherical wave front of light, which propagates with speed c and covers a spacial distance $\sqrt{dx_1^2 + dx_1^2 + dx_3^2}$ in a time dx_0. In this case, the distance is classified as light-like and the events can be causally connected. This also occurs if $c^2 dx_0^2$ is greater than $dx_1^2 + dx_2^2 + dx_3^2$ ($ds^2 > 0$, time-like distance). If instead $dx_1^2 + dx_2^2 + dx_3^2 > c^2 dx_0^2$ ($ds^2 < 0$, space-like distance), the events cannot be causally connected, because that would require an information propagation speed greater than c (which is not allowed by relativity).[3]

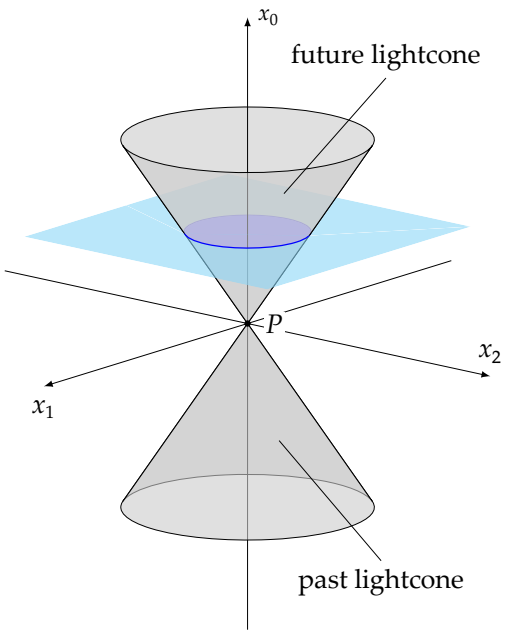


Figure 1.1: $1D + 2D$ sketch of the past and future lightcones (grey) of a reference event P. With a restriction to a fixed time (cyan rectangle), the space component of the lightcone is a circle (blue), whereas it would be a sphere with three spatial dimensions (figure adapted from D'Inverno 1992).

By choosing a reference event (typically a point of space at the present time) it is possible to build the set of all events that are causally connected to that event, both in the past (negative time difference) and in the future (positive time difference). In the $1D + 2D$ representation of **fig. 1.1**, the two sets of

[3]The signs of η_{ij} in **eq. 1.2** and **eq. 1.3** are defined conventionally and in fact can be swapped (to yield $ds^2 < 0$ as a time-like distance and $ds^2 > 0$ as a space-like distance). The notation adopted here is the one employed in Coles and Lucchin (2002) and D'Inverno (1992).

causally connected events have the shape of cones with vertices located on the reference event, hence the name "lightcone" given to them.

The concept of metric can be easily adapted to the Cosmological Principle to yield the most general metric which obeys the assumptions of homogeneity and isotropy, i.e. the Friedmann-Lemaître-Robertson-Walker metric (FLRW, e.g. Walker 1937):

$$\mathrm{d}s^2 = c^2\mathrm{d}t^2 - a(t)^2\left[\frac{\mathrm{d}r^2}{1-kr^2} + r^2(\mathrm{d}\theta^2 + \sin^2\theta\mathrm{d}\phi^2)\right], \tag{1.4}$$

where $a(t) \geq 0$ is the so-called scale factor, a parameter which describes the dynamical behaviour of the Universe (i.e. its expansion or contraction), is usually dimensionless and depends on the time t. Note that spherical coordinates have been employed, but in place of the usual radial coordinate R there is the quantity $r \equiv R/a$. The parameter k can only be equal to 0, 1 or −1 and is related to the geometry of the Universe as follows:

- $k = 0$ corresponds to a so-called flat Universe: the space is Euclidean and its 2D representation is a flat surface;
- $k = 1$ corresponds to a closed (spherical) Universe, which has finite volume but no boundaries, such as the 2D surface of a sphere.
- $k = -1$ corresponds to an open (hyperbolic) Universe; as in the Euclidean case, this space is infinite.

The FLRW metric represents a fundamental tool to build a cosmological model, mainly because inserting that metric in the Einstein field equations of general relativity yields the Friedmann equations, which describe the time evolution of the geometry of the Universe.

1.1.2 The Friedmann equations and cosmological parameters

The Einstein field equations of general relativity can be written as (e.g. D'Inverno 1992):

$$R_{ij} - \frac{1}{2}g_{ij}R = \frac{8\pi G}{c^2}T_{ij}\,, \tag{1.5}$$

where R_{ij} and R are the Ricci tensor and the Ricci scalar, which depend on the gravitational potential, G is the gravitational constant and T_{ij} is the stress-energy tensor, which quantifies the distribution of matter and energy. Here this tensor is assumed to be that of a perfect fluid, as is usually the case in a cosmological framework. By using the Friedmann-Lemaître-Robertson-Walker metric of **eq. 1.4** to describe g_{ij},

the resulting equations are not identities only if $i = j$, with the three spatial components yielding the same equation (as it is expected under the Cosmological Principle). These time-time and space-space relations yield the following Friedmann equations for $a(t)$:

$$\ddot{a} = -\frac{4\pi}{3}G\left(\rho + \frac{3p}{c^2}\right)a \,, \tag{1.6}$$

$$\dot{a}^2 + kc^2 = \frac{8\pi}{3}G\rho a^2 \,, \tag{1.7}$$

where ρ is the fluid density and p is its pressure. Note that in the first equation, $\ddot{a}$ is always negative if the density and pressure are both positive (which is the case of ordinary fluids), meaning that the expansion or contraction of a Universe modeled by these equations is decelerating. It is also worth noting that the above equations do not allow a static solution (that is, dimensions aside, $\ddot{a} = \dot{a} = 0$), unless pressure and density have opposing signs. In fact, this requirement can be met by arbitrarily altering the geometry of space-time through a so-called cosmological constant $\Lambda > 0$, which modifies the Einstein equations as follows:

$$R_{ij} - \frac{1}{2}g_{ij}R - \Lambda g_{ij} = \frac{8\pi G}{c^2}T_{ij} \,. \tag{1.8}$$

Note that Λ can still theoretically represent an additional matter or energy component (whose density and pressure have opposing signs) if intended as a positive addition to T_{ij}. Thanks to this new component (or geometry modification), the Friedmann equations can have a static solution (the Einstein model, Einstein 1917), which holds for a closed Universe ($k = 1$) made of dust ($p = 0$), where the cosmological constant has exactly the value needed to yield a density ρ_Λ and pressure p_Λ such that $\rho_\Lambda = -3p_\Lambda/c^2$. This is a very restrictive and unstable condition, because any variation of Λ leads the Universe to a dynamic state. Moreover, our Universe does not seem to be closed; in fact, studies such as the one carried on through the Wilkinson Microwave Anisotropy Probe (WMAP, Bennett et al. 2003) and a subsequent one by the Planck collaboration (2016) have shown that our Universe is flat with a high confidence level. These observational results are usually achieved by fitting observational data with models which depend on quantities named cosmological parameters.

First and foremost, these quantities include the so-called Hubble parameter $H \equiv \dot{a}/a$, whose name comes from its direct involvement in the aforementioned Hubble-Lemaître law. Specifically, consider the following time derivative of the quantity $R = ra$ of a certain astrophysical object (i.e. the modulus of its velocity $\mathbf{v}$):

$$|\mathbf{v}| = \frac{\mathrm{d}(ra)}{\mathrm{d}t} = r\frac{\mathrm{d}a}{\mathrm{d}t}a\frac{\mathrm{d}r}{\mathrm{d}t} = R\frac{\dot{a}}{a} + a\mathbf{v}_{pec} = RH + a\mathbf{v}_{pec} \,. \tag{1.9}$$

When the peculiar velocity of an object (i.e. the component of its motion which is not due to the expansion of the universe) is neglected, the Hubble parameter represents the proportionality constant of the relation between the distance of an object and its velocity. Since the scale factor is a function of time, so is the Hubble parameter; its value at the present time is often referred to as "Hubble constant" H_0 because it is independent of space. Furthermore, the fact that objects are receding from the observer implies that their emission will be shifted to redder wavelengths due to the Doppler effect. By considering a monochromatic beam of light with a rest wavelength λ_e emitted by a source and its observed wavelength λ_o, it is possible to define the redshift z of that source as

$$z \equiv \frac{\lambda_o - \lambda_e}{\lambda_e} \,. \tag{1.10}$$

It can be shown (e.g. Condon and Matthews 2018) that the redshift of a beam of light emitted at a time t and reaching the observer at t_0 is related to the scale factor $a(t)$ as $z = a(t_0)/a(t) - 1$. In fact, the redshift is often employed independently of a specific source and generally intended as $z(t)$, with $z(t_0)$ being 0 by definition.

The other major cosmological parameters Besides H_0 are directly related to the geometry of the Universe and its matter and energy components. These parameters are obtained by setting $k = 0$ in **eq. 1.7** and solving for ρ:

$$\rho_c = \frac{3H^2}{8\pi G} \,. \tag{1.11}$$

This "critical density" ρ_c is the value needed to make the Universe flat. Moreover, if $\rho > \rho_c$ the Universe is closed, whereas if $\rho < \rho_c$ the Universe is open. The density parameter $\Omega \equiv \rho/\rho_c$ summarizes this concept by being either equal to 1, greater than 1 or smaller than 1, respectively. Generally speaking, there are multiple components that add up to the total density parameter Ω_{tot}, notably baryonic and dark matter ($\Omega_m = \Omega_b + \Omega_c$), radiation ($\Omega_r$), and dark energy ($\Omega_\Lambda$). The latter is needed because, as previously mentioned, the accelerated expansion of the Universe discovered in 1998 does not agree with the Friedmann equations, which only allow decelerating Universes. To solve this issue, it is possible to reintroduce the cosmological constant Λ, which rather than nullifying $\ddot{a}$ (as in the Einstein model), changes its sign. Moreover, observational constraints on $\Omega_{tot,0}$ (i.e. its value at the present time) show that it is very close to one (the aforementioned Planck collaboration result is $1 - \Omega_{tot,0} = -0.052^{+0.049}_{-0.055}$), but matter only contributes to that as $\Omega_{m,0} = 0.315 \pm 0.013$; furthermore, the energy density of radiation yields $\Omega_{r,0} \approx 10^{-5}$ due to the extremely small temperature of the main source of photons (that is, the CMB, which has $T \approx 2.72$ K, e.g. Planck Collaboration 2016). The cosmological constant acts as the missing component to ensure the flatness of the Universe and is

associated to an unknown "dark energy", which is also responsible for the accelerated expansion.

Another component whose nature is still unknown is dark matter (DM), which in fact constitutes the 84% of the total matter in the Universe (Planck Collaboration 2016). It has never been directly observed because it is supposed to interact extremely weakly with electromagnetic radiation (hence the name), but its existence is proven by the gravational effects that it exerts on visible matter, such as the flattening of galactic rotation curves (e.g. van Albada et al. 1985), the increase of the velocity dispersion in galaxy clusters (e.g. Zwicky 1933), and lensing effects (e.g. Zheng et al. 2012).[4] Moreover, dark matter plays a major role in the formation of structures (e.g. Blumenthal et al. 1984) as will be explained in the next section, which also briefly introduces the framework of the early Universe.

1.2 Structure formation in the early Universe

Observations show that our Universe is homogeneous and isotropic of sufficiently large scales, which nowadays span at least ≈ 150 Mpc (Marinoni et al. 2012). Smaller scale density fluctuations, however, do indeed exist and give rise to the distribution of astrophysical objects, from the larger spatial scale of galaxy clusters to the smaller scale of stars, planets and minor bodies. Aggregations of these objects form the large scale cosmic structure, where sites of clustering are connected by web-like filaments (Bond et al. 1996) and leave vast volumes of the universe (the so-called cosmic voids) mostly free of matter. First of all, it is appropriate to understand how these inhomogeneities formed in the first place by briefly analysing the history of the early Universe. Without a cosmological constant, the function $a(t)$ can only be concave down (due to the negative sign of $\ddot{a}$, see **eq. 1.6**), and this concavity necessarily implies an intersection with the temporal axis, marking the "time zero" of the Universe. Theoretically, a cosmological constant could change the sign of $\ddot{a}$ and avoid the intersection, but in our Universe the value of Λ is so small ($< 10^{-55}$ cm^{-2}, Linde 1974) that its effects are not relevant in the early Universe.

The instant $t = 0$ is called "Big Bang", and in fact it represents a mathematical singularity, because both the Hubble parameter and the density tend to infinity for $t \to 0$. In the extreme conditions of the Big Bang, general relativity does not hold anymore, because both quantum and gravitational interactions share the same scale. In fact, there is a specific time t_p in which the typical scales of gravity and quantum

[4] For an extensive review of dark matter and of its observational evidence, see for example Bertone et al. (2005).

mechanics are exactly equal. This so-called Planck time $t_p = 10^{-43}$ s marks the moment after which gravitational effects can be safely described without employing quantum corrections. However, for $t < t_p$, including the singularity at $t = 0$, physics is currently unable to properly treat the evolution of the Universe.

There are other issues affecting the standard cosmological model: for example, the unknown nature of the cosmological constant Λ, as well as its extremely small but non-zero value necessary to ensure both the accelerated expansion and the flat geometry. In fact, flatness itself is also a problem because of the following dependency of Ω_{tot} on the scale factor and consequently, on time (Linde 1984):

$$|\Omega_{tot} - 1| = \dot{a}(t)^{-2} \, . \tag{1.12}$$

In the early Universe, the matter-energy contribution of cosmological constant was negligible and the expansion was decelerating, yielding $\ddot{a} < 0$ and $\dot{a}$ decreasing with time. Consequently, $|\Omega_{tot} - 1|$ has an increasing trend which leads to $|\Omega_{tot}(t_0)| \gg 1$ if the total density parameter was not extremely close to 1 in the early Universe ($|1 - \Omega_{tot}(t_p)| < 10^{-59}$, Linde 1984). This is a very strict condition, and ascribing it to coincidence would not be satisfactory from a physical point of view. To provide a theoretical explanation for the flat geometry, it is possible to introduce a so-called inflationary period in the early Universe ($t \lesssim 10^{-10}$ s, Linde 1984; Linde 1990; Starobinskii 1983), in which the expansion is characterised by $\ddot{a} > 0$, and thus by a decrease of $|\Omega_{tot} - 1|$ with time. If the inflationary period covers a long enough time interval $\Delta t = t_f - t_i$, it can make Ω_{tot} asimptotically tend to 1 for $t \to t_f$. Moreover, inflation plays a key role in structure formation: the abrupt accelerated expansion that characterises this period generates small fluctuations[5] in the density field $\rho(\mathbf{x})$ (where "small" means $(\rho(\mathbf{x}) - \rho_0)/\rho_0 \ll 1$, with ρ_0 mean density, Linde 1990). These fluctuations can be studied by intending them as perturbations of the otherwise constant field $\rho(\mathbf{x})$, and can eventually grow into the gravitationally bound structures observed today.

1.2.1 Jeans perturbation theory

The Jeans perturbation theory (Jeans 1902) provides a criterion to determine whether a perturbation is able to grow over time (i.e. if it is unstable) or not, under the effect of a gravitational potential. More specifically, consider the following system of equations which describe the evolution of a fluid (e.g. Landau and Lifshitz 1959; Shu

[5]These fluctuations are actually detectable in the CMB radiation, where they arise as temperature inhomogeneities δT, with $\delta T/T \approx 10^{-6}$ (Smoot et al. 1992).

1992):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \,, \tag{1.13}$$

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\frac{\nabla P}{\rho} - \nabla \Phi \,, \tag{1.14}$$

$$\nabla^2 \Phi = 4\pi G \rho \tag{1.15}$$

$$\frac{\partial s}{\partial t} + \mathbf{v} \cdot \nabla s = 0 \,, \tag{1.16}$$

where ρ is the fluid density, $\mathbf{v}$ is its velocity, p its pressure, s its entropy per unit mass and Φ the gravitational potential acting on it. Note that **eq. 1.16**, which expresses the conservation of entropy per unit mass, excludes any energy losses due to viscous or thermal dissipation effects. In fact, the fluid described by the system above is inviscid, which is the case of most astrophysical fluids (Shu 1992). The other equations are the continuity equation (**eq. 1.13**), the Euler equation (**eq. 1.14**) and the Poisson equation (**eq. 1.15**). These equations are satisfied by the static solution with ρ equal to a constant value $\rho_0 \neq 0$, $p = p_0$, $s = s_0$, $\nabla\Phi = 0$ and $\mathbf{v} = 0$. According to **eq. 1.15**, however, $\nabla\Phi = 0$ implies $\rho_0 = 0$, i.e. a null density distribution. Even if this static solution is technically unphysical, the resulting criterion is still a qualitatively correct tool to analyse perturbed systems. Consequentially, this problem can be set aside by arbitrarily assuming that **eq. 1.15** only applies to the density and gravitational potential perturbations $\delta\rho$ and $\delta\Phi$ (the so-called "Jeans swindle", e.g. Binney and Tremaine 2008).

These two perturbations, along with $\delta\mathbf{v}$, δp and δs, constitute a state with $\rho = \rho_0 + \delta\rho$, $\mathbf{v} = \delta\mathbf{v}$, $p = p_0 + \delta p$, $\Phi = \delta\Phi$ and $s = s_0 + \delta s$, and this state can be inserted in the above system of equations. In a linear framework, terms which depend more than linearly on the perturbations δ are discarded; this is a good approximation as long as the fluctuations represent small displacements from the unperturbed state. The solutions of the perturbed system of equations are assumed to be in the form of plane waves:

$$\delta u = \hat{\delta u} e^{(i\mathbf{k}\cdot\mathbf{x} - \omega t)} \,, \tag{1.17}$$

where u is a generic fluid property among those listed above, $\hat{\delta u}$ is the constant amplitude of the perturbation, $\mathbf{k}$ is the wavevector and ω the angular frequency. Performing the necessary calculations yields the dispersion relation

$$\omega^2 - c_s^2 k^2 + 4\pi G \rho_0 = 0 \,, \tag{1.18}$$

where $c_s^2 \equiv \delta p / \delta\rho$ is the sound speed. Having $\omega^2 < 0$ leads to instability and growth of the perturbations (i.e. there is a solution with $\mathrm{Im}(\omega) > 0$). This concept can be

expressed in terms of the Jeans wavelength λ_J, whose value follows from the dispersion relation through $\lambda^2 = 4\pi^2/k^2$ and $\omega^2 = 0$:

$$\lambda_J = c_s\left(\frac{\pi}{G\rho_0}\right)^{1/2} . \tag{1.19}$$

Perturbations with wavelength $\lambda < \lambda_J$ are stable and propagate as sound waves, whereas if $\lambda > \lambda_J$ instability arises and the amplitude of the perturbations changes with time; in the case of $\delta\rho$, this can lead to a gravitational collapse (in the limit $\lambda \gg \lambda_J$, the propagation speed of the perturbation is the free-fall time). A useful quantity derived from λ_J is the Jeans mass $M_J = \rho_0(4\lambda_J^3\pi/3)$, with $M > M_J$ leading to instability and $M < M_J$ leading to stability. Qualitatively speaking, the Jeans criterion can be explained by taking into account the competition of gravity and pressure forces: a fluid with a large mass has a strong self-gravity, which is harder to overcome by the pressure force with respect to a smaller mass.

The analysis of $M_J(t)$ yields an evolutionary picture of the perturbations that are stable or unstable at a given time, allowing to build dynamic models of structure formation. However, the description of gravitational instabilities carried on so far holds for collisional particles (i.e. baryonic matter) in a Newtonian framework; nonetheless, the Jeans length for dark matter is the same as that in **eq. 1.19**, except that c_s is replaced by an estimate v_* of the mean DM particle velocity.

To take into account the expansion of the Universe, it is appropriate to switch from phsyical coordinates $\mathbf{x}$ to the so-called comoving coordinates $\mathbf{x}_c$, related by $\mathbf{x_c} = a_0\mathbf{x}/a$ (where $a_0 = a(t_0)$). Generally, its harder for perturbations to grow in an expanding Universe with respect to the Newtonian case, because the expansion acts against the gravitational pull of overdense regions. Moreover, the different components of the early Universe (i.e. dark matter, baryonic matter and radiation[6]) need different dispersion relations that lead to different behaviours of $M_J(t)$. Therefore, tracing the evolution of the Jeans mass requires knowledge of the dominant component of the Universe in every stage of its evolution, knowledge which is obtained through a comparison of the time evolution of Ω_r, Ω_{DM} and Ω_b. This comparison yields a so-called equivalence redshift $z_{eq} \approx 3400$ (e.g. Planck Collaboration 2020), which separates the period when $\Omega_r > \Omega_m$ ($z > z_{eq}$, $t < t_{eq}$) from the one in which $\Omega_m > \Omega_r$ ($z < z_{eq}$, $t > t_{eq}$). An additional aspect to consider is that matter can be coupled to radiation if the interaction rate between the two is higher than H, i.e. the expansion rate of the Universe. While this is the case, independently of the dominant

[6] As mentioned earlier, the cosmological constant contribution to the total matter-energy density of the universe is negligible in the early stages of the evolution of our Universe; its crucial role in making the universe flat only arises in relatively recent cosmic times.

component, perturbations have an oscillatory behaviour (Padmanabhan 1993) and do not grow. In fact, baryons remain coupled to radiation until $z_{rec} \approx 1100$, when nuclei and electrons recombine and photons are free to reach the observer, constituting what is known as the CMB. From this point onwards, baryonic overdensities with $M > M_J$ can become unstable. Dark matter perturbations can begin to grow even earlier (because dark matter, if the model chosen for it allows weal interactions with the radiation field, decouples from it at $z_{dec,DM} > z_{eq}$) and by z_{rec} baryons can group up faster because the potential wells have already been deepened by DM instabilities (e.g. Blumenthal et al. 1984). Without this dark matter support, baryon perturbations alone would not be able to grow efficiently enough to collapse into the structures that we observe today, which is one of the main reasons why dark matter plays a major role in the standard cosmological model. Additionally, the early decoupling of dark matter raises the possibility that at $z_{dec,DM}$ the Universe was hot enough for DM particles to be relativistic (i.e. with speed $v \sim c$), yielding a case known as hot dark matter (HDM). On the contrary, cold dark matter (CDM) is already not relativistic at $z_{dec,DM}$, and leads to a different structure formation scenario.

1.2.2 Hot and cold dark matter

When studying the behaviour of $M_J(z)$ for hot dark matter, the only redshift intervals involved are $z > z_{dec,DM}$, $z_{dec,DM} > z > z_{eq}$ and $z < z_{eq}$. In this case, the evolutionary trend of the Jeans mass shows an increase with decreasing z, leading to a very high peak value $M_J(z_{eq}) \approx 10^{15}$ $M_\odot$, where $M_\odot$ is the solar mass (Padmanabhan 1993). On the other hand, the same analysis for cold dark matter has to take into account the additional redshift $z_{nr} > z_{dec,DM}$, when dark matter becomes non-relativistic while still being coupled to radiation. In the new time interval $z_{nr} > z > z_{dec,DM}$, M_J has a slower increasing trend with respect to HDM, which ultimately leads to $M_J(z_{eq}) \approx 10^6$ $M_\odot$. In both cases, after the equivalence the Jeans mass decreases as $a^{-3/2}$.

It is worth noting that while according to **subsec. 1.2.1** perturbations with $M < M_J$ oscillate and can technically grow if M_J becomes smaller than M, in reality the DM particles move with mean velocity v_* (the one to insert in the DM equation for λ_J) over a typical distance λ_{fs}, called free streaming length (e.g. Blumenthal et al. 1984). Perturbations with wavelength $\lambda < \lambda_{fs} < \lambda_J$ do not actually oscillate, but are smeared out and cancelled by DM particles escaping potential wells due to a sufficiently large λ_{fs}. As in the case of the Jeans mass, it is possible to define a free streaming mass M_{fs} from λ_{fs}, and for both HDM and CDM $M_J(z_{eq}) = M_{fs}(z_{eq})$.

This means that $M_J(z_{eq})$ sets a lower limit for the mass of overdensities which can collapse into structures after the equivalence. In the hot dark matter case this lower limit is very high, initially resulting in the formation of superclusters, with smaller structures such as clusters and galaxies being formed afterwards by fragmentation (top-down scenario). Conversely, cold dark matter allows the formation of smaller objects right after z_{eq}, and bigger structures can grow subsequently through gravitational merging (bottom-up scenario). Therefore, checking if the youngest structures are superclusters or smaller objects can help in understanding the type of DM in the Universe. In fact, galaxies are observed from $z \approx 12$ (Robertson et al. 2022), whereas the biggest structures are still undergoing gravitational collapse (Ryden 2017), thus excluding the hot dark matter case. Further constraints come from Lyα forests which are linked to the shape of DM halos, related in turn to the type of dark matter. These constraints indicate that a hypotetical dark matter particle should be more massive (i.e. colder) than 5.3 keV (Iršič et al. 2017). In fact, even though the standard cosmological model is usually referred to as ΛCDM model (because it includes a cosmological constant and cold dark matter), there are some discrepancies between models of cold dark matter and observational data which led to consider warm DM models (e.g. Bode et al. 2001) and even a mixed scenario including different types of dark matter.

1.3 Processes of galaxy formation

So far, the evolution of perturbations has been described in a linear framework, which holds as long as $\delta \equiv \delta\rho/\rho_0 \ll 1$. In this case, assuming a perturbation as a spherical overdense region, the sphere will actually keep expanding with the rest of the Universe, albeit slightly slower. Eventually, it will reach a maximum radius, after which the overdensity will start contracting due to its own gravitational pull and enter a non-linear evolutionary regime. This process will carry on until the gravitational force is balanced by other effetcs, notably the pressure force for baryons (yielding hydrostatic equilibrium) and the velocity dispersion for dark matter particles; when this happens, the overdensity is said to be in virial equilibrium (because it obeys the virial theorem, e.g. Ciotti 2021). If there were no other processes, the baryonic gas inside a DM virialised structure (a so-called halo) would not evolve further and stars and galaxies would not exist. Unlike dark matter, however, baryons are able to undergo further contraction by radiating away part of their energy through a process called radiative cooling, and eventually part of these baryons will collapse into stars and give birth to the gravitationally bound systems of gas, stars and dark

matter that we commonly call galaxies.

1.3.1 Gas cooling and star formation

The mechanisms behind gas cooling include a wide range of physical processes, which lead to different cooling efficiencies depending on the gas properties. For this reason, it is useful to model the gas in the dark matter halos of the early Universe. For the sake of simplicity, the gas can be assumed to have an isothermal, spherically simmetric distribution of radius R_{vir}, while the total mass of the halo and the gas is M_{vir}. The comparison of the gravitational potential energy and the internal energy of the gas through the virial theorem yields the following definition of virial temperature (e.g. Rees and Ostriker 1977), which provides a rough estimate of the gas temperature:

$$T_{vir} \simeq \frac{G m_H M_{vir}}{k R_{vir}} , \tag{1.20}$$

where m_H is the hydrogen mass and the approximation comes from the fact that an exact relation would require knowledge of the DM halo density profile, as well as of the chemical composition of the gas inside it. The relationship between the gas temperature and the virial mass M_{vir} is very important to understand which halos can form galaxies: if M_{vir} is too high, so is the virial temperature and cooling becomes inefficient.[7] Other than that, the temperature determines the ionization state of the gas components, which allows some cooling mechanisms over others. Specifically, $T \approx 10^4$ K is the ionization temperature of hydrogen (i.e. the most abundant element of the Universe), and separates the following main cooling regimes.

A gas with $T \gtrsim 10^4$ K can cool mainly through Bremsstrahlung (free-free), recombination (free-bound) and radiative de-excitation (bound-bound). The two latter mechanisms are closely related to collisional excitation and collisional ionization: after a collision, a bound electron acquires enough energy to move to a higher level or even free itself from an atom. These processes can fuel the free-bound and bound-bound cooling mechanisms, and in fact, one of the simplest assumptions to describe cooling is that recombination is entirely due to collisional ionization; in other words, the two phenomena balance each other. Other commonly employed conditions are an instantaneous radiative return to the ground state after an excitation and the absence of any external photoionising field. When all these are applied to a low-density

[7] Here there is the implicit assumption of the gas being in virial equilibrium with the DM halo when cooling starts to take place; in fact, the full picture is much more complicated than that, and the gas might very well enter the halo with $T < T_{vir}$ and skip the virial equilibrium phase (e.g. Birnboim and Dekel 2003; Kereš et al. 2005).

gas, they constitute the collisional ionization equilibrium assumption (CIE, e.g. Cox and Tucker 1969). If CIE holds, it is relatively easy to calculate the rate at which the gas loses its internal energy:

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}t} = -n_t n_e \Lambda(T) \,, \tag{1.21}$$

where ε is the internal energy density, n_t is the ions and neutral atoms number density, n_e is the electron number density and $\Lambda(T)$ is the so-called cooling function. In this temperature regime, $\Lambda(T)$ has a peak around $2 \cdot 10^4$ K due to the maximum efficiency of recombination and de-excitation of hydrogen. Smaller peaks at higher temperatures are due to other elements such as helium and, if present, metals.[8] Eventually, at very high temperatures ($T \gtrsim 10^7$ K), Bremsstrahlung remains the only contribution to $\Lambda(T)$.

If the temperature of a gas is $\lesssim 10^4$ K, cooling is generally much less efficient, because matter is mostly neutral and so the free-bound and free-free processes cannot take place. Moreover, there are not enough free electrons to make collisional excitation efficient. If the gas is metal-rich, there are additional transitions that can fuel cooling, but even so, the cooling function is usually not nearly as high as when $T \gtrsim 10^4$ K. In fact, the cold dark matter scenario leads to the smaller DM halos ($M_{vir} \lesssim 10^5$ $\mathrm{M}_\odot$) having $T_{vir} \lesssim 10^3$ K. The gas in these halos is believed to be the source of the stellar component of the oldest galaxies; however, no atomic cooling mechanism described so far is able to operate at these low temperatures, especially when considering the absence of metals in the primordial gas. This complication raises the question of how to cool the primordial gas, in order to make it contract below R_{vir} and form stars. The answer lies in molecular coolants: at $T \lesssim 10^3$ K, the most abundant molecule of the primordial gas is molecular hydrogen (H_2), which is able to provide efficient cooling in low density regimes up to a minimum temperature $T \approx 200$ K (Galli and Palla 2013). From this point onwards, gas compression slowly rises its temperature and a protostellar core is formed, which accretes matter from its surroundings and gives birth to an actual star. In fact, it is possible to calculate the expected accretion rate in order to have an estimate of the final mass of the star M_{star}. In the temperature range of the gas surrounding the core ($T \approx 300 - 1000$ K) protostellar cores can accrete gas with a rate of 10^{-3} $\mathrm{M}_\odot$/yr, yielding $M_{star} \approx 100$ $\mathrm{M}_\odot$. These very massive stars are also extremely hot, with a surface temperature of 10^5 K, which implies the emission of many ionizing photons. These are able to increase the ionization fraction of the surrounding gas, which in turn favors the formation chan-

[8] Metals, which in an astrophysical framework are generally elements heavier than helium, are actually found only in gas which has been chemically enriched by stars, and are absent in the primordial baryonic matter.

nels of H_2 and enables more efficient cooling, up to $T \approx 150$ K. Moreover, molecules with a permanent electric dipole such as hydrogen deuteride (HD) and LiH can further cool the gas up to $T \approx 30$ K (Galli and Palla 2013), a temperature which allows an easier gravitational collapse thanks to the low pressure support ($p \propto T$ in a perfect gas). In fact, this temperature is similar to the typical value of the modern sites of star formation, the so-called giant molecular clouds (GMCs). These are dense structures susceptible to Jeans instability and with a complex internal morphology, which includes the effects of turbulence and magnetic fields. As in the primordial Universe, the gravitational collapse leads to the formation of a protostar, but the lower medium temperature implies lower accretion rates and star masses with respect to the oldest objects.

1.3.2 Feedback processes

The galaxy components described so far, i.e. stars and the gaseous medium surrounding them, are obviously not independent of each other: gas instabilities can lead to the formation of stars, and in turn the evolution of stars influences the interstellar medium (ISM). The latter process includes a variety of phenomena which are collectively known as stellar feedback. First of all, stars form elements heavier than helium which can be returned to the ISM mainly through supernova explosions: core-collapse supernovae (i.e. SN types II, Ib and Ic) mark the death of stars with mass $\gtrsim 8$ $M_\odot$, which have relatively short lifespans and mainly eject alpha-elements (i.e. those which can be formed by aggregation of helium nuclei). On the other hand, remnants from much longer lived stars ($M_{star} \lesssim 1$ $M_\odot$) can undergo gravitational instability through mass accretion and give rise to a type Ia supernova (SN Ia), which destroys the remnant and injects Fe-group elements into the ISM (e.g. Matteucci and Greggio 1986). Generally, supernovae also transfer kinetic energy to the surrounding medium, which can lead to the removal of gas from the galaxy and to subsequent star formation quenching (negative feedback), but also to local gas compression, which increases its self gravity (positive feedback). Normally, the energy transfer efficiency of core-collapse supernovae is very low, but when coupled with stellar wind bubbles[9] they can lead to the removal of a fraction of gas from the ISM. This gas can either fall back to the ISM in a different location (galactic fountain) or remain in the outer regions of the galaxy (galactic wind). In the latter case, which

[9]Stellar wind bubbles form through mass ejection from very massive stars (spectral class O/B). The radiation pressure pushes away with high speed the outer layers of the star, creating a shock when this "wind" interacts with the ISM and a reverse shock in the wind itself. This heats up the wind and produces a "bubble", which sweeps the shocked ISM by expanding adiabatically (Cimatti et al. 2020).

tends to only take place in galaxies with a very high star formation rate (starburst galaxies), the ejected gas velocity can be high enough to let it escape the galactic potential well.

In fact, feedback processes also arise from the so-called active galactic nuclei (AGN), which are produced by the accretion of surrounding matter onto the supermassive black hole (SMBH) found at the center of most galaxies. Tipically, the infalling matter has a finite angular momentum which prevents it from going straight into the SMBH; instead, it arranges itself in a disk-like structure around the central object, thanks to the outwards transfer of angular momentum. This disk rotates differentially around the SMBH, i.e. with a radius-dependent angular velocity; this implies a viscous tension between adjacent "rings", causing a loss of angular momentum as matter spirals inwards. The resulting release of gravitational energy fuels the rotational kinetic energy and thermal energy of the disk, which then cools radiatively (Shakura and Sunyaev 1973). Specifically, this radiation represents the optical and UV emission of an AGN, composed of blackbodies at different temperatures, each of which corresponding to a specific ring of the disk. In fact, optical photons also fuel the AGN emission in other bands, namely X (e.g. Haardt and Maraschi 1991) and mid-infrared (e.g. Pier and Krolik 1992). It should be noted that the above description of an AGN disk holds as long as the disk is able to carry the thermal energy on its surface, where it can be radiated away. However, this is not always the case, as the infalling matter can retain most of its internal energy without being able to have an efficient radiative cooling. This advection-dominated accretion flow (ADAF) can lead to the formation of powerful relativistic outflows (i.e. "jets"), preferentially directed along the rotation axis of the disk (e.g. Narayan and Yi 1995) and hosting electrons which radiate in the radio band through synchrotron.

In terms of feedback, AGN with ADAF-like disks tend to operate in the so-called kinetic mode (or radio mode), with jets inflating hot bubbles in the galactic halo; one of the main effects of these bubbles is the halting of cooling flows[10] in galaxy clusters, because the bubbles provide a mechanical energy source capable of reheating the cooled gas (e.g. Tucker and David 1997). On the other hand, if the disk is radiatively efficient, the feedback is mainly provided by photons (quasar mode): radiation pressure creates high-velocity ($\sim 0.1c$, where c is the speed of light) winds in close proximity to the central AGN, which in turn sweep the surrounding ISM, driving a shock through it and generating a gas outflow. If the shocked material is

[10] According to **eq. 1.21**, a higher density environment yields more efficient energy losses through cooling; this is exactly what should happen in the dense inner regions of galaxy clusters, where cooled gas is pushed further inwards (creating a cooling flow) by the overlying medium (Fabian 1994).

able to cool efficiently (momentum-driven outflow), its impact on the ISM is very mild, and the outflow eventually stops close to the central nucleus (within ~ 1 *kpc*, Zubovas and King 2012). On the other hand, if the cooling efficiency of the shocked material is low (energy-driven outflow), the wind is eventually able to sweep gas far from the AGN and significantly reduce (possibly even to zero) the accretion rate. Moreover, the energy budget involved in this latter type of feedback can be high enough to make the ISM of the host galaxy escape its potential well, yielding the quenching of star formation (e.g. Zubovas and King 2012). In fact, energy-driven outflows are thought to be responsible for the absence of star formation in the most massive galaxies, where the energy released through the aforementioned supernova feedback is not high enough to sweep them clear of their gaseous content.

1.4 The numerical approach

The combination of the processes described above gives rise to the general structure and galaxy evolution theory, which makes us able to model and interpret the ever-growing amount of observational data available. However, due to the complexity and non-linearity of the involved phenomena, it is extremely challenging, and in most cases even impossible, to treat them in a purely theoretical fashion (e.g. Springel 2016; Vogelsberger et al. 2020). For this reason, the task of modeling and interpreting observations is usually carried out through cosmological simulations, which evolve over a certain time period a finite volume of the Universe, typically a cube with comoving side and periodic boundary conditions (to mimic the homogeneity and isotropy set by the Cosmological Principle). The main components to include in a cosmological simulation are, first and foremost, dark matter and dark energy, since the former drives the formation of structures (as seen in **section 1.2**) and the latter accelerates the expansion of the Universe[11]. In fact, many simulations, such as the Millennium-XXL run (Angulo et al. 2012), the Dark Sky Simulations (Skillman et al. 2014), the DEUS Full Universe run (Alimi et al. 2012), the Bolshoi simulation (Klypin et al. 2011), the Phoenix runs (Gao et al. 2012) and the Via Lactea simulation (Diemand et al. 2008) only employ these components, because their goal is to analyse the large scale structure and the morphology and distribution of DM halos. In this framework, the only force to treat in a simulation is gravity; in fact, the theory adopted to describe this force is usually the Newtonian one instead of general relativity, because in a linear approximation the two give the same results, and the

[11] In fact, dark energy is not directly treated in a dynamical evolution framework and only arises in a simulation through its effects on $H(z)$.

velocities involved in the non-linear regime are much lower than the speed of light (Vogelsberger et al. 2020). Other projects like the IllustrisTNG simulations (Springel et al. 2018), the Auriga suite (Grand et al. 2017), the EAGLE project (Schaye et al. 2015), the Horizon-AGN simulation (Dubois et al. 2014), the LATTE-FIRE run (Wetzel et al. 2016) and the massiveblack-II simulation (Khandai et al. 2015) also take into account baryons and their complex physics, including for example cooling, magnetic fields, models of star formation, feedback by both stars and AGN and chemical enrichment of the gas.

In order to adapt the gravitational and hydrodynamic models to numerical simulations, it is necessary to discretize the equations and quantities involved in those models. For example, a continuous density field $\rho(\mathbf{x})$ can be sampled with a set of N discrete point masses; clearly, the larger is N, the better the particles sample the field (i.e. the higher the simulation resolution) and the more accurately the physical processes involving that field are described. In fact, both dark and baryonic matter are actually made of physical discrete particles, but following each of them through a simulation would be an impossible task to handle for modern computers, because of the incredibly large N.[12] Normally, the particles followed in a simulation are far less than the physical ones and thus much more massive, with N ranging from a few millions to even trillions in the biggest projects. With this "tracer particles" approach, simulations can actually be performed with a reasonable amount of computational resources, both during the run (e.g. in terms of run time and memory) and after it (e.g. in terms of storage space) and yield results which make us able to interpret and make predictions on observational data. However, there are still hardware limitations which define an upper bound for the number of tracer particles, which otherwise would always be high enough to achieve the desired resolution. As observational campaigns cover increasingly large regions of the Universe with high accuracy, this upper bound becomes more and more incovenient, because the simulation quality and/or volume coverage tend to be too low for a comparison with real data. This is especially true in the case of simulations of galaxy formation, because baryon physics takes a much higher toll on computational resources with respect to gravity-only calculations. Specifically, if modern galaxy formation simulations have very high resolutions, they usually cover a small volume of the Universe or even a single DM halo; conversely, simulations which evolve large volumes of the Universe are necessarily resolution limited. Therefore, this work aims to develop an algorithm which can ease the computational burden of cosmological simulations, in order to

[12] Considering N as the number of nucleons (i.e. protons and neutrons) and according to Avogadro's number, for a single gram of matter $N \approx 10^{23}$.

take a step further towards the possibility to simulate large-scale galaxy formation with a high resolution. Before trying to understand how this algorithm works, it is necessary to briefly introduce some numerical techniques already employed in gravity and hydrodynamic calculations to satisfy the need for a large number of tracer particles.

2 | Numerical techniques

The importance of cosmological simulations has been briefly explained in the previous chapter, which also highlighted the inability to simulate the entirety of the particles which constitute the matter of our Universe, relying instead on macro particles which trace the underlying physical ones. Actually, this is not the only approximation usually employed in modern simulation codes, because further techniques are used to increase the number of simulated particles while keeping the computational cost in check. In fact, there is a large variety of different methods to achieve this result, each with its strenghts and weaknesses and some more suited to particular types of simulations (or even specific stages of a simulation) than others; the ones presented in this chapter will be the starting point to achieve a better performance in DM-only and potentially galaxy formation simulations. Techniques which address the gravitational force calculations are presented in **section 2.1**, whereas hydrodynamic ones are described in **section 2.2**.

2.1 N-body solvers

When N particles or objects in general interact gravitationally, the calculations of their individual trajectories constitute a so-called N-body problem. This problem is analytically solved through an integration of the following Newtonian equations of motion, which hold for system of N point-like particles labeled with $i = 1, \dots, N$:

$$\ddot{\mathbf{x}}_i = -\nabla_i \Phi(\mathbf{x}_i) \,, \tag{2.1}$$

$$\Phi(\mathbf{x}) = -G \sum_{j=1}^{N} \frac{m_j}{\sqrt{(\mathbf{x} - \mathbf{x}_j)^2 + \varepsilon^2}} \,, \tag{2.2}$$

where $\mathbf{x}_i = (x_{1i}, x_{2i}, x_{3i})$ is the position vector of each particle, m_i is their mass and ∇_i is the operator $(\partial/\partial x_{1i}, \partial/\partial x_{2i}, \partial/\partial x_{3i})$. The quantity ε is called "softening length" and its main effect is to dampen the gravitational potential $\Phi(\mathbf{x})$ at short distances, in order to ensure the collisionless behaviour of the system (e.g. Springel 2016). In fact,

the only case in which the problem allows an analytical integration of the equations above is when $N = 2$ (e.g. Goldstein et al. 2008). For larger N, which is obviously the case of cosmological simulations, the solution cannot be calculated with analytic techniques. However, it is still technically possible to perform an exact calculation of the gravitational force acting on each particle through the formula

$$\ddot{\mathbf{x}}_i = -G \sum_{j=1}^{N} \frac{m_j}{[(\mathbf{x}_i - \mathbf{x}_j)^2 + \varepsilon^2]^{3/2}} (\mathbf{x}_i - \mathbf{x}_j) \,, \tag{2.3}$$

which is a sum with $N-1$ addends for each of the N particles, yielding a total number of operations which scales with N as $\mathscr{O}(N^2)$ (equivalently, it can be said that the direct summation has a scaling of $\mathscr{O}(N^2)$). When trying to carry on this summation in a simulation framework, this scaling translates in a number of operations to be performed which quickly becomes prohibitive for large N. The aforementioned additional techniques, usually applied on top of the tracer particles approximation, provide a scaling improvement at the cost of approximate gravitational force calculations. Among these techniques, the most relevant for this work are the particle-mesh (PM) method and especially the hierarchical multipole ("tree") method; in fact, a combination of the two is often employed in modern simulation softwares, yielding the so-called "tree-PM" algorithm.

2.1.1 Particle-mesh method

As the name suggests, the particle-mesh method involves the construction of a mesh over the simulation domain, in order to obtain a density field ρ to be inserted in the Poisson equation (**eq. 1.15**), which is then solved for the gravitational potential Φ. There are various methods to obtain an estimate of Φ, one of which takes advantage of the properties of linear differential operators, specifically of the concept of Green functions (e.g. Ciotti 2021). Given a linear operator $\mathscr{L}$, a Green function provides a way to solve the general differential equation

$$\mathscr{L}[f(\mathbf{x})] = h(\mathbf{x}) \,, \tag{2.4}$$

where $h(\mathbf{x})$ is a given function of the Cartesian coordinate $\mathbf{x}$. A function $g(\mathbf{x}, \mathbf{y})$ is called a Green function for the operator $\mathscr{L}$ if the application of $\mathscr{L}$ to g yields the Dirac delta:

$$\mathscr{L}[g(\mathbf{x}, \mathbf{y})] = \delta(\mathbf{x} - \mathbf{y}) \,. \tag{2.5}$$

At this point, solving **eq. 2.4** for f can be reduced to the following integration problem:

$$f(\mathbf{x}) = \int_{\mathbb{R}^n} h(\mathbf{y}) g(\mathbf{x}, \mathbf{y}) \mathrm{d}^n \mathbf{y} \,. \tag{2.6}$$

When the concept of Green functions is applied to **eq. 1.15**, the above equation can be further simplified to a convolution between g (now a function of $\mathbf{x}-\mathbf{y}$) and $h(\mathbf{x}) = 4\pi G\rho(\mathbf{x})$:

$$\Phi(\mathbf{x}) = 4\pi G \int_{\mathbb{R}^3} \rho(\mathbf{y}) g(\mathbf{x}-\mathbf{y}) \mathrm{d}^3\mathbf{y} \,. \tag{2.7}$$

By the convolution theorem, this implies that the Fourier transform $\mathscr{F}$ of Φ is equal to the product between the Fourier transforms of ρ and g. In fact, in the periodic box with side L commonly employed in cosmological simulations, $\rho(\mathbf{x})$ is a periodic function and can be expressed as a Fourier series:

$$\rho(\mathbf{x}) = \sum_{\mathbf{k}} \rho_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{x}} \quad \text{with} \quad \rho_{\mathbf{k}} = \frac{1}{L^3} \int_{L^3} \rho(\mathbf{x}) e^{-i\mathbf{k}\cdot\mathbf{x}} \mathrm{d}\mathbf{x} \,, \tag{2.8}$$

where the quantities $\rho_{\mathbf{k}}$ are the Fourier coefficients, or modes, of the series. In this framework, **eq. 1.15** is solved by computing and multiplying each mode by the Green function in Fourier space $g_{\mathbf{k}}$ to yield the coefficients $\Phi_{\mathbf{k}}$, which can be used to finally calculate $\Phi(\mathbf{x})$ through its Fourier series. The reduction of **eq. 2.7** to simple products, coupled with the ease and accuracy provided by modern Fourier transform algorithms[1], has made this solution of the Poisson equation one of the most widely employed in cosmological simulations.

The Poisson equation, however, requires a density field, whereas the building blocks of cosmological simulation are tracer particles. Therefore, the first task that the particle-mesh method has to accomplish is the construction of a density field: particles inside each cubic mesh cell contribute to the total mass of the cell, which in turn provides a local density estimate when divided by the cell volume. The exact contribution of each particle can be evaluated by defining a so-called shape function $S(\mathbf{x})$; if each cell center, or grid point, is indexed by $\mathbf{p} = \{p_x, p_y, p_z\}$ and the linear size of a cell is h, then the fraction $W_{\mathbf{p}}(\mathbf{x}_i)$ of mass assigned by the particle i to the cell $\mathbf{p}$ is

$$W_{\mathbf{p}}(\mathbf{x}_i) = \int_{\mathbf{x}_\mathbf{p}-h/2}^{\mathbf{x}_\mathbf{p}+h/2} S(\mathbf{x}_i - \mathbf{x}) \mathrm{d}^3\mathbf{x} \,, \tag{2.9}$$

assuming that $S(\mathbf{x})$ is normalized to unity. Consequently, the density estimate for each cell will be

$$\rho_{\mathbf{p}} = \frac{1}{h^3} \sum_{i=1}^{N} m_i W_{\mathbf{p}}(\mathbf{x}_i) \,. \tag{2.10}$$

[1] Similarly to the gravitational force computation, calculating the Fourier transform of a discretized field of N elements through a direct summation has a scaling $\mathscr{O}(N^2)$ (e.g. Springel 2016). Nowadays, numerical Fourier transforms are calculated through the fast Fourier transform (FFT) algorithm (Cooley and Tukey 1965), which provides both a better scaling (i.e. $\mathscr{O}(N \ln N)$) and a higher accuracy with respect to direct summation.

As previously mentioned, the contribution of each particle to each cell, and thus the specific value of $\rho_{\mathbf{p}}$, is influenced by the choice made for the shape function: common examples are the Dirac delta, which leads to the nearest grid point (NGP) assignment, and a cubic or triangular "cloud", which leads to the clouds-in-cell (CIC) and the triangular shaped clouds (TSC) assignments, respectively.

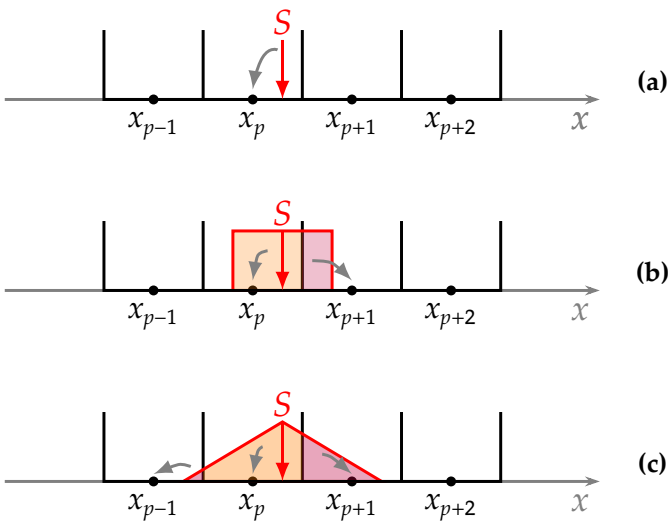


Figure 2.1: one-dimensional representation of three mass assignment schemes for the same particle position (red arrow): the NGP **(a)**, the CIC **(b)** and the TSC **(c)** (figures adapted from Springel 2016).

If $S(\mathbf{x}) = \delta(\mathbf{x})$, the fraction $W_{\mathbf{p}}(\mathbf{x}_i)$ can only be equal to 1 or 0, depending on the position of particle i: if $\mathbf{x}_i \in [\mathbf{x_p} - h/2, \mathbf{x_p} + h/2]$, all of its mass m_i is assigned to cell $\mathbf{p}$, otherwise m_i does not contribute at all to $\rho_{\mathbf{p}}$. In other words, the mass of each particle is assigned to the closest cell center with respect to the position of the particle, as shown in **fig. 2.1a**.

In the CIC scheme the shape function is the following:

$$S(\mathbf{x}) = \frac{1}{h^3}\Pi\left(\frac{\mathbf{x}}{h}\right), \tag{2.11}$$

where $\Pi(\mathbf{x})$ is the top-hat function

$$\Pi(\mathbf{x}) = \begin{cases} 1 & \text{if } |\mathbf{x}| \leq 1/2 \\ 0 & \text{if } |\mathbf{x}| > 1/2 \end{cases} . \tag{2.12}$$

In this case, depicted in **fig. 2.1b**, $W_{\mathbf{p}}(\mathbf{x}_i)$ will be $\neq 0$ only when the cubes with side h centered on $\mathbf{x}_i$ and $\mathbf{x_p}$ overlap, and the fraction of mass given to each cell will be proportional to that overlap. Usually, the total mass m_i will be split between two cells in $1D$, four in $2D$ and eight in $3D$; in fact, there is a general rule stating that the number of cells with a non-zero fraction of m_i scales with the number of dimensions d as 2^d. Moreover, as opposed to the previous scheme, this density estimate is piecewise linear and continuous, but its first derivative is not.

The derivative can be made continuous through the TSC scheme (**fig. 2.1c**), whose $2D$ geometrical interpretation is a triangle with base $2h$; the mass assignment is carried on similarly to the CIC scheme, namely by considering which mesh cells overlap

with the shape function centered on $\mathbf{x}_i$. In this case, the number of cells involved in the mass assignment scales with the dimensions as 3^d. Practically speaking, both the CIC and the TSC schemes are good choices for the particle-mesh method, as they retain a certain degree of information about the position of particles inside a cell.

Once the density field has been created and the Poisson equation has been solved, the gravitational accelerations need to be calculated from the potential $\Phi_{\mathbf{p}}$, where the subscript refers to the fact that the density field, and thus the potential, has been calculated at the mesh cell centers. The discretization of **eq. 2.1** which is usually employed is a simple finite differencing scheme, with truncation errors of $\mathscr{O}(h^4)$ or $\mathscr{O}(h^5)$. The resulting values $\ddot{\mathbf{x}}_{\mathbf{p}}$ can be finally used to calculate the gravitational forces $\mathbf{F}_i$ of each particle i by interpolating accelerations from the cell centers. It is mandatory to use the same function $W_{\mathbf{p}}(\mathbf{x})$ employed in the mass assignment stage, to ensure a vanishing self-force and antisymmetric forces between particle pairs: if these are not guaranteed, a particle might start moving by itself, violating momentum conservation. With this requirement, the force $\mathbf{F}_i$ acting on particle i is

$$\mathbf{F}_i = m_i \sum_{\mathbf{p}} \ddot{\mathbf{x}}_{\mathbf{p}} W_{\mathbf{p}}(\mathbf{x}_i) \,. \tag{2.13}$$

The main advantage of the particle-mesh method is its scaling of $\mathscr{O}(N)$ (plus the cost of the Fourier transform[2]), but its biggest limit is represented by the mesh itself, which sets a fixed spatial resolution given by h: in a cosmological simulation, structure formation evolves hierarchically over a large dynamic range, for which the smallest scales of interest might be unresolved by an excessively coarse mesh.

2.1.2 Hierarchical multipole method

The hierarchical multipole method (Barnes and Hut 1986) aims to speed up the direct summation of gravitational partial forces (**eq. 2.3**) by approximating groups of distant particles through their multipole expansion. This expansion follows from the Taylor series of the gravitational potential Φ generated in a point $\mathbf{x}$ by a group of N particles:

$$\Phi(\mathbf{x}) = -G \sum_{i=1}^{N} \frac{m_i}{|\mathbf{x} - \mathbf{x}_i|} = -G \sum_{i=1}^{N} \frac{m_i}{|\mathbf{x} - \mathbf{s} + \mathbf{s} - \mathbf{x}_i|} \,, \tag{2.14}$$

where the last expression introduces the center of mass

$$\mathbf{s} = \frac{\sum_{i=1}^{N} m_i \mathbf{x}_i}{\sum_{i=1}^{N} m_i} \,, \tag{2.15}$$

[2]See **footnote 1** for the scaling of the popular FFT algorithm.

useful to expand Φ under the condition $|\mathbf{x}_i - \mathbf{s}| \ll |\mathbf{x} - \mathbf{s}|$; this basically implies that the angle θ under which the N particles are seen from the position of the particle located in $\mathbf{x}$ is small, as seen in **fig. 2.2**. After defining $\mathbf{y} \equiv \mathbf{x} - \mathbf{s}$ the Taylor expansion can be written as

$$\frac{1}{|\mathbf{y} + \mathbf{s} - \mathbf{x}_i|} = \frac{1}{|\mathbf{y}|} + \frac{\mathbf{y} \cdot (\mathbf{s} - \mathbf{x}_i)}{|\mathbf{y}|^3} + \frac{\mathbf{y}^T[3(\mathbf{s} - \mathbf{x}_i)(\mathbf{s} - \mathbf{x}_i)^T - (\mathbf{s} - \mathbf{x}_i)^2]\mathbf{y}}{2|\mathbf{y}|^5} + \dots \,. \tag{2.16}$$

This result can be inserted in **eq. 2.14** to yield various multipole moments. The first term gives rise to the monopole moment M; the second term (i.e. the dipole moment) vanishes because the Taylor expansion has been calculated with respect to the center of mass $\mathbf{s}$, and the third term becomes the quadrupole moment:

$$\text{monopole: } M = \sum_{i=1}^{N} m_i \,, \tag{2.17}$$

$$\text{dipole: } \sum_{i=1}^{N} m_i(\mathbf{s} - x_i) = (M\mathbf{s} - M\mathbf{s}) = 0 \tag{2.18}$$

$$\text{quadrupole: } Q_{ij} = \sum_{k=1}^{N} m_k[3(\mathbf{s} - \mathbf{x}_k)_i(\mathbf{s} - \mathbf{x}_k)_j - \delta_{ij}(\mathbf{s} - \mathbf{x}_k)^2] \,. \tag{2.19}$$

These moments can now yield an approximate estimate of the gravitational potential

$$\Phi(\mathbf{x}) \simeq -G\left(\frac{M}{|\mathbf{x} - \mathbf{s}|} + \frac{\mathbf{y}^T \mathbf{Q} \mathbf{y}}{2|\mathbf{x} - \mathbf{s}|^5}\right), \tag{2.20}$$

from which the gravitational acceleration can be obtained through **eq. 2.1**.

In order to effectively calculate forces with the hierarchical multipole method, it is necessary to define suitable groups of particles to calculate moments from, and criteria to decide when the gravitational effects of these groups can be approximated (in fact, one criterion has already been mentioned: a small opening angle θ).

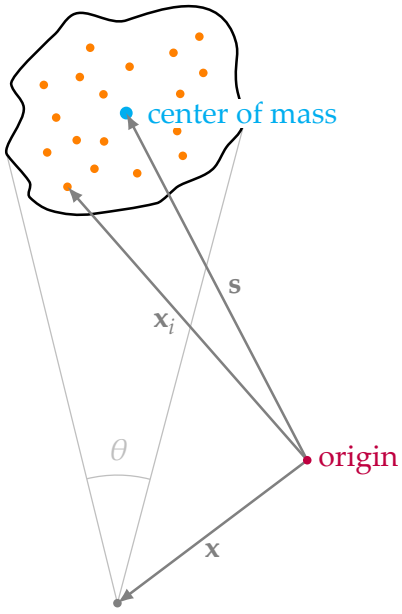


Figure 2.2: sketch showing a particle distribution far enough from the point $\mathbf{x}$ to yield a small opening angle θ; the combined gravitational effects of the particles are approximated through a multipole expansion at their center of mass (figure adapted from Springel 2016).

Among the many ways to accomplish the first of these two tasks, the oct-tree algorithm (Barnes and Hut 1986, hence the alternate name of this N-body solver) is one of the most popular: the simulation domain, that is, a cube with side L which contains every simulation particle, represents the so-called "root node", i.e. the first tree level. This cube is divided into 8 cubic subdomains with side $L/2$, which constitute the nodes of the second tree level. Each of these nodes is further divided into 8 smaller nodes with side $L/4$, creating the third tree level; this node splitting process continues recursively until each node in the lower levels contains at most one simulation particle, as shown in **fig. 2.3**.

To calculate the force acting on a particle $\mathbf{x}_i$, a so-called "tree walk" is performed: starting from the root node (i.e. the whole box), each node is tested with an opening criterion; if the criterion is satisfied, the multipole moments of the node can be employed right away in the force calculation, and the walk along that "branch" (i.e. the set of lower level objects starting from a node) stops. In this case, the tree walk continues to "siblings" (i.e. the nodes on the same level), without the need to open the node and analyse all of its eight "children" in the next tree level. This latter task, however, must be accomplished when a node does not satisfy the opening criterion: in this case, the multipole expansion would be too rough of an approximation, and force calculations must be refined by carrying on the tree walk to lower levels and children nodes.

The criteria commonly employed in this framework include the aforementioned geometrical criterion, which can be formalised as follows:

$$\frac{L}{|\mathbf{s} - \mathbf{x}_i|} \leq \theta_{geom} \,, \tag{2.21}$$

where L is the linear size of the node tested with the criterion and θ_{geom} is a free parameter which sets a tolerance level for the force accuracy (if θ_{geom} was set to 0, the force calculation would be carried on by direct summation). If a node satisfies the above relation, there is no need to open it and its multipole moments can be used. Another criterion widely employed is the so-called relative or dynamical criterion (Springel 2005), which aims to limit the maximum force error introduced by the tree method. This is achieved through a comparison of a rough estimate of the force error per unit mass (obtained from the truncation error of **eq. 2.16**) with a fraction of the expected force value, given by the free parameter α times the acceleration $|\mathbf{a}_{i,old}|$ which particle i had in the previous simulation timestep. The truncation error ΔF_{node} of the force per unit mass exerted on $\mathbf{x}_i$ by a node can be written as

$$\Delta F_{node} \sim \frac{G m_{node}}{|\mathbf{s} - \mathbf{x}_i|^2} \theta^2 \,, \tag{2.22}$$

where m_{node} is the mass of the node and $\theta = L/|\mathbf{s} - \mathbf{x}_i|$ is the angle depicted in **fig. 2.2**. Given the above error, the relative criterion is the following:

$$\frac{Gm_{node}}{|\mathbf{s} - \mathbf{x}_i|^2}\theta^2 \leq \alpha|\mathbf{a}_{i,old}| \,. \quad (2.23)$$

If a node satisfies this criterion, the force error introduced by its multipole moments is small enough to safely use that node without opening it. Typically, most simulation softwares offer the possibility to choose a geometrical opening criterion or a relative one.

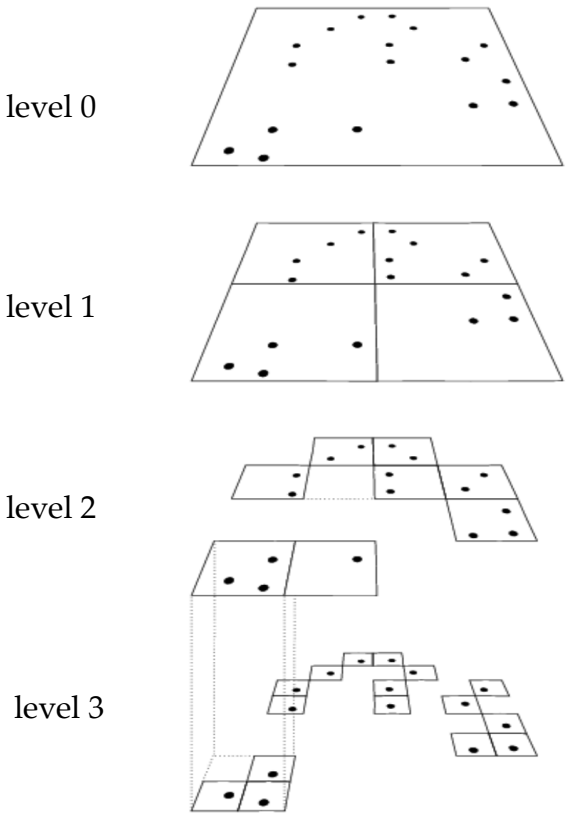


Figure 2.3: two-dimensional sketch of the oct-tree algorithm (a quad-tree in 2D). Note the node in level 2 which already contains only one particle, and is not further divided. Moreover, empty nodes do not need to be stored (figure taken from Springel 2016).

It can be shown (e.g. Barnes and Hut 1986; Springel 2016) that the computational cost of gravitational force calculations through the tree method scales as $\mathcal{O}(N \ln N)$, which is a great improvement over the direct summation scaling. This method is also able to automatically adjust to the dynamic scales of the simulation, because more smaller nodes are created as particles cluster. The main disadvantage of the tree method, however, is its application to almost homogeneous matter distributions (such as the universe at high redshifts): in this case, the very small force felt by a particle is the sum of many partial contributions from other particles and nodes, so obtaining accurate results becomes computationally expensive.

2.1.3 Tree-PM method

Interestingly enough, the situations in which the PM method does not work optimally, for example when particles cluster below the mesh cell size, are well handled by the tree method and vice versa, since the PM method probably represents the fastest approach to obtain accurate results for a highly homogeneous field. For this

reason, it is common practice to use a combination of the two algorithms in cosmological simulations, which calculate the long-range gravitational force through the PM method and the short-range one through the tree method. This force splitting starts with the Fourier series of the gravitational potential, whose modes $\Phi_{\mathbf{k}}$ are divided as follows:

$$\text{short-range component: } \Phi_{\mathbf{k}}^{short} = \Phi_{\mathbf{k}}[1 - \exp(-\mathbf{k}^2 r_s^2)] \,, \tag{2.24}$$

$$\text{long-range component: } \Phi_{\mathbf{k}}^{long} = \Phi_{\mathbf{k}} \exp(-\mathbf{k}^2 r_s^2) \,, \tag{2.25}$$

where r_s is the so-called force splitting scale. The long-range potential can be handled in Fourier space through the PM algorithm, whereas $\Phi_{\mathbf{k}}^{short}$ needs to be brought back to real space right away. For a single particle with mass m in a cubic periodic box with side L, and for $r_s \ll L$, the short-range potential is:

$$\Phi^{short}(\mathbf{x}) = -G\frac{m}{r}\mathrm{erfc}\left(\frac{r}{2r_s}\right), \tag{2.26}$$

where $r = \min(|\mathbf{x} - \mathbf{r} - \mathbf{n}L|)$ represents the smallest distance which separates the particle at $\mathbf{r}$ from the point $\mathbf{x}$ where Φ^{short} is calculated, among all of the periodic images of the box ($\mathbf{n} \in \mathbb{Z}^3$). Aside from the quantity r, which takes care of the periodic boundary conditions, Φ^{short} is simply the newtonian potential of a point mass multiplied by the error function

$$\mathrm{erfc}(x) = 1 - \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2} \mathrm{d}t \,, \tag{2.27}$$

which dampens the potential when r becomes similar to r_s (at $r \sim 5r_s$, Φ^{short} is negligible). The short range potential in real space is processed by the tree algorithm, restricted to nodes closer than $5r_s$. This leads to faster computation times with respect to an approach which covers the whole box.

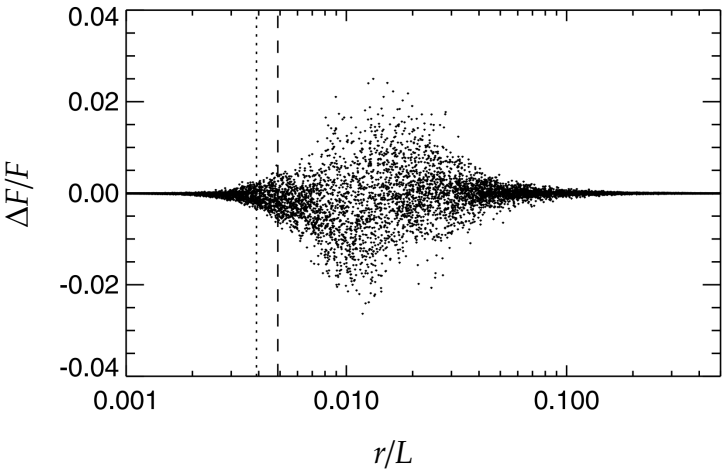


Figure 2.4: errors introduced by the particle-mesh method for the force due to a single particle. The dotted line represents the mesh size, the dashed line the force splitting scale (figure taken from Springel 2005).

The long-range force introduces an error which is due to the mesh anisotropy and is most prominent on scales similar to the cell size (Springel 2005). Nevertheless, this anisotropy can be arbitrarily reduced by increasing the splitting scale r_s, even

though in that case the tree algorithm would have to work on larger portions of the simulation domain to calculate the short-range force. As shown in **fig. 2.4**, setting r_s to a value larger than the cell size leads to errors not greater than 2%, and the rms force error is even smaller, usually sitting below 1%; a typical sweet spot for r_s to minimise errors is 4.5 times the cell size (i.e. the default value of the `AREPO` code, Weinberger et al. 2020).

2.2 Hydrodynamics

The system of equations which describes the behaviour of an inviscid fluid is the aforementioned system of **eqs. 1.13**, **1.14** and **1.16**. In fact, solving the system formed by these equations requires an additional equation, i.e. a closure relation, represented by an equation of state which links the pressure with the other variables; for ideal fluids, the equation is

$$P = (\gamma - 1)\rho\epsilon \,, \tag{2.28}$$

where γ is the adiabatic index.

The hydrodynamic system of equations can be analysed in two distinct ways: the so-called Eulerian approach, which aims to describe a fluid through its properties in fixed points of space, and the Lagrangian approach, which instead follows the evolution of each "fluid element" as it moves with the flow. The two approaches also lead to equivalent but distinct ways of writing the fluid equations: the Eulerian formulation is that of **eqs. 1.13**, **1.14** and **1.16**, whereas the Lagrangian formulation of the same equations is the following:

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} + \rho\nabla\cdot\mathbf{v} = 0 \,, \tag{2.29}$$

$$\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t} = -\frac{\nabla P}{\rho} - \nabla\Phi \,, \tag{2.30}$$

$$\frac{\mathrm{D}s}{\mathrm{D}t} = 0 \,, \tag{2.31}$$

where the operator $\mathrm{D}/\mathrm{D}t \equiv \partial/\partial t + \mathbf{v}\cdot\nabla$ is the Lagrangian derivative, which adds a component to the usual time derivative to take into account the motion of the fluid. The distinction between the two approaches persists in the numerical methods employed to solve the system of equations. Specifically, this work will focus on Lagrangian algorithms like the smoothed particle hydrodynamics (SPH) and hybrid approaches such as moving mesh schemes[3], as they are employed in the numerical codes which will be introduced later on.

[3] As will be seen in **subsec. 2.2.2**, a moving mesh does indeed follow the fluid in its motion, but

2.2.1 Smoothed particle hydrodynamics

The smoothed particle hydrodynamics approach is based on a set of fluid particles, which makes it well suited to the tracer particles formalism of cosmological simulations. These particles are used to construct a density field, which in turn is employed in the SPH equations of motion. Unlike the case of the particle-mesh method described in **subsec. 2.1.1**, the SPH algorithm operates in a mesh-free fashion: for a given field $F(\mathbf{x})$, it is possible to obtain a smoothed version $F_s(\mathbf{x})$ through the following operation:

$$F_s(\mathbf{x}) = \int F(\mathbf{x}')W(\mathbf{x}-\mathbf{x}', h)\mathrm{d}^3\mathbf{x}' \,, \tag{2.32}$$

where $W(\mathbf{x}, h)$ is the interpolation kernel and h is its width. Even though kernels like the Gaussian distribution can technically be employed, it is usually preferred to adopt functions with a finite support[4], such as a cubic spline (**fig. 2.5**) which is second order accurate if the particles are regularly distributed. If the starting field F is only known at the particle positions, it is possible to approximate **eq. 2.32** through a summation over all particles:

$$F_s(\mathbf{x}) \simeq \sum_i \frac{m_i}{\rho(\mathbf{x}_i)} F(\mathbf{x}_i)W(\mathbf{x}-\mathbf{x_i}, h) \,, \tag{2.33}$$

where the ratio $m_i/\rho(\mathbf{x}_i)$ represents the volume element assigned to each particle. The formula for $\rho(\mathbf{x})$ can be readily obtained from the above equation by setting $F = \rho$:

$$\rho_s(\mathbf{x}) \simeq \sum_i m_i W(\mathbf{x}-\mathbf{x}_i, h) \,. \tag{2.34}$$

It should be noted that the kernel has to be wide enough to include a significant number N_{ngb} of neighboring particles for an accurate estimate of the underlying field.[5] Actually, most practical applications of the SPH algorithm use a position-dependent kernel width, which takes care of variations in the neighboring particle density and tries to keep N_{ngb} approximately constant. This width can be either a function of $\mathbf{x}$ ("scatter" approach) or $\mathbf{x}_i$ ("gather" approach): while the former leads to a density field whose integral returns the total mass of the system, the latter is gen-

the method cannot be considered entirely Lagrangian as mesh cells can exchange mass (so there are no fixed fluid elements to follow). In fact, each mesh cell can be considered as a site where fluid properties are updated through exchanges with neighboring cells (in an Eulerian fashion), but the cells are not fixed in space, which is the Lagrangian aspect of this method.

[4]The support of a function f with real values is the closed subset of the domain where the function has a non-zero value. When $f \neq 0$ for a finite set of values, the function is said to have a finite support.

[5]For example, a 1D particle distribution with equal spacing d can lead to a second order accurate field estimate if $h = d$ is chosen as the kernel width (Springel 2016).

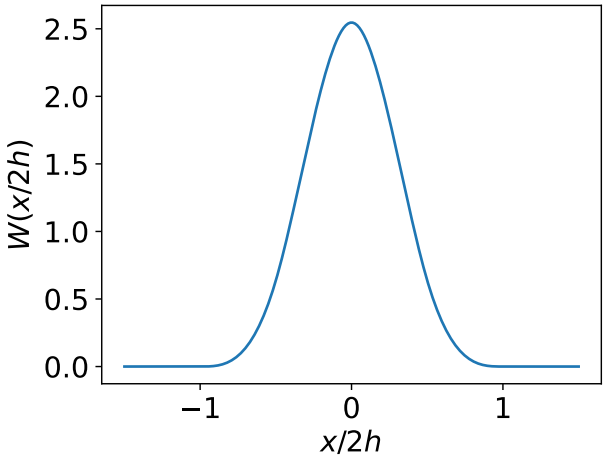


Figure 2.5: one-dimensional example of cubic spline kernel, usually a function of $x/(2h)$. Note that $W[x/(2h)] \neq 0$ only in the finite interval between –1 and 1 (figure adapted from Springel 2016).

erally preferred because it requires to define h only at the particle positions. Moreover, the summation depicted in **eq. 2.34** is not performed over every simulation particle, provided that W has finite support: in this case, only the N_{ngb} closest particles of particle $\mathbf{x}$ have $W(\mathbf{x} - \mathbf{x}_i) \neq 0$. Therefore, the computational cost of the whole field calculation scales as $\mathscr{O}(N_{ngb}N)$, since it is necessary to calculate ρ only at the position of each of the N total particles (i.e. $\mathbf{x} = \mathbf{x}_j,\ j = 1, 2, \dots, N$). If the kernel did not have a finite support, calculating $\rho(\mathbf{x}_j)$ would require a sum with N terms, yielding a $\mathscr{O}(N^2)$ total scaling.

It should be noted that **eqs. 1.13**, **1.14** and **1.16** derive from a known Lagrangian. Therefore, after obtaining an estimate for ρ, it is possible to apply the Euler-Lagrange equations to this Lagrangian to obtain the equations of motion; these are a set of relatively simple ordinary differential equations which substitute the entire system of the Euler partial differential equations. Moreover, **eqs. 1.13** and **1.16** are automatically satisfied by the SPH method, which conserves mass and specific entropy.

The specific entropy, however, will vary when dealing with shocks, in which the fluid properties jump by finite amounts. In a shock front, the differential form of the Euler equations no longer holds, but the integral form (i.e. the so-called weak formulation) still does, and leads to the Rankine-Hugoniot jump conditions (Rankine 1870). These conditions imply that the specific entropy of a fluid always increases when passing through a shock, therefore the inviscid fluid assumption does not hold at the shock front, where kinetic energy is turned into heat. A possible way to treat this complication is to introduce an artificial viscosity, which is set to act only at a shock front and introduces adequate dissipation effects. Equivalently, artificial viscosity can be considered as a way to broaden the shock front and turn the mathematical discontinuities, i.e. the jumps, into thin regions where the fluid properties have large but continuous variations, as if there was no shock at all in the first place (Von Neumann and Richtmyer 1950). In this framework, the differential form of the Euler equations still accurately describes the fluid, and so do the SPH equations of

motion; however, the added viscosity is technically unphysical, and information is lost on scales smaller than the broadened shock front.

Among the main reasons to employ the SPH algorithm is its ability to conserve (to machine precision) mass, linear momentum, angular momentum and energy; moreover, this method is invariant under Galilean transformations and as a Lagrangian approach, it can adjust to the large spatial range encountered in cosmological simulations by following particles as they cluster. On the other hand, SPH indeed has some limitations: under certain conditions, it can suppress fluid instabilities, aside from being unable to handle contact discontinuities with high accuracy (Springel 2016).

2.2.2 Moving mesh schemes

Historically, hydrodynamical mesh schemes have been mostly Eulerian, employing a fixed mesh which grants very high accuracy but lacks the adaptivity and Galilean invariance of Lagrangian algorithms such as SPH. To maintain the accuracy and grant the adaptivity, the mesh could be allowed to move with the fluid; this "moving mesh" approach, however, has to deal with its fair share of complications, notably the mesh tangling which excessively distorts cells by giving them a characteristic "bow-tie" shape. To overcome this issue, it is possible to use an unstructured mesh such as the Voronoi tessellation of some dynamic mesh-generating points (Springel 2010; Springel 2016; Weinberger et al. 2020), where each cell of the tessellation comprises the volume of space around a point which is closer to that point than to any

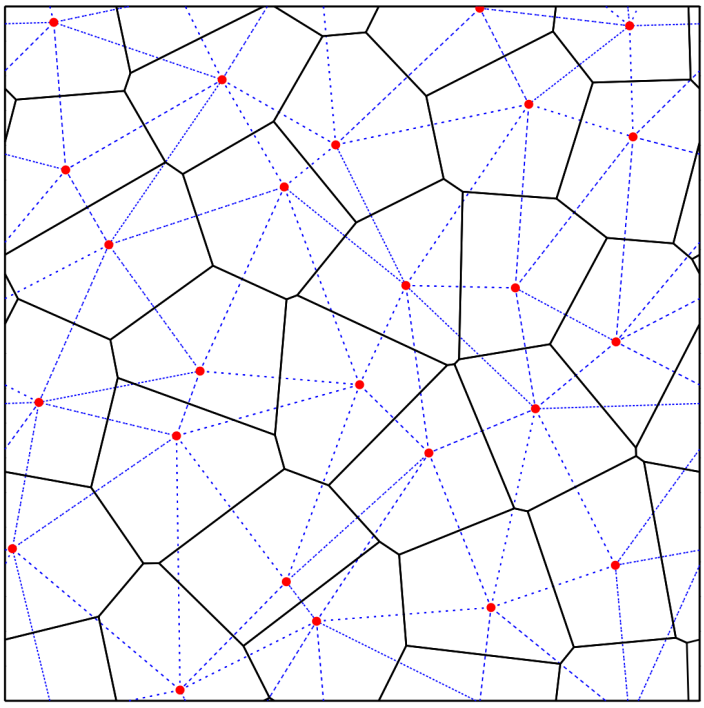


Figure 2.6: Voronoi tessellation (solid black lines) and Delaunay triangulation (blue dashed lines) of a set of mesh-generating points (red) enclosed in a $2D$ periodic box. Each vertex of the Voronoi mesh corresponds to a circumcircle midpoint of the triangulation (figure taken from Springel 2016).

other. The topological dual of the Voronoi tessellation is the Delaunay triangulation, which is uniquely built by requiring that the circumcircles of the triangles having three mesh-generating points as vertexes do not contain any of the other points. The Voronoi tessellation and the Delaunay triangulation of a set of points in $2D$ are shown in **fig. 2.6**; in $3D$, the circumcircles become circumspheres and the triangles become tetrahedra.

To hydrodynamically describe each cell of the tessellation, it is possible to apply the same procedure which would be used for an Eulerian mesh, namely the evolution of the mean properties of cells through flux exchanges with neighboring cells. More specifically, this approach consists of:

1. a reconstruction step, in which cell-averaged properties are used to compute these properties everywhere the cell, typically in a piecewise linear fashion;

2. an evolution step, in which different fluid properties on the shared side of two cells (i.e. the cell interface) consitute a $1D$ piecewise constant initial values problem; in fact, this is a so-called Riemann problem, which can be solved (exactly or approximately) to yield the fluxes at the cell interface and evolve the fluid properties in the two cells by a time Δt;

3. an averaging step, which theoretically needs to be mentioned because changes in the fluid properties propagate as waves that need to be spatially averaged to yield the properties of each cell after a time Δt; in practice, however, this step is not explicitly performed because the fact of approximating fluid properties through their values in each cell handles this step implicitly, since the values assigned to each cell are averaged across its volume.

This reconstruct-evolve-average (REA) scheme is repeated on every timestep, with the addition of a Voronoi mesh reconstruction because mesh-generating points are allowed to move with the fluid as it evolves. This reconstruction actually eliminates the aforementioned mesh tangling issue, because it allows the mesh itself to move according to the motion of the fluid.

The general equations employed in the REA scheme (specifically in the evolution step) can be obtained by considering the following compact form of **eqs. 1.13**, **1.14** and **1.16**, which hold for each cell of the Voronoi mesh (Springel 2016):

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = 0 \,, \tag{2.35}$$

where $\mathbf{U} \equiv (\rho, \rho\mathbf{v}, \rho s)$ and $\mathbf{F} \equiv (\rho\mathbf{v}, \rho\mathbf{v} \otimes \mathbf{v} + P, \rho s\mathbf{v})$, with $\otimes$ being a tensor product. Note that the term with the gravitational potential, which appears in **eq. 1.14**, has

been neglected for simplicity. Normally, it arises in the right hand side of **eq. 2.35** as a so-called source term. The integration of the compact form over the volume V of a cell yields

$$\frac{\partial \mathbf{Q}}{\partial t} \equiv \frac{\partial}{\partial t}\int_V \mathbf{U}\mathrm{d}V = \int_{\partial V}[\mathbf{F} - \mathbf{U}\mathbf{w}^T]\mathrm{d}\mathbf{n}\,, \tag{2.36}$$

where the last equality represents a surface integral over the cell boundary ∂V (with $\mathbf{n}$ being a normal vector to that boundary), and follows from the Gauss theorem (e.g. Ciotti 2021). The quantity $\mathbf{w}$ (transposed as a line vector in the above equation) is the velocity of each point of the boundary, and is generally constant in both Eulerian and Lagrangian mesh codes; specifically, $\mathbf{w} = 0$ in the former case (the mesh is fixed) and $\mathbf{w} = \mathbf{v}$ in the latter case (the mesh follows the local velocity of the fluid). In the hybrid approach of an unstructured mesh, however, $\mathbf{w}$ must be included explicitly in the expression for $\mathbf{Q}$, and taking this quantity into account renders this method Galilean-invariant. The physical meaning of **eq. 2.36** refers to the aforementioned second step of the REA scheme: the fluid properties of a cell depend on the fluxes which enter or exit that cell during a given timestep. Therefore, the discretization of **eq. 2.36** over a timestep Δt and over the finite volume of cells requires an expression for these fluxes, expression which can be found by considering two mesh cells, labeled i and j, and their $2D$ cell interface $\mathbf{A}_{ij}$, (i.e. a vector with modulus equal to the interface area and oriented normally to it). The averaged flux across the interface ij is then defined as

$$\mathbf{F}_{ij} = \frac{1}{|\mathbf{A}_{ij}|}\int_{|\mathbf{A}_{ij}|}[\mathbf{F} - \mathbf{U}\mathbf{w}^T]\mathrm{d}\mathbf{A}_{ij}\,, \tag{2.37}$$

which can be inserted in **eq. 2.36** to yield the integrated fluid properties of cell i, that is, $\mathbf{Q}_i$, through a summation over all interfaces with neighboring cells j:

$$\frac{\partial \mathbf{Q}_i}{\partial t} = -\sum_j |\mathbf{A}_{ij}|\mathbf{F}_{ij}\,, \tag{2.38}$$

which can be discretized as

$$\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n - \Delta t \sum_j |\mathbf{A}_{ij}|\hat{\mathbf{F}}_{ij}^{n+1/2}\,, \tag{2.39}$$

where the superscripts indicate the timestep each quantity is calculated at. Specifically, the quantities $\mathbf{Q}_i^{n+1}$ are the time evolution of $\mathbf{Q}_i^n$ after a time Δt, and depend on the time-averaged fluxes $\hat{\mathbf{F}}_{ij}^{n+1/2}$, which are instead calculated at an intermediate timestep $\Delta t/2$. It is clear that the practical estimate of the fluxes $\hat{\mathbf{F}}_{ij}$ (i.e. the solution of the Riemann problem at the cell interface ij) is a crucial part of the moving-mesh scheme. There are a variety of methods to obtain this estimate: for example, without going into details, an exact iterative Riemann solver (e.g. Gottlieb and Groth

1988) is employed in the moving-mesh code AREPO (Springel 2010; Weinberger et al. 2020), but there are also approximate methods such as the Harten-Lax-Van Leer (HLL, Harten et al. 1983).

In summary, solving the mesh tangling problem through a Voronoi tessellation yields an accurate, highly adaptive and galilean invariant fluid treatment, which, along with the tree method for gravity calculations, is also suited to the new algorithm which will be described in the next chapter.

3 | The Dynamic Zoom Simulations algorithm

The first two chapters provided an introduction to the general cosmological and galaxy formation framework, along with the techniques adopted in modern simulations to follow the involved physical processes with a good level of accuracy. This accuracy critically depends on the number N of tracer particles; as mentioned in **section 1.4**, however, there is still an upper bound to the possible achievable resolution, which will eventually become too coarse to obtain the accuracy required for the interpretation of observational data. In fact, the upcoming generation of instruments such as Euclid (Laureijs et al. 2011), WFIRST (Spergel et al. 2015), LSST (Ivezić et al. 2019) and SKA (Braun et al. 2015) will provide extensive amounts of data with an unprecedented quality, and obtaining simulated results with a corresponding quality could already be too expensive in terms of run time and storage requirements. Regarding the latter, it should be noted that simulation output is traditionally saved to disk as so-called "snapshots", which depict the whole simulation box at a given cosmic time; on the other hand, observational data comes in a lightcone form, due to the light generated by objects at different distances having a finite speed. Therefore, snapshots cannot be directly compared to observations; instead, they have to be combined into "lightcone-like" data, for example by selecting from each snapshot the particles located in a thin spherical shell with radius equal to the lightcone radius.[1] The thickness of each shell yields the accuracy of this piecewise-constant approximation; tipically, an acceptable accuracy requires a large number of snapshots, in order to frequently sample the cosmic time covered by the simulation. Obviously, this results in a large storage requirement, which rapidly becomes prohibitive as more particles are included in a simulation.

An easy solution to the storage problem, at least in theory, is to perform the

[1]The lightcone radius R_{lc} at a given time t_{lc} is the radius of the sphere obtained from the lightcone through the constraint $t = t_{lc}$ (**fig. 1.1**.) at the time the snapshot was taken and centered on the observer. The lightcone itself is calculated with the observer as the reference event.

piecewise-constant approximation that leads to the lightcone-like output during the simulation, without saving to disk the whole snapshots. Regarding the computational resources required during the run, they should reflect the change to lightcone-like output and concentrate on those portions of the domain which will, at some point, be saved to disk.

3.1 Algorithm outline

If a simulation software generates outputs in a lightcone-like fashion from the time when the lightcone is entirely inside the simulation box (i.e. when $R_{lc} < L/2$, with L box side, when the observer is in the center of the box), an increasingly large fraction of particles will not be saved to disk. This concept is depicted in **fig. 3.1**, where structures in the grey shaded area of the simulation domain will not be included in any lightcone-like output. From a computational point of view, however, this area is usually treated no differently than the area inside R_{lc}, so there is a vast amount of resources and run time spent to evolve particles which do not show up in the output files. To save these resources, it is possible to simply remove the grey area of **fig. 3.1** from the simulation domain, which in practice becomes smaller and smaller towards $z = 0$ (Shrinking Domain Framework, SDF, Llinares 2017). However, it is worth noting that this results in a loss of periodicity of the system, which implies modifications of the Poisson equation and, consequently, of the gravity solver. These modifications also take care of large-scale gravitational effects, which otherwise would be lost due to the removal of particles. With a fully relativistic treatment of gravitational interactions, particles inside of the lightcone would not be affected by those outside (due to the finite propagation speed of gravitational perturbations), but as mentioned in **section 1.4**, many softwares actually operate in the newtonian framework described in **section 2.1**. In this case, the force field felt by a particle is istantaneously generated by every other particle in the simulation domain (including the ones outside of the lightcone).

To efficiently combine the relativistic concept of lightcones and the newtonian approximation of simulation codes, a new method dubbed "Dynamic Zoom Simulations" (DZS, Garaldi et al. 2020) has been recently developed and implemented in a DM-only fashion into the code `PGADGET-3`.[2] This code is capable of describing gravitational interactions with a tree-PM algorithm, while also handling hydrodynamics through an SPH approach. Moreover, `PGADGET-3` is also massively parallel, i.e. it allows multiple computational units (called "tasks") to work simultaneously on

[2] An earlier version of this code, called `GADGET-2`, is presented in Springel (2005).

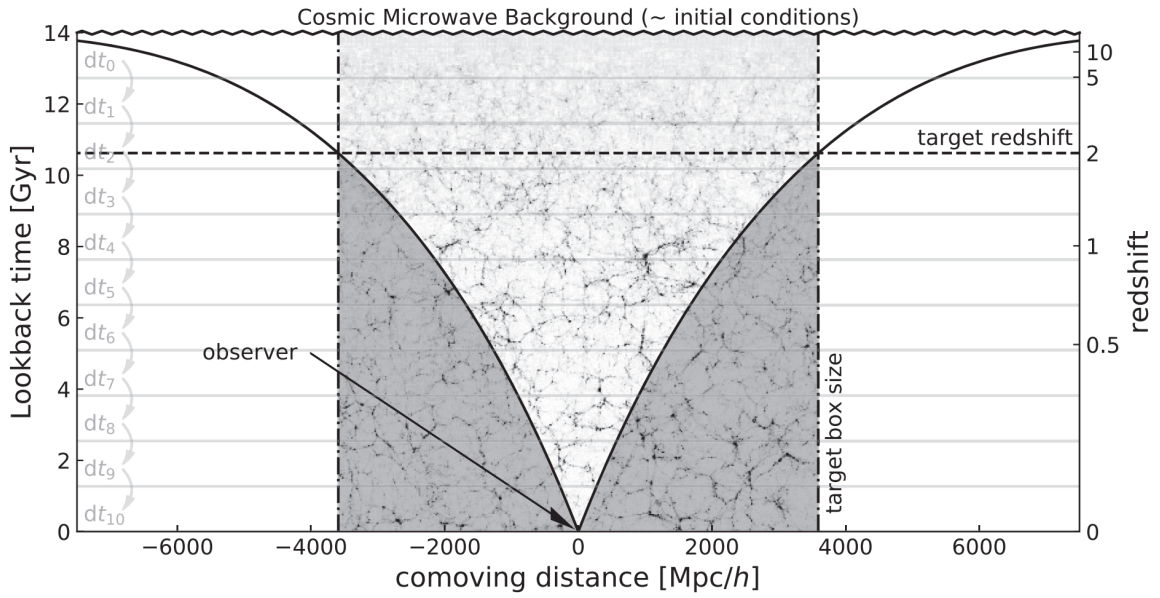


Figure 3.1: space-time diagram of a cosmological simulation in a 1D+1D fashion; the light-cone radius (black solid lines) crosses the box boundaries at a certain "target" redshift, i.e. the maximum redshift which allows the production of lightcone-like output. This output is created in every discrete timestep dt_i from particles crossing the lightcone radius; therefore, the grey shaded area is always discarded in the output process (figure taken from Garaldi et al. 2020).

different fractions of the whole simulation domain while a "master task" performs coordination and serial operations when needed. Specifically, the parallelization of `PGADGET-3` is managed by the Message Passing Interface (MPI, Clarke et al. 1994). This is a so-called distributed memory model, which means that every task accesses its own memory and variables, which can (and usually will) share the same name across tasks but store different content. To access this content from other tasks, it is necessary to resort to communications.

Unlike the SDF, the DZS approach does not remove particles outside of the light-cone, rather, it merges them, thus reducing their number and the simulation resolution; this effectively reduces the computational effort outside of the lightcone and concentrates it inside R_{lc}, where the original number of particles is left untouched as those particles will actually be stored in a lightcone-like output. Moreover, the system of particles remains fully periodic, so the gravity solver does not need any modifications to run DZS correctly. It should be noted, however, that the decreased resolution outside of the lightcone does indeed represent an approximation which leads to differences with respect to a "standard" run. In fact, as it will be shown in **chapter 4**, these differences are usually small, and in any case they can be reduced through the parameters of the algorithm, at the cost of a smaller performance gain.

3.1.1 Tree-based DZS

The particle merging operation and subsequent resolution reduction which characterise the DZS method can be naturally carried out in a gravitational oct-tree framework. In fact, tree nodes approximate the combined gravitational effect of the particles they contain, as if each node not opened during a gravitational force calculation contained a single, massive particle. The DZS algorithm takes this concept a step further by actually converting into such particles (hereafter, "derefining") the tree nodes that fulfill a specific criterion, which first and foremost checks if the nodes are located outside of the lightcone, because only such nodes are allowed to be derefined. When a new particle is created from a node, that particle inherits the properties of the node, for example its total mass, center of mass and center of mass velocity, whereas the particles located inside the derefined node are eliminated from the simulation domain. The whole process is carried out across the timesteps of a simulation (in order to follow changes in R_{lc}), with the net result of a "zoomed" high resolution region (i.e. the lightcone) which dynamically shrinks towards the observer as $z \to 0$.

First of all, the node derefinement process requires every tree node to be tested with the criterion and eventually flagged for derefinement; in this case, the particles inside the node are also flagged for removal. This whole procedure requires a single depth-first tree walk, which makes the algorithm very efficient. After the walk, particles are actually created from nodes and eliminated depending on the previously assigned flags. The tree walk is performed as shown in **fig. 3.2** for a single task on a given timestep: starting from the root node labeled as 1 (which is never derefined[3]), the walk passes to node 2, which is checked with the criterion but does not satisfy it. Since that node has a child, the walk continues through the dashed red arrow to object 10: this is a particle which does not need to be eliminated, since its father node is not going to be derefined. The walk cannot go deeper from here, because object 10 does not have any children, so it moves on to its siblings. Object 11 is a node, and as such is tested with the derefinement criterion; in fact, this node satisfies the criterion (as indicated by the green arrow), so it is flagged for derefinement. The tree walk then moves on to particle 34, which is flagged for elimination because it is located inside a node to be derefined. After this flagging, the walk continues to siblings: in the case of **fig. 3.2**, all of these siblings are particles, and therefore they are also flagged for removal. Once every sibling has been walked, the walk simply returns

[3]Derefining the root node would lead to a simulation with only one particle which retains the mass of the whole box, i.e. a particle which would not have any interactions with anything, except for its periodic images.

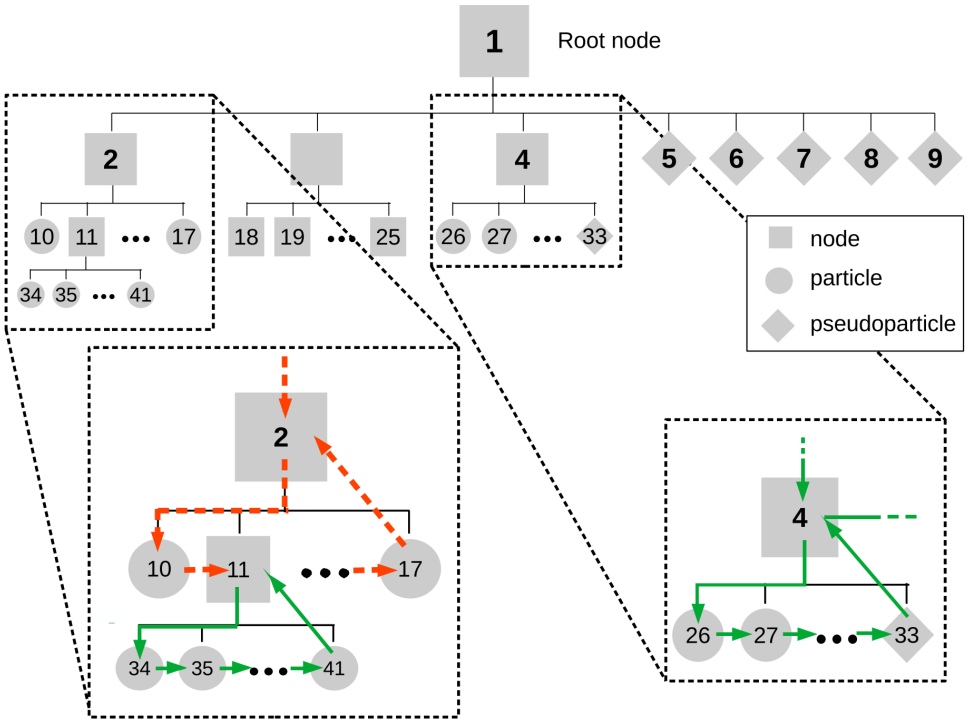


Figure 3.2: scheme of the tree walk performed by the DZS algorithm in `PGADGET-3` Specifically, the two zoomed boxes highlight the tree walk through the use of arrows: solid green arrows refer to portions of the walk which lead to particles to eliminate, whereas objects walked through dashed red arrows are left untouched by the algorithm (figure taken from Garaldi et al. 2020).

to the last object analysed in the previous tree level (i.e. node 11) and moves on to its siblings. Once those have also been analysed, the walk goes up yet another level, i.e. to node 2, before going to its siblings.

A different situation is encountered in node 4, which is represented in the rightmost zoom box of **fig. 3.2**. The node satisfies the derefinement criterion, so the walk from that node follows a path similar to that seen in node 11. This time, however, the walk will eventually encounter object 33, which is neither a node nor a particle; in fact, it is a so-called pseudoparticle. In a parallel computing framework, particles are distributed among tasks in a process called domain decomposition[4], which means that the particle content of some nodes may be located across multiple tasks. This is where pseudoparticles, i.e. objects that cannot be opened and simply trace the mass of particles assigned to other tasks, come into play. In the DZS framework of **fig. 3.2**,

[4] A domain decomposition employs a special oct-tree whose lower level nodes, called leaves, contain a set number of particles (unlike the gravitational tree described in **subsec. 2.1.2**, which has at most one particle per node in the lower levels) and are distributed among tasks. The tasks which hold the particles inside these leaves will actually employ those particles while building the gravitational tree.

pseudoparticles are not locally flagged by the algorithm (as they do not contain any local particles); instead, object 33 will be handled by the task which stores its particle content.

As mentioned above, the node derefinement process is carried out according to a certain criterion whose main purpose is to exclude from derefinement the nodes inside the lightcone. In fact, since the criterion applies to tree nodes, it is possible to generalise the one normally used by the hierarchical multipole method (**subsec. 2.1.2**), expressed in **eq. 2.21** or **eq. 2.23** (depending on which one is chosen by the tree method), to test if a node can be derefined. The generalisation is necessary because the tree method uses the geometrical and dynamical criteria locally, i.e. by relating a node and the particle which feels the gravitational force. In the DZS framework, the involved objects are the node to be tested and a point on the spherical surface of the lightcone; specifically, this point is located at $\mathbf{r}_{lc}$ ($|\mathbf{r}_{lc}| = R_{lc}$), on the line connecting the observer and the center of mass of the node $\mathbf{s}$ (such that $|\mathbf{s} - \mathbf{r}_{lc}| = |\mathbf{s}| - |\mathbf{r}_{lc}| = |\mathbf{s}| - R_{lc}$). Formally, this yields the following criteria applicable to the DZS algorithm:[5]

$$\text{geometrical:} \quad \frac{L}{|\mathbf{s}| - R_{lc}} \leq \theta_{geom} \,, \tag{3.1}$$

$$\text{dynamical:} \quad \frac{Gm_{node}}{(|\mathbf{s}| - R_{lc})^2}\theta^2 \leq \alpha|\mathbf{a}_{lc,old}| \,, \tag{3.2}$$

where $\mathbf{a}_{lc,old}$ is the minimum acceleration for the previous timestep among those of the particles located inside of the lightcone, and generalises the quantity $\mathbf{a}_{i,old}$ of **eq. 2.23**. Note that **eq. 3.2** requires the additional condition $|\mathbf{s}| - R_{lc} > 0$, i.e. the center of mass of the tested node should be located outside of the lightcone. Moreover, the opening angle θ_{geom} (or the fraction α if one wishes to use the dynamical criterion) can be viewed as a free parameter of the DZS algorithm: a large θ_{geom} yields more derefined nodes, i.e. fewer particles overall; this translates into a larger performance gain, but also into heavier modifications of the simulation domain and consequently a coarser approximation with respect to a standard run. Conversely, reducing the angle sacrifices some performance gain for more contained approximations.

So far, Dynamic Zoom Simulations have been described in a rather general fashion, highlighting the employed criteria and the core operations of the tree walk. From a practical point of view, however, implementing the algorithm into a numerical code involves many more steps, for example the actual derefinement of nodes and removal of particles, the lightcone radius calculations and the relationship between the algorithm and the workflow of the code. Moreover, there are additional

[5]Note that the system of coordinates depicted here uses the observer, which is usually assumed to be at the center of the simulation box, as its origin.

adjustments to be made in order to ensure the correct behaviour of a simulation performed with the DZS algorithm. In the following section, all these topics will be discussed as part of the new implementation of DZS capabilities in the numerical code AREPO.

3.2 The AREPO implementation

The existing PGADGET-3 implementation of the DZS algorithm has the benefits of being fast and efficient in terms of run time and memory and work-load balance,[6] as well as yielding very contained errors (Garaldi et al. 2020). This implementation is currently limited to dark-matter only simulations; however, as mentioned in **section 1.4**, simulations of galaxy formation usually are the most computationally expensive, so to fully exploit the potential of this technique, it is necessary to adapt the DZS algorithm also to the case of full-physics simulations. For this reason, this work presents an implementation of the DM-only DZS algorithm to a state of the art software for galaxy formation simulations, i.e. the public release of AREPO (Springel 2010; Weinberger et al. 2020), as well as a first approach to baryon derefinement.

The AREPO code, which is written in the C language, uses a tree-PM scheme to treat gravitational effects (similarly to PGADGET-3) and handles the hydrodynamic evolution through a moving mesh scheme. It is also massively parallel and uses the MPI standard. By default, AREPO includes 6 different particle types, ranging from gas particles (type 0) and dark matter particles (type 1) to stars (type 4) and black holes (type 5), each type with its own softening length. It should also be noted that each of these particles is able to evolve in time according to its own timestep, for example to allow regions with a very high particle density to use smaller timesteps than lower-density portions of the domain. Every individual timestep $\mathrm{d}t_n$ is related to a "system" timestep hierarchy $\mathrm{d}t$ as $\mathrm{d}t_n = \mathrm{d}t/2^n$, in order to have so-called global simulation steps[7] where all particles have a simultaneous evolution.

In describing the new implementation of the DZS algorithm, it is first appropriate to follow the footsteps of Garaldi et al. (2020) and start with its DM-only implementation in AREPO, from the initial setup (**subsec. 3.2.1**) to the tree walk and particle creation and elimination in simulation steps (**subsecs. 3.2.2–3.2.4**). Due to the vari-

[6] The work-load balance of a simulation step is defined as the maximum across all tasks of the wall-clock time spent on the tree algorithm, divided by its average value.

[7] It is appropriate to make a distinction between particle timesteps, which refer to the evolution of single particles, and simulation steps, in which multiple particles, potentially with different individual timesteps, are evolved up to a certain simulation time t (i.e. they are synchronised to that time).

ous operations and tree object types that participate in the derefinmement process, the general flow of the algorithm is not always easy to follow, which is why **fig. 3.3** provides a graphical overview of the whole DZS action and will be repeatedly taken up later. This series of schemes show a $2D$ particle distribution shared across two tasks from before the DZS algorithm starts to operate (**figs. 3.3a** and **3.3b**) to the final result (**fig. 3.3f**), passing through the tree walk (**figs. 3.3c** and **3.3d**) and the particle creation and elimination stage (**fig. 3.3e**). The node derefinement process is depicted only for a couple of nodes (namely, node 3 and node 8) in task 2 for the sake of simplicity. The inclusion of another task in **fig. 3.3** serves the purpose of showing a new kind of tree object, namely the imported point, which has been introduced in AREPO and was absent in PGADGET-3. The role and DZS-related treatment of imported points will be explained at the end of **subsec. 3.2.2**; for now, it is appropriate to consider them as any other particle, such as those which in **fig. 3.3d** are inside nodes to be derefined and are thus eliminated in **fig. 3.3f**. Once the dark matter related portion of the algorithm has been described, it is finally possible to introduce an approach to baryon derefinement (**subsec. 3.2.6**).

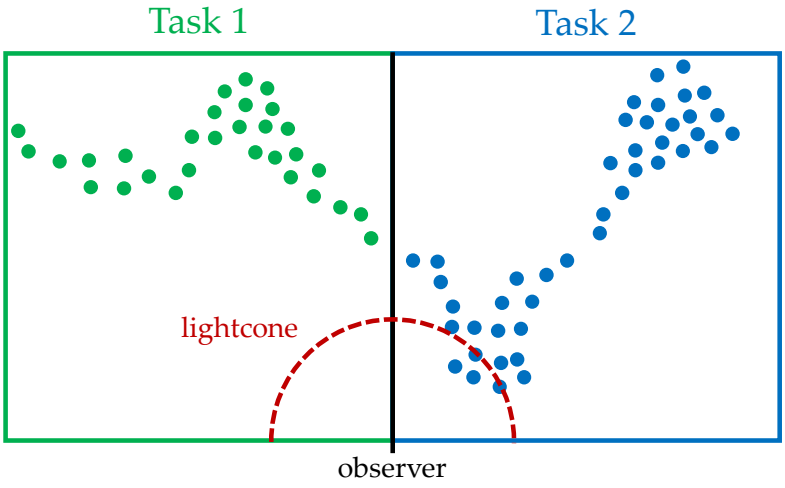
3.2.1 Initial setup

There is a minimal amount of instructions that should be carried on at the beginning of a Dynamic Zoom Simulation, which mainly involve determining the lightcone radius as a function of time. This can be accomplished through the following integral, which holds in a flat universe:

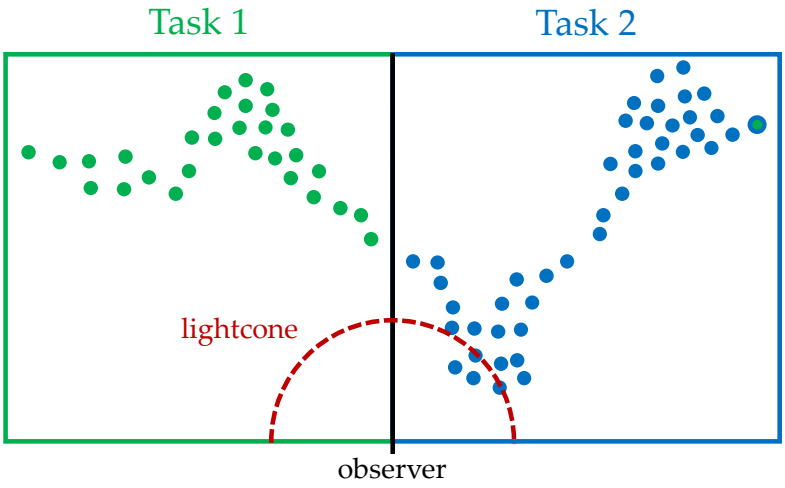
$$R_{lc}(z) = k \int_0^z \frac{\mathrm{d}z'}{[\Omega_{m,0}(1+z')^3 + \Omega_{\Lambda,0}(1+z')^2]^{1/2}} \,, \tag{3.3}$$

where k is a constant depending on the speed of light, the unit of length employed in the simulation and the chosen H_0 value,[8] which for every simulation of this work is set to $H_0 = 67.66\ km\ s^{-1}\ Mpc^{-1}$ (Planck Collaboration 2020). In practice, the integral is discretized as a sum with $\mathrm{d}z' = 10^{-5}$. Since the above integral depends only on the cosmological parameters $\Omega_{m,0}$ and $\Omega_{\Lambda,0}$, it is possible to use the same values across different simulations if the cosmological model remains unchanged; furthermore, employing cosmological models other than the ΛCDM one could lead to more complicated integrals which is not optimal to calculate on the fly. For these

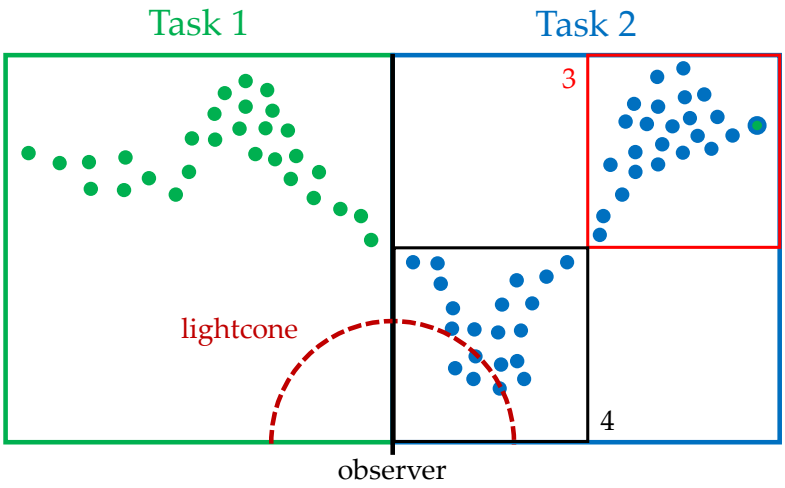
[8] Tipically, due to the usage of different values of H_0 among scientific works, it has become common pratice to express the Hubble constant in terms of the parameter "little h" as $H_0 = 10^2 h\ km\ s^{-1}\ Mpc^{-1}$. If the dependency on h of each quantity (e.g. distance, mass, etc.) is clearly expressed, adjusting the results of scientific works to a certain value of H_0 is as simple as substituting the appropriate value for h (Croton 2013).


Task 1
Task 2
lightcone
observer

(a)


Task 1
Task 2
lightcone
observer

(b)


Task 1
Task 2
3
lightcone
4
observer

(c)

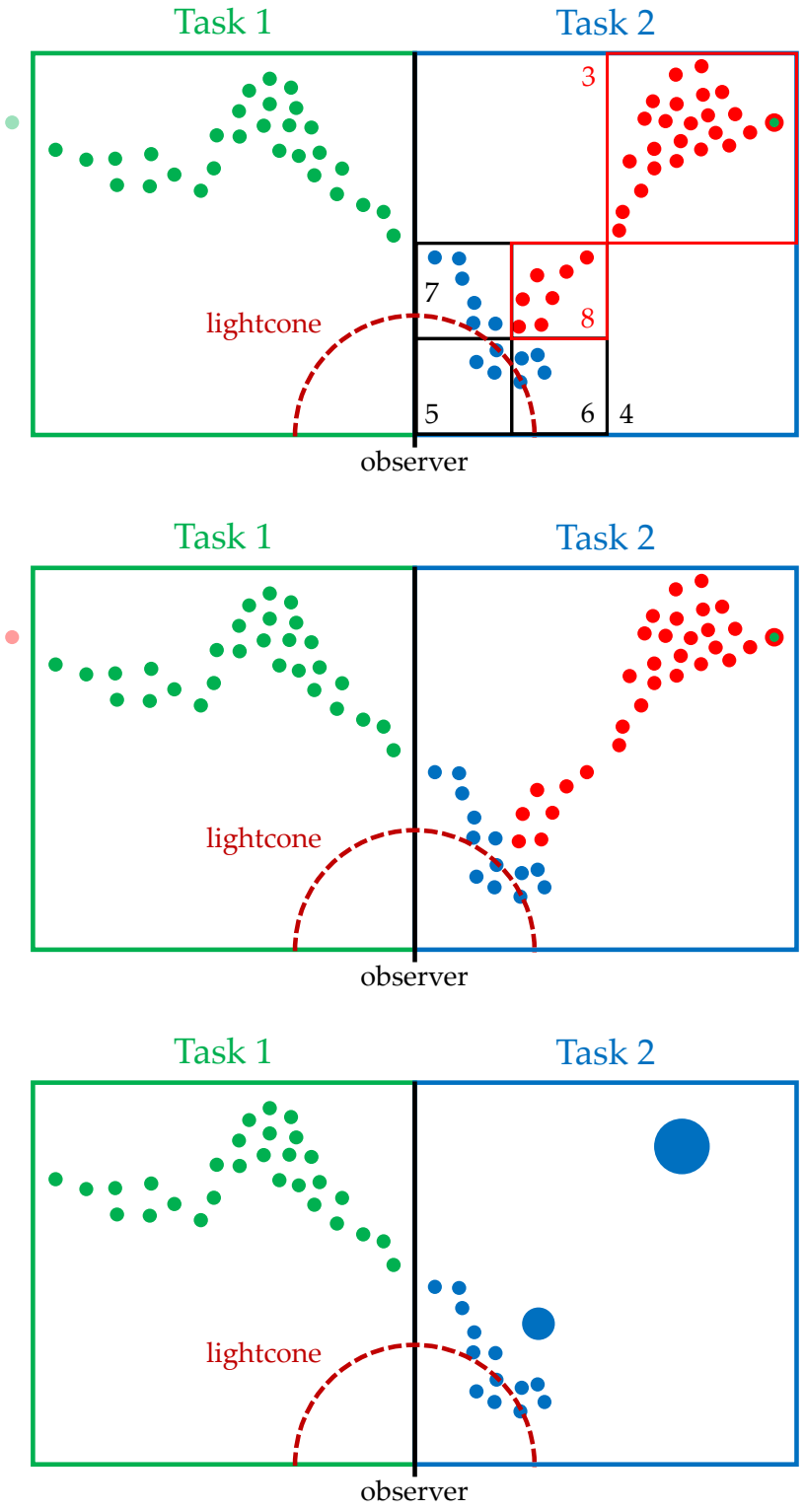


Figure 3.3: series of chronologically ordered 2*D* sketches showing how the node derefinement process takes place in one MPI task (namely, task number 2). The two bigger squares are the domain leaves assigned to each task, the smaller ones in **(c)** and **(d)** are tree nodes (with the red ones satisfying the derefinement criterion). The particles (circles) are colored depending on the task they are located in, and are marked in red when they are flagged for elimination. Finally, the green particle with a blue or red border in task 2 is an imported point linked to the slightly faded particle outside of the task 1 leaf.

reasons, the DZS algorithm offers the possibility to read at the beginning of the simulation a so-called lightcone file, containing pre-calculated values of the lightcone radius at certain cosmic times (possibly with more advanced integration techniques than the direct summation). At any given simulation step, these values are linearly interpolated to yield the lightcone radius corresponding to the current simulation time.

3.2.2 Tree walk

One of the main parts of the DM-only DZS algorithm is the tree walk, qualitatively described in **subsec. 3.1.1**. This description already highlighted the different operations performed on the various kinds of tree-related objects: nodes are tested and eventually flagged for derefinement, particles are flagged for elimination, pseudoparticles are mostly ignored. In fact, the first question to ask in a practical implementation of the walk is how to distinguish different tree objects: `AREPO` assigns to each of them a unique integer tag, i.e. a variable called `no`, and depending on the specific value of this tag, each entity is classified as a node, a particle, a pseudoparticle or an imported point. In practice, the tree walk is performed by each task through the function `walk_tree`, which takes the tag of the root node as its first argument, with an additional integer argument `erase_particles`, initially set to zero. When the walk reaches a node `no` which satisfies the derefinement criterion, this latter argument is changed to a non-zero value, signaling that the particles inside `no` should be flagged for removal.

As mentioned earlier, the first job performed by the function `walk_tree` is checking what kind of object its first argument represents; initially, `no` is the root node which, as mentioned above, cannot be derefined. For this reason, the function immediately checks for the first child node of `no`, named `nno`. At this point, `walk_tree` simply calls itself, but with `nno` as its first argument. In fact, this recursive behaviour characterises the whole tree walk: for example, instances of `walk_tree` related to nodes call themselves with their first child as an argument, and these new instances will eventually call `walk_tree` for their children and siblings. Once an instance returns (i.e. finishes to do its operations), the workflow of the tree walk simply resumes from immediately after that instance was called. It should also be mentioned that each separate instance of `walk_tree` has its own variables, which are not shared with other instances unless they are provided as arguments of a function call.

As mentioned above, the first nested call of the tree walk function will involve a child of the root node; generally, the operations performed by `walk_tree` when `no`

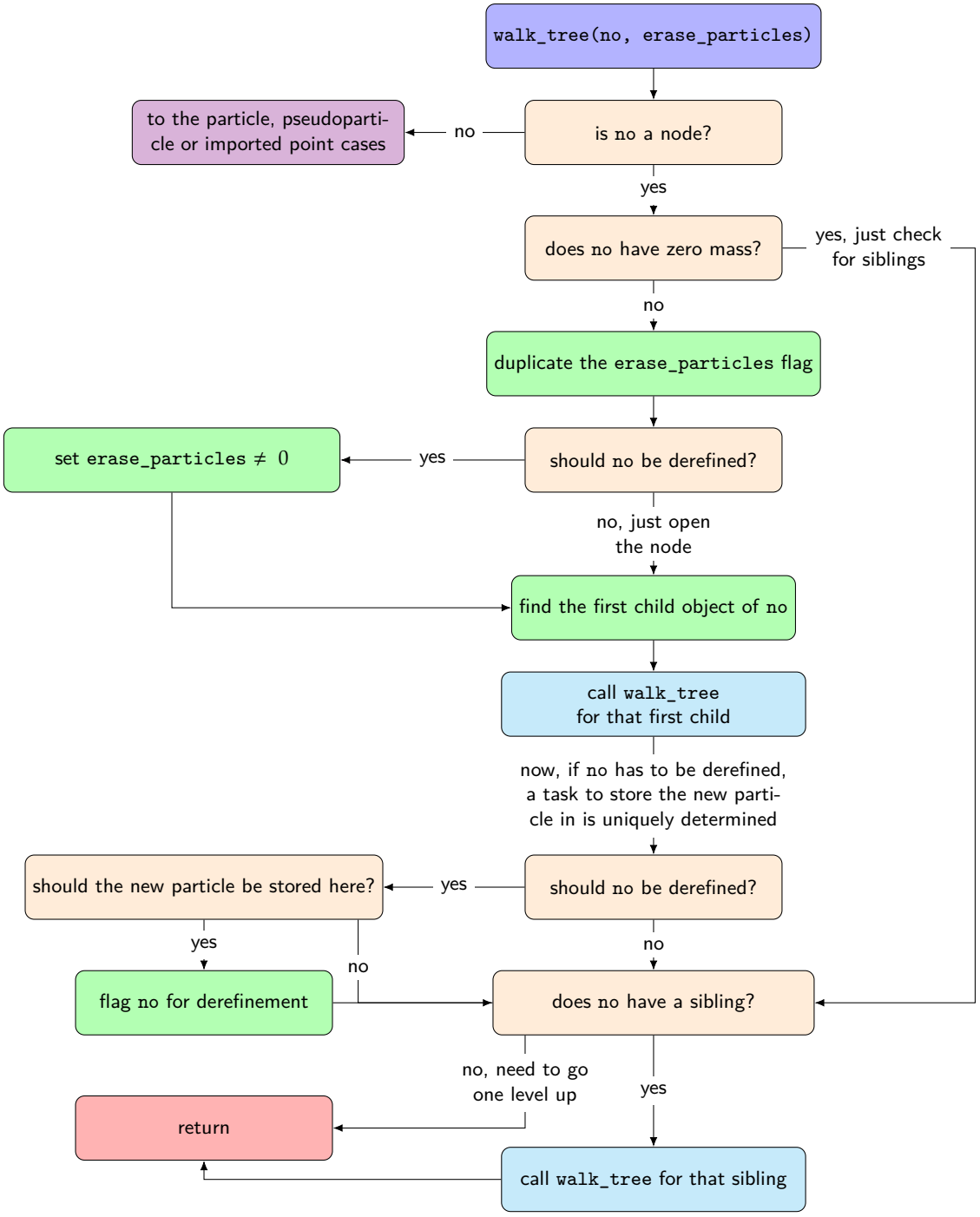


Figure 3.4: flowchart of the function `walk_tree` in the case of `no` = node. The start of the function is marked in blue, operations in green, yes/no checks in orange and recursive calls of walk tree within the function itself are highlighted in cyan. The red box represents the end of the function, whereas the purple box refers to other `no` types.

refers to a node are outlined in **fig. 3.4** and sketched in **fig. 3.3c** and **fig. 3.3d**. First of all, it should be verified that the node has a non-zero mass, otherwise there is nothing to derefine in the first place and the walk can simply move on to siblings. On the

other hand, if the node is not empty, the derefinement criterion checks wheter or not the node `no` needs to be derefined and, if so, the flag `erase_particles` is updated to a non-zero value. This communicates to the subsequent nested call to the first child of `no` (the upper cyan box in **fig. 3.4**) that the particle content of the tree branch departing from `no` should be flagged for removal. Graphically speaking, this stage of the walk corresponds to the case `no` = 3 in **figs. 3.3c** and **3.3d**, where node number 3 is marked in red because it satisfies the derefinement criterion. More specifically, **fig. 3.3c** represent the state of the simplified domain with two tasks just before the upper cyan box of **fig. 3.4**, i.e. when node 3 has already been tested, but its particle content has not been reached by the walk yet. After all the nested calls on the branch departing from `no` = 3 have returned, the system is depicted by **fig. 3.3d** with the red particles being flagged for removal. Note that if the case discussed here was that of `no` = 4 (outlined in black in **fig. 3.3c** as it does not need to be derefined), the flowchart in **fig. 3.4** would simply continue the walk to lower levels of the tree to test children nodes with the derefinement criterion. In fact, when passing from **fig. 3.3c** to **fig. 3.3d** from the point of view of node 4, one of its children, namely node 8, satisfies the derefinement criterion and its particle content is flagged for removal.

If the node `no` examined in **fig. 3.4** has to be derefined, the particles inside the branch of that node will compare their tasks of belonging in order to find that with the minimum rank[9] and select it as the holder of the new particle. This approach, altough possibly leading to an uneven new particle distribution among tasks (which will be fixed by a domain decomposition anyway), is able, through a simple minimisation, to easily and uniquely determine a task to store new particles in. Most importantly, it does not require any data communication between tasks, which can be an expensive operation in terms of performance. After the task to store the new particle in has been determined, it is necessary to check once again if node `no` has to be derefined. This repeated check, which will obviously yield the same result as the upper one in **fig. 3.4**, is needed because at this point the workflow comes from an operation which needs to be performed for every node, i.e. calling `walk_tree` for its first child, whereas the next operation depicted in **fig. 3.4** only concerns nodes to be derefined. This operation, namely the actual flag for derefinement of node `no`, will only be carried on by the task whose rank is the minimum identified above, i.e. the task which will create the new particle. After this operation, the walk along the tree branch of `no` is effectively done, and it can continue to the siblings of `no`, if there are any which have not been walked to yet. Otherwise, the function `walk_tree` can

[9] In `AREPO` (and generally in the MPI framework), each task is identified by an integer value, called rank, ranging from zero to the total number of tasks minus one.

return. When all the functions in a level of a branch have returned, the walk goes up a level in the tree and continues from where it called the functions (i.e. after the upper cyan box in **fig. 3.4**). Eventually, the instance of `walk_tree` with `no` = rootnode will also return and the walk will finally end.

For the sake of completeness, it is appropriate to add a couple of details to the role of the `erase_particles` flag. As described above, this flag signals that the node `no` has to be derefined by changing to a non-zero value, and that value is passed to lower levels of the tree within the first nested call in **fig. 3.4** to be used for particle flagging. However, the non-zero flag also notifies the children of `no` (in fact, the whole tree branch of `no`) that their father has already satisfied the derefinement criterion, and it is not necessary to separately derefine the children because their particle content will be eliminated anyway due to the flagging of `no`. Specifically, each of the checks "should `no` be derefined" in **fig. 3.4** is complemented by an additional check, to verify that `erase_particles` is still zero up to that point in the tree walk. If that is not the case, there is no point in flagging for derefinement the node currently tested with the criterion, because one of its "ancestors" (i.e. father, grandfather etc.) has already been flagged in an upper tree level. In any case, the currently tested node should be able to pass this `erase_particles` information to its siblings, because they share the same relation with higher levels of the tree. At the same time, if the flag is still zero, a node should be able to set it to a non-zero value and signal to its children, grandchildren etc. that their particle content should be eliminated. For this reason, `erase_particles` is duplicated, with one flag being communicated to siblings of the currently examined node and the other one, eventually changed as needed to a non-zero value, to its children.

After having described how the tree walk behaves when `no` is a node, it is now appropriate to also describe the case of the other tree objects. The operations performed by `walk_tree` when `no` is a particle are depicted in **fig. 3.5**. The function now simply checks if `erase_particles` has been changed from its initial value of zero, signaling that a parent node of particle `no` needs to be derefined. In this case, the function minimises the task rank to store the new particle in, as mentioned earlier, and flags the particle `no` for elimination. This flagging procedure sets the particle mass, unique identifier[10] and velocity components to zero. In fact, before setting the `ID` to zero, its value is saved so that the ID can be inherited by new particles created from derefined nodes, ensuring ID uniqueness. In fact, the specific ID assigned to

[10]This non-negative integer identifier, aptly called `ID`, is independent of the tree and remains unchanged for the whole simulation (if a particle does not need to be eliminated); it is also unique among tasks, meaning that two particles cannot have the same `ID` even if they are located in different tasks.

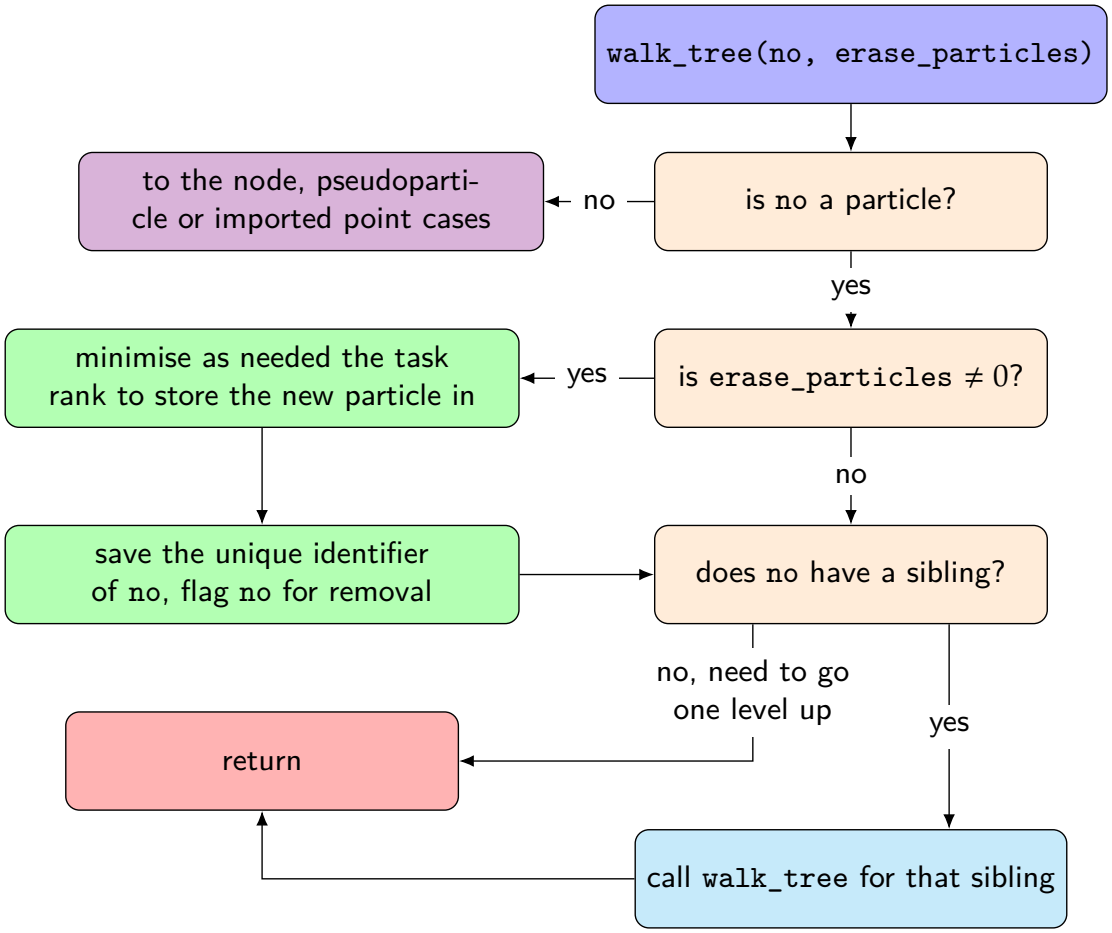


Figure 3.5: flowchart of the function `walk_tree` in the case of `no` = particle, with the same color scheme as **fig. 3.4**.

the new particle will be that of the particle flagged last in the branch of the former node, because only one ID per new particle is needed and so every flagged particle overwrites the same ID variable. Once these flagging operations are done, the function tries to walk to a sibling of `no`, similarly to the `no` = node case. Referring to the tree walk that describes the derefinement of node 3 in **fig. 3.3**, the particle flagging operation takes place between **figs. 3.3c** and **3.3d**.

The structure of `walk_tree` for a pseudoparticle, depicted in **fig. 3.6**, is essentially identical to the particle one, but it has a couple of important differences: first of all, the aforementioned "task of belonging" for the minimisation process is the task where the particles which `no` traces are stored. Using the rank of this task ensures that every task contributing to the content of a node to be derefined achieves, at the end of the walk along its branch, an identical result for the task to store the new particle in. The second main difference, as mentioned in **subsec. 3.1.1**, regards the fact that, even with `erase_particles` $\neq 0$, there is no flagging operation to be

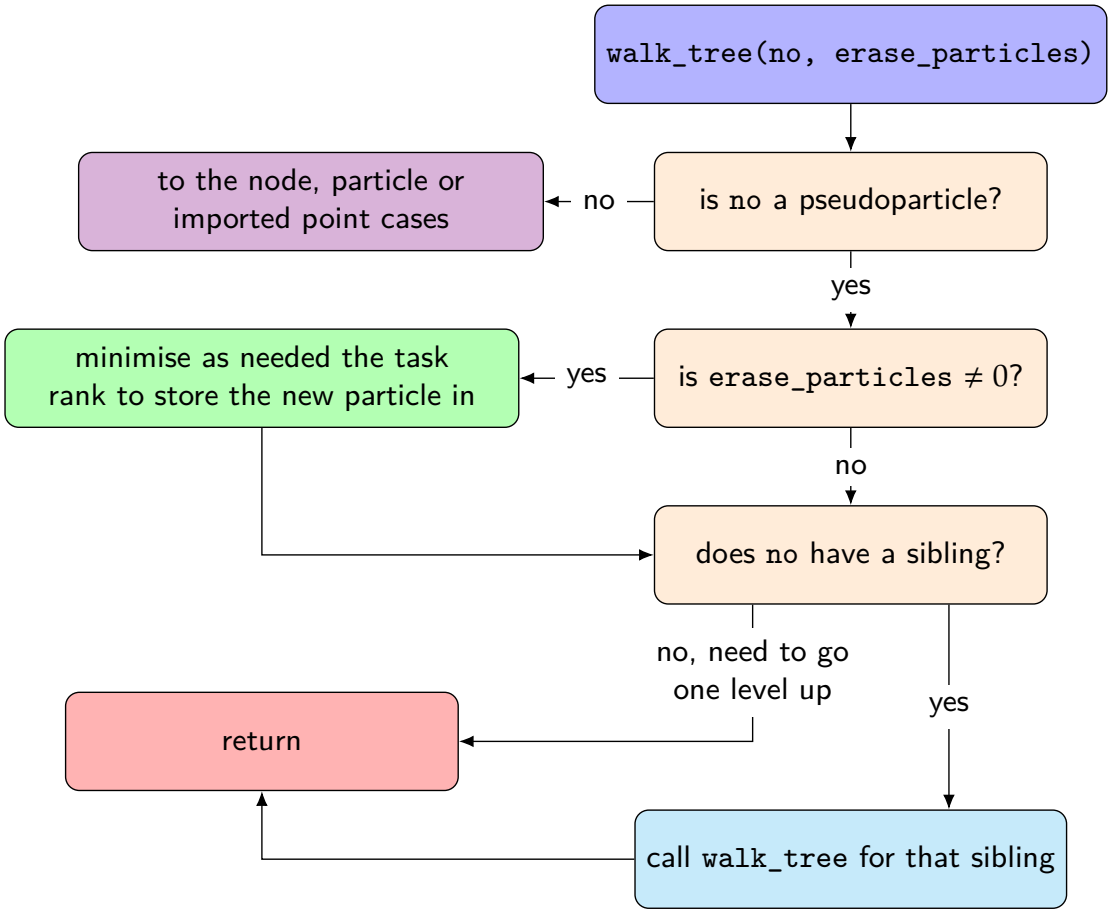


Figure 3.6: flowchart of the function `walk_tree` in the case of `no` = pseudoparticle, with the same color scheme as **fig. 3.4**.

performed, as there are no local particles to flag inside `no`. Checking the value of `erase_particles` has the only purpose of performing the minimisation operation if needed, and from that point, the tree walk to siblings continues with the exact same operations seen in other cases.

The handling of imported points by the tree walk, on the other hand, is formally identical to the case `no` = particle (**fig. 3.5**), but their flagging and elimination operations are much more complicated. First of all, it is necessary to explain what imported points are: following Springel (private communication), the need for them in AREPO arises because not every simulation timestep performs a domain decomposition, and particles, during their time evolution, may move out of the leaf of the domain tree that they were assigned to and "invade" the boundaries of other tasks. Ideally, these particles would be assigned to a new task thanks to a domain decomposition, which however is not performed. Nonetheless, the tree construction (which relies on the domain oct-tree) and related gravity calculations should be in-

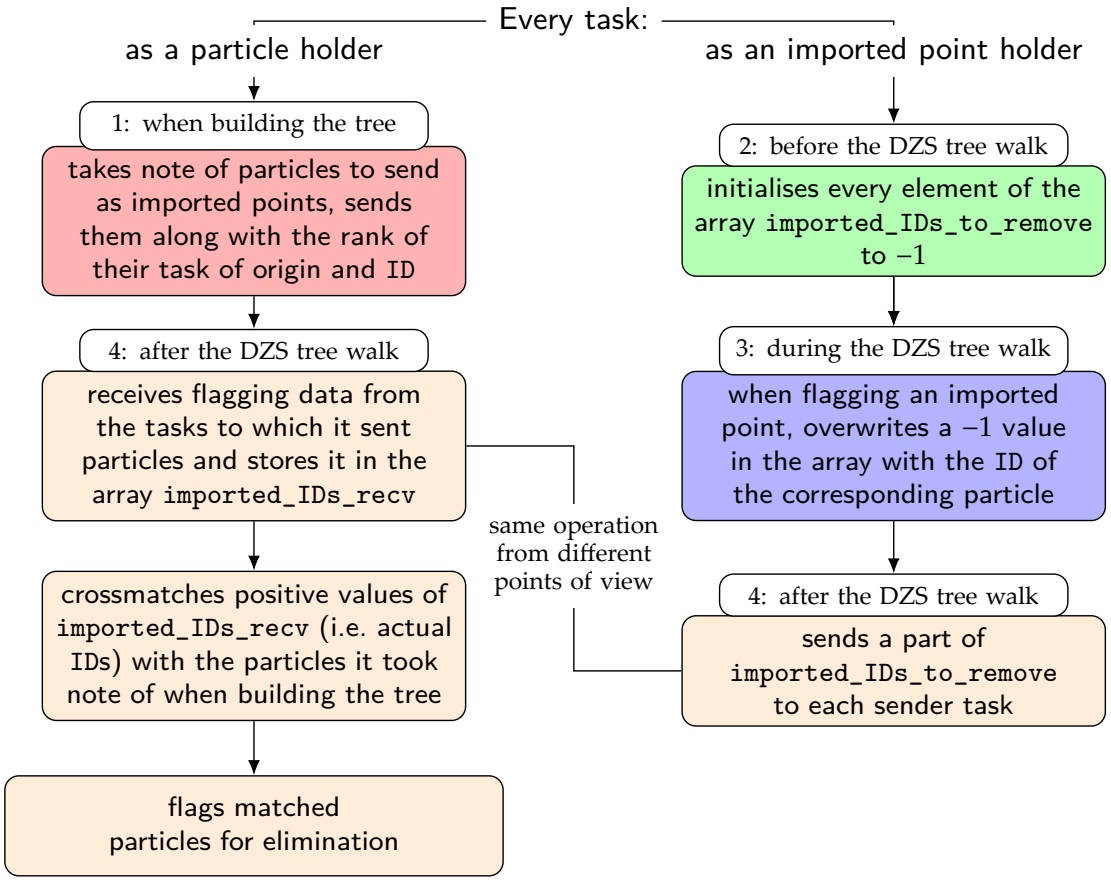


Figure 3.7: flowchart showing the various operations that need to be performed by the DZS algorithm to handle imported points. The colors indicate stages of the workflow with respect to the tree construction and its DZS walk, with numbers highlighting their chronological order.

dependent of the presence or not of a domain decomposition, which is why these particles are indeed assigned to another task, but by the tree construction itself; in fact, particles sent to another task are indexed by the latter as imported points. It is crucial to stress that this new task assignment is temporary and that imported points exist only within the tree framework: actual particles that moved out of the boundaries of a leaf remain in the task holding that leaf until a new domain decomposition is performed. From the DZS point of view, this means that directly erasing imported points is useless, as they will be eliminated anyway when the tree is erased from memory. Instead, the tasks which hold imported points need to flag them and send this flagging information to the tasks which physically hold the particles corresponding to imported points.

The role of an imported point and its behaviour in the DZS framework are graphically represented in **figs. 3.3a**, **3.3b**, **3.3d** and **3.3e**: the first of these figures represents

the state of the system of particles immediately after a new domain decomposition has been performed, with all particles inside the leaves of their task of belonging. In **fig. 3.3b**, corresponding to when the gravitational tree to walk is built, it is possible to notice that the leftmost particle belonging to task 1 has exited the leaf node (as shown by the faded green particle). Due to the periodic boundary conditions, that particle will be assigned to task 2 by the next domain decomposition; in the meantime, it shows up as an imported point in task 2 (green particle with blue border). This imported point is flagged with the particle content of node 3 in **fig. 3.3d** (as shown by that point now having a red border), but the faded green particle, still located in task 1, is not flagged yet. This flagging operation has to wait until the tree walk has been completed: in **fig. 3.3e**, task 2 has communicated to task 1 the fact that the imported point has been flagged, so the faded particle is marked in red and will be eliminated by task 1.

To describe in detail as clearly as possible how imported points are handled by the DZS algorithm, tasks will be hereafter labelled as "particle holders" and "imported point holders", depending on wheter they act on the particles they sent or those received as imported points; in fact, each task plays both roles, altough in different parts of the workfow. Relevant operations performed by both categories, even out of `walk_tree`, are shown in **fig. 3.7**. First of all, the DZS algorithm makes some adjustments to the imported point communication phase, which takes place during the gravitational tree construction: every task, now acting as a particle holder, stores the `no` value of each particle to be sent as an imported point in a dedicated array, and when the actual communication is performed, adds to the default packet to be sent the particle `ID` and the local task rank. By doing so, the imported point holder task which receives the communication has access to both these values, needed respectively to flag the imported point if needed and to know which task to send the flagging information to. Specifically, imported point holders create an array called `imported_IDs_to_remove` with size equal to the number of imported points received from every other task, and initialise every element to −1 (i.e. no valid particle `ID`) before `walk_tree` is called for the first time. During the tree walk, if an imported point is inside a branch of a node to be derefined (`erase_particles` $\neq 0$), a −1 element of `imported_IDs_to_remove` is overwritten by the `ID` of that imported point (more properly, of the particle associated to the point). After the walk, each value of `imported_IDs_to_remove` is sent back to the appropriate particle holder tasks. This operation is necessary because, as mentioned before, there would be no point in erasing the imported points locally, and instead each `ID` and −1 value (which signals that a particle needs to be eliminated or kept, respectively) needs to be communicated to

the correct particle holder task which stores the actual particles to flag. In fact, each particle holder creates an array called `imported_IDs_recv` in which it stores the received data. The content of this array is then compared to the `IDs` of the particles stored in the tree construction stage. When a match between the two is found (by construction, every non-negative value in `imported_IDs_recv` is guaranteed to have a match), the corresponding local particle is flagged for removal, in the exact same way as any other particle during the tree walk.

In summary, local particles linked to imported points of other tasks cannot be flagged directly during the walk; instead, they have to wait until every instance of `walk_tree` has returned and the imported point holders have sent the necessary data. Unfortunately, due to the fact that the elimination information for an imported point is located in one task and the corresponding particle in another task, it is necessary to resort to communications to ensure the correct behaviour of the DZS algorithm. In fact, since the number of imported points is usually very small (in the tree which the algorithm acts on, the ratio of this number and that of local particles is in the interval $[10^{-5}, 10^{-3}]$) their communications do not to affect the performance in a noticeable way.

3.2.3 Node derefinement

The end of the tree walk leaves some particles with mass, velocity components and `ID` all equal to zero, and some node indexes `no` flagged for removal. While the former will not actually be treated until the next domain decomposition (see next subsection), the latter are used to build new particles shortly after the walk. In fact, the first operations performed after the walk regard the orange boxes of **fig. 3.7**, which are all enclosed in the function `handle_imported_points`. After all these operations have been carried on, each flagged index `no` (which identifies a node to be turned into a particle locally) is passed to the function `make_particle_from_tree_node`; this function creates a new particle-like element and fills it with data located inside the `Nodes` structure. Specifically:

- the mass of the new particle is the mass of the node (i.e. of all of its former content);
- the three position components of new particle are the components of the center of mass **s** of the node;
- the velocity components of the new particle are the components of the center of mass velocity of the node, which are evaluated for the DZS algorithm during

the tree construction phase, similarly to how the mass and center of mass of each node are normally calculated;

- the type of the new particle is an user-defined integer value, which should be $\neq 0$, 4 and 5 (i.e. the type of gas particles, stars and black holes, respectively);
- the simulation time at which the new particle starts its evolution is the current simulation time;
- the specific particle timestep assigned to each new particle is the smallest one currently evolved, to avoid the risk of larger timesteps giving too rough of an approximation for the trajectory of the particle;
- the old acceleration of the new particle (i.e. the acceleration which it would have had in the previous timestep, if it existed) is the mimimum value of the old acceleration across every particle outside of the lightcone (this quantity is needed for the tree-opening dynamical criterion);
- the `ID` of the new particle is the one determined during the tree walk;
- the softening value of the new particle can be the specific user-defined value assigned to the DZS particle type; alternatively, the user can choose to scale the softening ε of every new particle according to its mass m as $\varepsilon = \varepsilon_{base}(m/m_{base})^{1/3}$, where m_{base} and ε_{base} are the mass and softening values of a type 1 simulation particle. This ensures that when the new particles are very massive they do not get really close to other particles, resulting in a strong gravitational interaction which would require many small timesteps to be accurately evolved in time (and since this interaction would take place outside of the lightcone, it is of no interest to describe it accurately).

When all these tasks have been completed, `make_particle_from_tree_node` inserts the newly created particle at the end of the local particle structure.

The operations described above all take place along with a few others in a function called `merge_nodes`, which is summarised in **fig. 3.8**. Furthermore, this function is inserted in a wrapper function for all the main processes carried on by the algorithm. The wrapper function is called at a specific time during each global timestep, and also takes care of particle elimination.

3.2.4 Particle elimination and insertion point

The description carried on so far has shown that the DZS algorithm modifies substantially the simulation domain; this makes it very important to appropriately in-

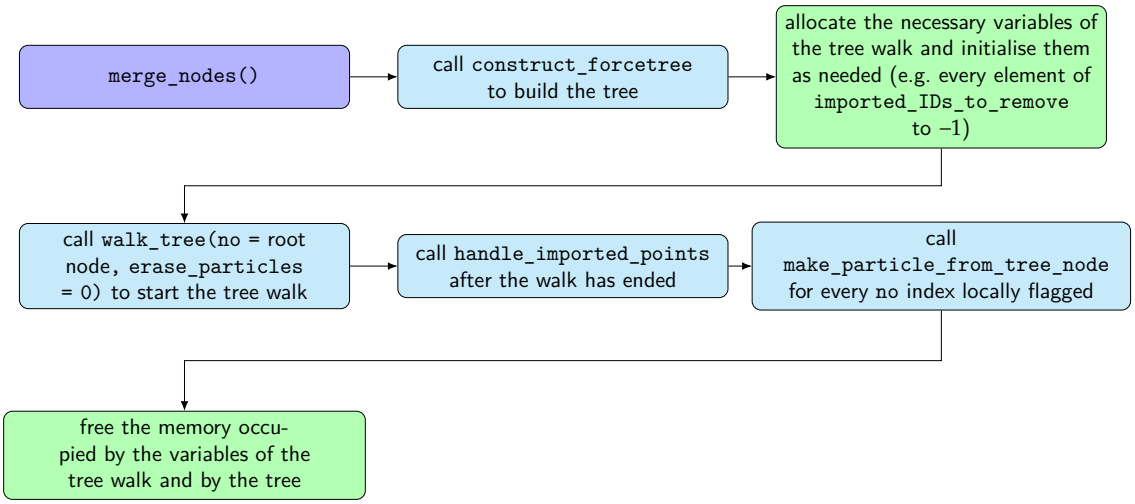


Figure 3.8: flowchart showing the operations performed by `merge_nodes`. The start of the function is marked in blue, operations in green and calls to other functions in cyan.

sert the algorithm into the workflow of the standard code, in order to avoid any incorrect behaviour. For example, the algorithm should be only allowed to operate during global simulation steps, because the particle merging operation requires every particle to be at the same simulation time. Moreover, for an optimal distribution of particles and of the work-load associated with them, it is necessary to perform a domain decomposition after the DZS algorithm has modified the number of simulation particles, because, for example, even if the particles inside a derefined node were distributed across multiple tasks, the new particle will be stored in a single one of those tasks. Therefore, a domain decomposition is performed as an attempt to evenly distribute the work-load; furthermore, the elimination of flagged particles is carried on within the decomposition, in the same place where other particle rearrangements are normally performed by the code (i.e. in the function `domain_rearrange_particle_sequence`).

The specific method employed to eliminate the particles also mimics pretty closely the ones adopted in other particle rearrangements: particles are indexed by an integer value p which varies from zero to the local number of particles `NumPart` (excluded) with increments of one. If the mass and `ID` of the particle corresponding to a specific value p_{spec} are equal to zero, that particle is swapped with the one indexed as `NumPart` −1 in the structure holding particle data, and `NumPart` becomes equal to itself minus 1. Afterwards, the new particle in the same position p_{spec} is checked to see if it also has mass and `ID` = 0, and is eventually swapped with the last particle. These checks on the same p_{spec} continue until the particle in that position is one not flagged for elimination, and the iteration goes on to $p_{spec} + 1$. When the iteration is

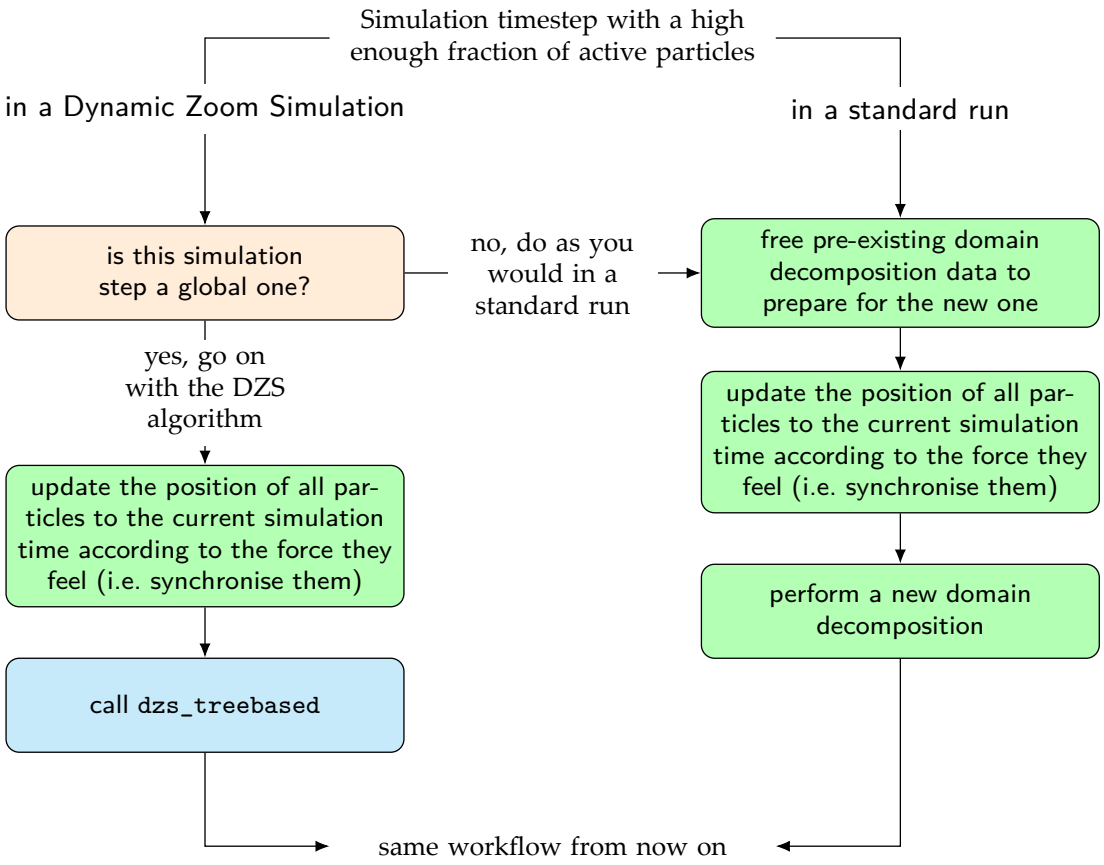


Figure 3.9: flowchart showing the workflow of a timestep from the check for enough active particles (in this case, the condition is satisfied) with and without the DZS compile time option. Checks are depicted in orange, operations in blue, and function calls in cyan.

over, all particles have been eliminated, and a global communication sums up the new local number of particles across all tasks, to yield the updated global number of particles in the simulation.

As mentioned before, a domain decomposition needs to be performed after the operations carried on by `merge_nodes`; however, it should be noted that a domain decomposition is one of the most resource-hungry processes in a cosmological simulation, so performing one just for the DZS algorithm would impact its performance gain; instead, it is more computationally efficient to insert the DZS operations where there is already a domain decomposition by default. In fact, domain decompositions occur roughly in the middle part[11] of certain simulation steps, i.e. those with a high

[11] In a gravity and hydrodynamics framework, each simulation step $\mathrm{d}t_i$ in AREPO is made of two gravitational half-steps (each yielding a gravitational time evolution of $\mathrm{d}t_i/2$) and a full hydrodynamic timestep between the two gravitational ones (which evolves hydrodynamically the system by a time $\mathrm{d}t_i$). This approach, namely a second order operator-splitting (Glowinski et al. 2016; Weinberger et al. 2020) formally holds even in a DM-only framework (i.e. without hydrodynamics), and the "middle

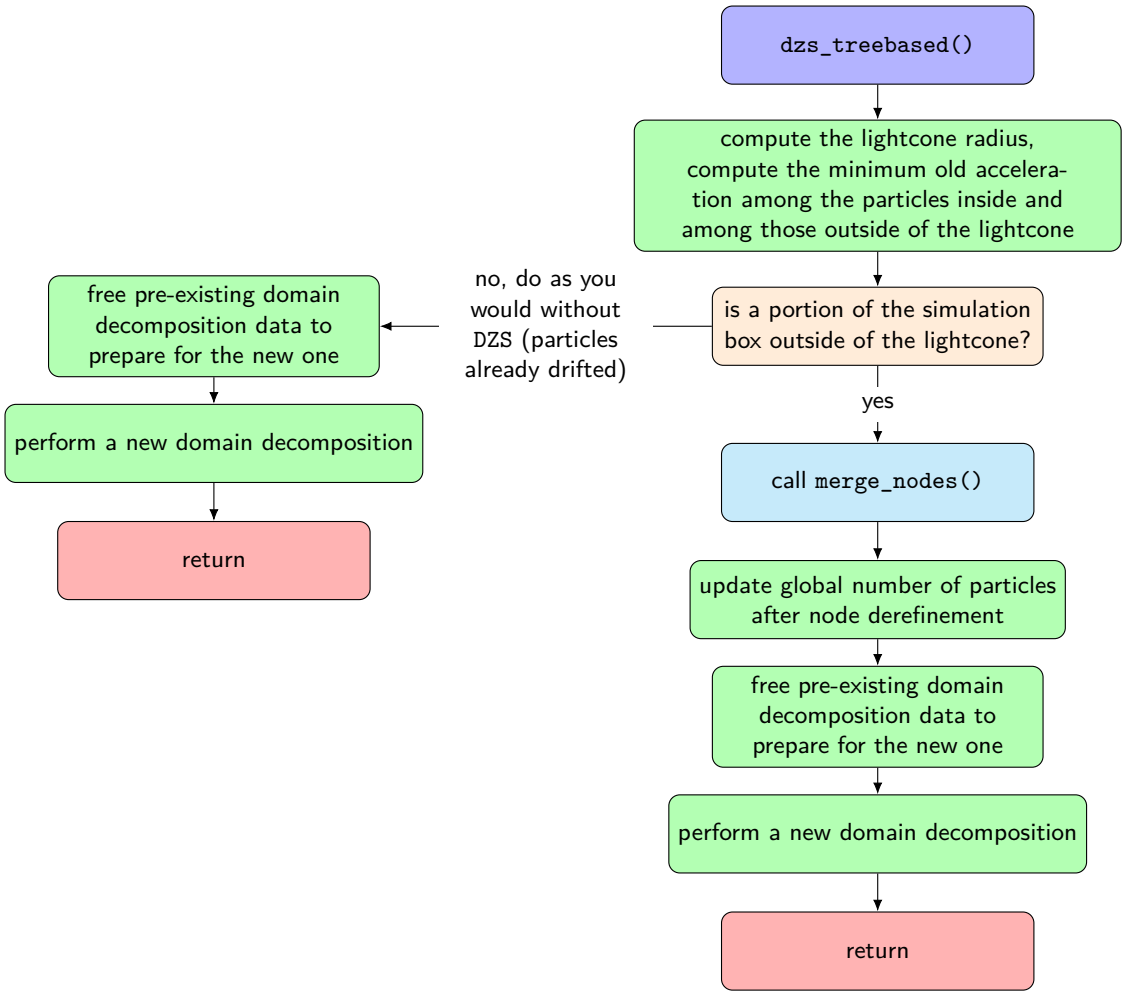


Figure 3.10: flowchart of the function `dzs_treebased`. The start of the function is marked in blue, operations in green, yes/no checks in orange and function calls in cyan.

enough fraction of particles (1% by default) which are evolved in that timestep. If the DZS algorithm is enabled in the simulation, the aforementioned wrapper function, called `dzs_treebased`, can be inserted in the framework of this domain decompostion when a simulation step is global, as shown in **fig. 3.9**. Note that the domain free operation and the subsequent decomposition are not explicitly included in the DZS case, because they are performed inside `dzs_treebased`. This detail leads to another difference between the two cases: the particle synchronisation operation is performed after the domain free operation in the standard framework and before it in the DZS framework. This ensures that the positions of particles analysed by the tree walk are actually updated to the same point in time and allows to locate new particles at that point, as is done in **subsec. 3.2.3**. It should be mentioned that swapping the synchronisation and domain free operations does not have any impact on

part" of a simulation step here refers to the portion between the two gravity half-steps.

the workflow, as the two processes are not consequential.

Aside from the domain decomposition, `dzs_treebased` performs other operations, as shown in **fig. 3.10**. First of all, the lightcone radius needs to be calculated (either by integration or linear interpolation, as described in **subsec. 3.2.1**), along with the minimum old acceleration among all particles inside the lightcone (needed if the dynamical derefinement criterion is employed) and outside of it (needed to be assigned to new particles, as mentioned in **subsec. 3.2.3**). As an optimization strategy, at first the newly calculated lightcone radius is then compared to the half-diagonal of the simulation box, to check if there is a portion of the latter which is located outside of the lightcone (i.e. a portion which the DZS algorithm can operate on). If this portion does not yet exist, `dzs_treebased` simply falls back to the standard domain free and decomposition operations, and then returns. On the other hand, if the lightcone has shrunk towards the observer enough to cross the boundaries of the simulation domain, the DZS algorithm is free to operate and the function calls `merge_nodes`. After this latter function returns, the total number of simulation particles needs to be updated before the usual processes related to the domain decomposition. After the latter has also took place, the wrapper function finally returns.

3.2.5 Additional considerations

The AREPO implementation described so far, albeit self-consistent, does not include some minor details which were omitted until now for the sake of clarity. For example, both the geometrical and the dynamical derefinement criterion offer the possibility to perform an additional check, i.e. that the side of the (cubic) node tested is smaller than a user-defined value. If that is the case, the node can eventually be derefined, otherwise it cannot, even if it satisfies the derefinement criterion employed. This measure prevents the derefinement of large nodes, so that the resolution outside of the lightcone does not become too low to sample the large-scale gravitational field with a reasonable accuracy. In fact, the evolution of this field still needs to be followed to some degree because of the influence that it has on the particle content of the lightcone.

Another detail included in the DZS algorithm implementation addresses the issue of not having any "zoomed" portion of the simulation domain (i.e. a portion at the original resolution of the simulation) at $z = 0$, because $R_{lc}(z = 0) = 0$. Generally, it might be useful to have such a volume even at zero redshift for output analysys purposes. The DZS algorithm grants the possibility to have such a volume through a user-defined buffer zone $b > 0$, which increases the lightcone radius and is used in

every check related to that radius; formally, this implies a substitution of the quantity R_{lc} in **eqs. 3.1** and **3.2** with $R_{lc} + b$, as if the real lightcone radius was not the former quantity but the latter. Note that this also applies to the yes/no check depicted in **fig. 3.10**. In fact, the buffer zone also helps in reducing the impact of node derefinement on the actual lightcone, even when employing an aggressive derefinement criterion.

Last but not least, the algorithm also sets a lower limit to the particle timestep of every particle outside of the lightcone, as a precaution measure against the situations where particles get too close as a consequence of derefinement. In fact, rescaling the gravitational softening according to the mass of a particle (**subsec. 3.2.3**) should already avoid most of these situations, but this timestep limit was originally introduced in the `PGADGET-3` implementation and it was carried over to `AREPO`, to tackle those small timesteps which rarely slip past the rescaled softening. Specifically, the limit is set to the smallest timestep inside of the lightcone, and is calculated by the DZS algorithm before the first gravitational simulation half-step. This calculation also assigns a flag to each particle, signaling if it is inside or outside of the lightcone radius. The flag is then used during the particle timestep calculation within the first gravity half-step, and if a particle is outside of the lightcone, its timestep can be limited as needed.

These details complement the description of the DM-only DZS algorithm implementation. However, `AREPO` is tailored for simulations which include baryon physics, which in fact requires most of the computational resources of the run. Therefore, employing the aforementioned tree-based algorithm (which can only act on DM particles) in this kind of simulations, altough being technically possible, would leave the gas content of the simulation untouched and not yield any performance gain from operations performed on that content. For this reason, it is appropriate to extend the capabilities of the DZS algorithm to include baryon derefinement. In fact, the moving mesh approach employed in `AREPO` provides a way to treat baryons in the DZS framework.

3.2.6 Baryon derefinement

As mentioned in **subsec. 2.2.2**, one major advantage of a Voronoi tessellation is to avoid the cell tangling which can arise when trying to follow the evolution of a fluid with a structured moving mesh. It should be noted, however, that the time evolution of an unstructured mesh such as the Voronoi one can yield cells with significant mass and/or size deviations with respect to the initial values, hindering the spatial adap-

tivity of the scheme. To address this issue, AREPO offers the possibility to refine or derefine mesh cells, respectively by splitting a mesh-generating point into two separate ones which are then displaced by a small quantity in random directions, and by "dissolving" the fluid properties within a cell into adjacent ones (Weinberger et al. 2020); in fact, the latter process provides a built-in baryon derefinement that the DZS algorithm can take advantage of. Specifically, the standard version of AREPO adopts by default a mass criterion to select cells with a mass lower than a set value and derefine them. Adapting this scheme to the DZS algorithm requires the introduction of an additional criterion to test mesh-generating points with, namely the geometrical criterion of **eq. 3.1**. The only quantity appearing in that equation which is not explicitly provided by the code is the cell "length" L, which is not properly defined since, unlike tree nodes, mesh cells are generally not cubic. Nonetheless, an estimate of L to insert in the baryonic derefinement criterion can be provided by $L = [3/(4\pi)V]^{1/3}$ (where V is the cell volume), under the assumption of spherical cells. The quantity L is also compared with the maximum allowed node size to avoid creating mesh cells too massive and/or with an excessive volume.

It is now appropriate to describe how this additional criterion fits within the general workflow of mesh refinement and derefinement, as well as within the workflow of the DM-only, tree-based DZS algorithm. In fact, the DM and baryon derefinement DZS algorithms can act independently of each other, with the only change with respect to a DM-only simulation occurring in the tree construction stage of **fig. 3.8**: to ensure that the properties assigned to new particles (e.g. mass and position) are correctly referred to the dark matter content of derefined nodes, the function `construct_forcetree` is set to only use DM particles to build the tree.

Regarding the standard mesh derefinement and refinement operations, it is worth mentioning that they can be enabled or disabled in a simulation thanks to dedicated options, and the additional criterion employed by DZS is set to work alongside these options, if present. More specifically, consider a global timestep of a simulation which includes both dark matter and baryons: in the function which normally tests a mesh generating point with the default mass criterion, the first criterion which is actually employed is the DZS geometrical one. If the point satisfies the criterion, it is flagged for derefinement and the corresponding cell will be dissolved into adjacent ones (the actual mesh generating point will be eliminated in the next domain decomposition). On the other hand, if the point does not satisfy the criterion, the testing function follows the standard workflow of AREPO. In comparison to the standard mesh refinement, the only modification needed in the DZS framework is to set the refinement process to only take place inside of the lightcone. This avoids the

refinement of mesh generating points outside of the lightcone, where the mesh resolution has been lowered by the DZS geometrical criterion. Without this limitation on mesh refinement, the code would constantly try to increase the resolution of the mesh while the DZS algorithm would try to achieve the opposite result, resulting in a waste of computational resources.

The above implementation of baryon derefinement in the DZS framework raises the question of why this derefinement was not carried on in a tree framework, similarly to the DM case. The answer lies in the fact that the Voronoi mesh is much more complicated to modify than the oct-tree. First of all, creating new mesh generating points from nodes of a (baryon only) oct-tree could alter the mesh geometry and lead to excessive distortions (for example, if the center of mass of a node to derefine is far from its geometric center). Second of all, each cell of the mesh stores fluid properties which need to be correctly assigned to new particles once those cells are eliminated. Performing this operation is far from trivial and also requires to recursively calculate fluid properties in the tree construction framework (similarly to the center of mass velocity in **subsec. 3.2.3**), as well as to find a way to define the values assigned to new particles. In short, it is safer and more natural to employ the mesh derefinement structure already included in `AREPO`, as the dissolving procedure of derefined cells distributes the fluid properties to neighboring cells and does not generally lead to major mesh distortions. This choice, however, is not without its flaws: one major downside of the employed mesh derefinement method is that, unlike the tree, the Voronoi tessellation is not a hierarchical structure: in the DM-only case the derefinement process could quickly turn many particles into one, if for example a large node with a high particle content was flagged for derefinement in a high tree level. In the Voronoi mesh, on the other hand, there are no levels and every mesh generating point has its own cell. This means that every point has to be tested with the derefinement criterion; moreover, the elimination process itself is quite slow, since the code prevents neighboring cells from being derefined at the same time (otherwise there could be missing neighboring cells to dissolve fluid properties in). In fact, there is no particle creation stage in this baryonic derefinement framework; instead, individual particles are slowly eliminated over the course of a simulation. This translates into an iterative derefinement process which spans many timesteps for the same volume of the simulation and takes up a significant amount of computational resources, also because the DZS geometrical criterion is much easier to satisfy with respect to the standard mass criterion and thus many more mesh cells are derefined in a Dynamic Zoom Simulation than in a standard one.

Whether or not a mesh-based derefinement is actually better than a tree-based

one is an open question which would require, first of all, a complete implementation of the latter approach and, second of all, output generation tools which allow an extensive analysis of DZS-related approximations in a full-physics framework. As will be explained in the next chapter, a meaningful comparison between runs with and without DZS requires lightcone-like output, which in AREPO, at the moment, does not include any hydrodynamical properties of saved particles. For this reason, and also because of some minor issues currently affecting the DZS-related baryon derefinement, I have chose not to include in this work results related to that derefinement, and instead to present them when an appropriate and complete analysis will be possible. Nonetheless, the next chapter presents a comprehensive study of DZS-related approximations and performance improvements in the DM-only scenario, showing that the present state of the algorithm is indeed accurate and safe to use and that it laids solid foundations for full-physics Dynamic Zoom Simulations.

4 | Algorithm validation

After explaining the fundamental reasons behind the need for a performance improvement in cosmological simulations, and after describing in detail how this work tries to achieve that improvement, it is finally time to check how the DZS algorithm performs in practice. This is carried out by performing pairs of runs ("twin" simulations) which have the exact same initial conditions (ICs, i.e. box side, number and phase-space distribution of particles), with one of them also employing the DZS algorithm; in fact, I have repeated such pairs of simulations with a variety of initial conditions. All these runs will be used to quantify the differences in the mass and density distribution, as well as in the positions of individual particles, that a Dynamic Zoom Simulation has with respect to a standard simulation thanks to a variety of output data, mainly in a lightcone-like form. In fact, a direct comparison of traditional output types, namely of the whole content of the simulation domain at a fixed time, would be impossible, since the DZS algorithm modifies the domain by changing the number of particles. Nonetheless, a direct comparison is meaningful inside the lightcone, where the number of particles and their physical properties should remain approximately unchanged in Dynamic Zoom Simulations and the only source of differences with respect to a standard run (that is, the approximations introduced by node derefinement) is the large-scale gravitational field generated outside of R_{lc}. Obtaining a numerical estimate of these approximations is crucial to verify the correct behaviour of the DZS algorithm (i.e. "validate" the code). Once it is verified that the approximations do not lead to significant modifications of the output data, it is possible to finally analyse how the DZS algorithm impacts the performance of the code in terms of run time and work-load balance.

4.1 Output analysis

The suite of simulations employed here is similar to those of Llinares (2017) and Garaldi et al. (2020) (so that the results can be easily compared), and is tailored to study how the DZS algorithm performs with different box sizes and resolutions.

name	L [$Mpc\ h^{-1}$]	N	M_{part} [$10^{10}\ M_\odot\ h^{-1}$]	ε [$Mpc\ h^{-1}$]
MEDIUM	2048	512^3	552.55	0.1
MEDIUMHR	2048	1024^3	69.07	0.05
LARGE	8192	512^3	35363.4	0.4
LARGEHR	8192	1024^3	4420.42	0.2

Table 4.1: simulations employed in the DM-only algorithm validation, where L is the box side, N is the number of tracer particles, M_{part} is the mass of each particle and ε is the softening length.

The four tested simulations, all of which are DM-only, are listed in **tab. 4.1**. The main difference with respect to the suite employed in Garaldi et al. (2020) lies in the substitution of the DUSTGRAIN simulation ($L = 2000\ Mpc\ h^{-1}$, $N = 2048^3$, not used here due to the large amount of resources required to perform such a run) with the new box MEDIUMHR; with a large number of particles in a relatively small box, the MEDIUMHR simulations represent the highest resolution available in the suite employed here.

The evolution of each simulated box is followed from $z = 50$ to $z = 0$ in a gravity-only framework, with a gravitational softening given by the reference formula $\varepsilon = (XL/N^{1/3}) = X\lambda$ (e.g. Zhang et al. 2019), with $X = 1/40$ being a fraction of the mean interparticle separation λ (particles created by the DZS algorithm will have their softening rescaled according to their mass, as mentioned in **subsec. 3.2.3**). The quantity λ is also used to define DZS-related run time options: the radius of the buffer zone and maximum linear node size (**subsec. 3.2.5**) are set at 5λ and 4λ, respectively. In practice, the latter value ensures a minimum resolution outside of the lightcone which is equivalent to that of a simulation with 1/64 of the original number of particles. Additionally, every Dynamic Zoom Simulation employed here uses a geometrical derefinement criterion with an opening angle $\theta_{geom} = 0.1$.

As mentioned before, the only direct way to compare output from a run with the DZS algorithm (herafter, "dzs" run) and a run without it (hereafter, "std" run) is by using lightcone-like data, whose method of acquisition, i.e. the use of thin spherical shells with radius equal to R_{lc}, was described at the beginning of **chapter 3**. In the DZS version of `AREPO`, this data is collected on the fly, resulting in a large reduction of the necessary storage space and input/output time in the simulation. I ported the code which saves lightcone-like data during the simulation to the public version of `AREPO` following the approach described in Fosalba et al. (2008), with some improvements leading to an easier handling of the output files. Specifically,

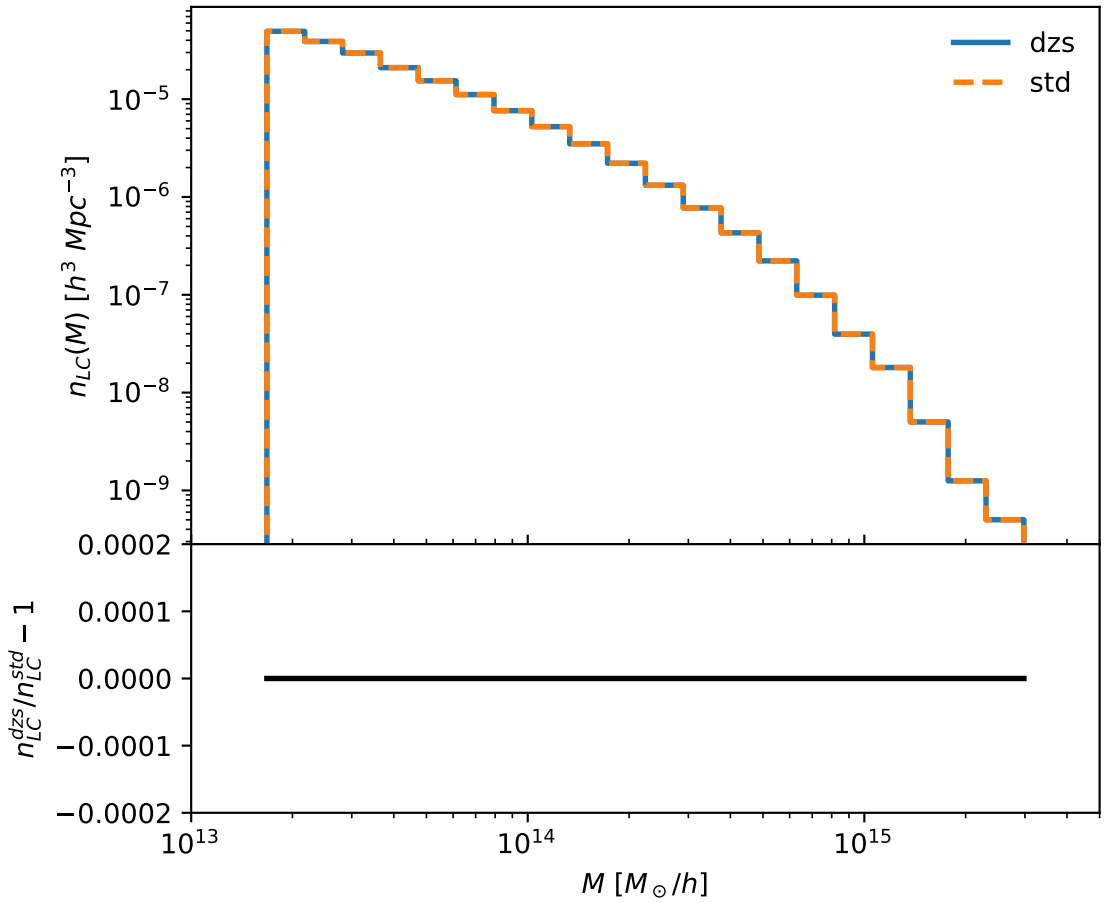


Figure 4.1: Lightcone Halo Mass Function for the twin MEDIUMHR simulations (top panel); the bottom panel represents relative erros in number density. In the bottom panel, the relative differences are always exactly zero, which is why the y scale has been chosen arbitrarily.

the code saves to disk the particles located in spherical shells with outer radius R_{lc} during selected simulation steps, with the inner radius of each shell being equal to the outer one of the last saved shell. In fact, this on the fly saving system also allows to arrange the shells into HEALPix maps (Górski et al. 2005), effectively producing a sky-projected lightcone output. Moreover, traditional outputs can be produced along with lightcone-like ones: for example, halo catalogs, i.e. files containing data regarding dark matter halos identified in the simulation domain by the so-called Friends-Of-Friends algorithm (FOF, first used with this purpose in Davis et al. 1985), are generated alongside lightcone shells in the MEDIUMHR twin simulations. I used these catalogs (which, similarly to a snapshot, store data collected at a fixed simulation time) to build the Lightcone Halo Mass Function (LCHMF), which is the first type of lightcone-like output analysed in this validation.

4.1.1 Lightcone Halo Mass Function

The Lightcone Halo Mass Function relates the mass and number density of the DM halos located in points of space-time which allow them to be indirectly detected by the observer through the propagation of light; in the piecewise-constant approximation employed here, these halos are inside the lightcone shells. The output data needed to create the LCHMF is only generated for the MEDIUMHR simulations because their relatively high resolution allows the FOF algorithm to identify the largest amount of structures and offer a larger sample of halos to be compared between the dzs and std runs, therefore providing the most stringent test possible with this simulation suite. Moreover, a box side $L = 2048\ Mpc\ h^{-1}$ allows the DZS algorithm to operate from $z \simeq 0.69$ to $z = 0$, i.e. nearly half of the timespan covered by the simulation in the employed ΛCDM framework. In fact, the LCHMF has been created from a slightly lower redshift (when the lightcone started being completely inside the simulation domain, to avoid the need for any box replications), namely $z \simeq 0.36$. In practice, the LCHMF was built by extracting haloes whose centers of mass are in adjacent thin shells with outer radius corresponding to the lightcone radius at the time the halo catalog was produced. While this does not ensure that every selected halo is entirely inside a lightcone shell, it guarantees that each catalog entry is included in at most one shell, so that the resulting LCHMF does not have any duplicate halos. The number $N(M)$ of selected halos as a function of mass M is estimated for an arbitrary number of mass bins (twenty, in this case), and each value $N(M)$ is divided by the volume of the sphere with radius equal to the lightcone radius when the first catalog was saved, to finally yield a number density estimate n_{LC}.

The LCHMF comparison for the twin MEDIUMHR runs is depicted in **fig. 4.1**. The first noticeable thing is the limited range of the x axis: despite the fact that the MEDIUMHR runs have the best resolution among the simulations listed in **tab. 4.1**, each DM particle starts with a mass $M_{part} \simeq 7 \cdot 10^{11}\ M_{\odot}$, which limits the mass range the FOF algorithm can operate on. Nonetheless, in the top-end view of the LCHMF provided by the MEDIUMHR runs, the std and dzs simulation yield identical results. This is a somewhat expected outcome, because even though the domain modifications due to the DZS algorithm are indeed significant in the MEDIUMHR simulations, they arise relatively late with respect to those occurring in larger boxes. These boxes include, for example, those employed in the LARGE and LARGEHR runs, where domain modifications and subsequent approximations should generally be higher. In fact, the LARGEHR pair of simulations will be included in the next subsection, which concerns the analysis of HEALPix output.

4.1.2 Sky-projected lightcone

The acronym HEALPix stands for Hierarchical Equal Area isoLatitude Pixelization (Górski et al. 2005), and it constitutes a tool employed in discretizing data on the surface of a sphere across pixels with the same area. Its main purpose is to allow an easier analysis of output data (both real and simulated) which spans the whole sky (i.e. as if the observer was in the center of a sphere), typically through a 2D angular projection. In this algorithm validation framework, each spherical lightcone shell is sky-projected and discretized with HEALPix. This procedure yields a pixelized spatial mass distribution of each lightcone shell, and when the mass content of each pixel is summed over every lightcone shell, the result is a lightcone-like mass distribution spanning a certain redshift range. This distribution has been generated for the MEDIUM, MEDIUMHR and LARGEHR simulations, with the latter having a redshift range of $z \in [\simeq 1.44, 0]$ without the need for any box replications. The other two pairs of simulations span the same range as the MEDIUMHR in the LCHMF subsection, i.e. from $z \simeq 0.36$ to $z = 0$. The number of pixels in the HEALPix distributions generated on the fly is the same for all simulations, namely $12582912 = 12 \cdot 1024^2$ or, equivalently, the N_{side} parameter is equal to 1024. For visualization purposes, during the output analysis the pixelization was downsampled to $N_{side} = 256$, i.e. 786432 pixels.

The sky-projected mass distributions of all the examined pairs of simulations are depicted in **fig. 4.2** (MEDIUM), **fig. 4.3** (MEDIUMHR) and **fig. 4.4** (LARGEHR). Each figure includes the distribution produced by the std run, the dzs run, the distribution of relative differences bewteen the two and of relative differences in the power spectrum C_l. This latter quantity is the spatially averaged difference between two points in a mass distribution as a function of their angular distance, expressed through the parameter l (linked to an actual angle θ roughly as $l \approx 180^\circ/\theta$). First of all, the first two figures refer to simulations sharing the same box size, and thus the same degree of modifications on the simulation domain by the DZS algorithm. The two pairs of simulation do, however, have different resolutions, which yield different degrees of accuracy. It should be noted that there are more pixels with a non-zero relative difference in the MEDIUM simulations with respect to the MEDIUMHR ones, meaning that a lower resolution results in more particles whose position is displaced over the pixel size due to the modified gravitational field. Moreover, in the MEDIUM runs those pixels also have higher relative difference values. A possible explanation for this lies in the fact that the MEDIUM runs have less particles than the MEDIUMHR ones to generate a mass distribution over the same number of pixels; therefore, every pixel has less counts on average and displacements over the pixel size (which generate non-zero

differences in the distributions) tend to weigh more. Anyway, in both cases the displacements are generally smaller than 2%. In fact, the few pixels with a non-zero relative difference in **fig. 4.3** have values always smaller than 0.3%, yielding very contained relative differences also in the angular power spectrum ($|\Delta C_l/C_l| \ll 10^{-4}$). The power spectrum differences are also very small in the MEDIUM runs of **fig. 4.2**, with $|\Delta C_l/C_l|$ being always $< 10^{-3}$.

As for the LARGEHR results depicted in **fig. 4.4**, there is a much higher impact of the DZS algorithm on the relative difference distribution due to the bigger box: first of all, there are many pixels which yield a non-zero difference, although with the usual small values (mostly below 1%). In fact, these values are even smaller than those of **fig. 4.2**. According to the comparison between the MEDIUM and MEDIUMHR results and given the fact that the MEDIUM runs actually have a resolution 8 times higher than the LARGEHR ones, the difference distribution in **fig. 4.4** should have higher values than that in **fig. 4.2**. The bigger redshift range employed in the LARGEHR results, however, includes more lightcone shells in the mass distributions and increases the average pixel count, resulting in a lower weight of displacements over the size of a pixel. This remark is supported by **fig. 4.5**, which depicts the LARGEHR results over the same redshift range as the MEDIUM and MEDIUMHR ones. In this case, displacements over the pixel size are actually more and have higher values than in **fig. 4.2**, which is the expected outcome from different mass resolutions; the power spectrum also has larger differences, which however never exceed the percent level. When considering the same spectrum over the larger redshift range of **fig. 4.4**, the LARGEHR runs behave marginally worse than the MEDIUM ones, as the combined effect of a larger number of non-zero pixels with lower values in the difference distribution yields values of $|\Delta C_l/C_l|$ at most slightly larger than 10^{-3}. In summary, DZS-related approximations are very contained in all the examined HEALPix output and provide evidence of the uselfulness of the algorithm.

4.1.3 3D lightcone

While the HEALPix projection described above allows an all-sky view and comparison of simulated results, it is also appropriate to analyse more directly the spherical lightcone shells. The combined particle content of all the shells produced in a run is sliced along the z axis (the slice is 30 *Mpc* thick), and a density field is created from the particles in the slice (similarly to how that field is obtained in the SPH framework, see **subsec. 2.2.1**). A 2D conic slice of this field is plotted over discrete pixels in **fig. 4.6** for the MEDIUM simulations and in **fig. 4.7** for the MEDIUMHR simulations,

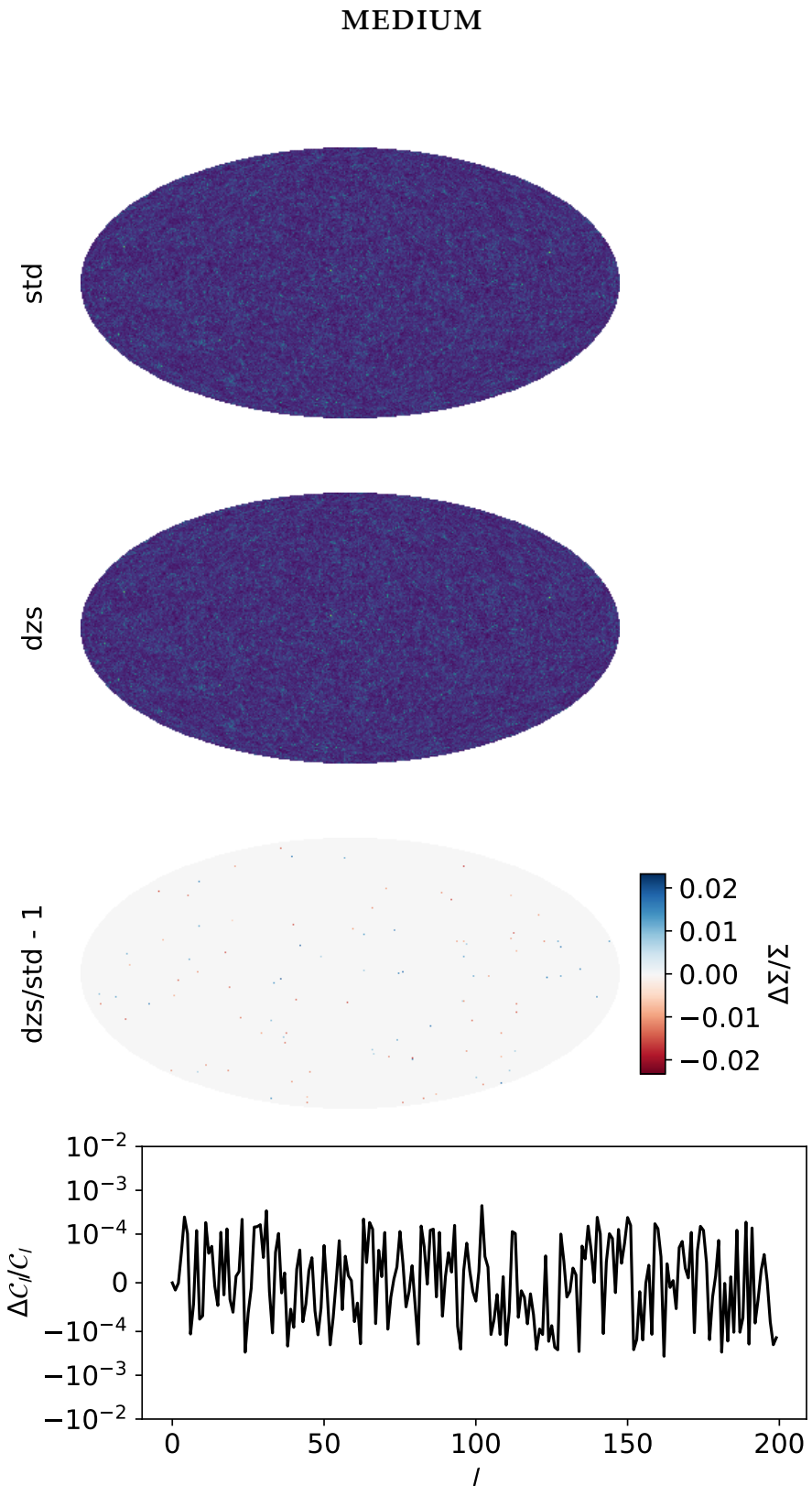


Figure 4.2: sky-projected mass distribution Σ of the std and dzs MEDIUM runs (top two panels, arbitrary units, darker colors corresponding to lower pixel counts). Third panel: distribution of relative difference between the std and the dzs results (the scale is symmetric with respect to the error with the highest absolute value). Fourth panel: plot of relative difference in the angular power spectrum C_l , with linear scaling in the y axis for $|\Delta C_l/C_l| < 10^{-4}$ and logarithmic scaling otherwise.

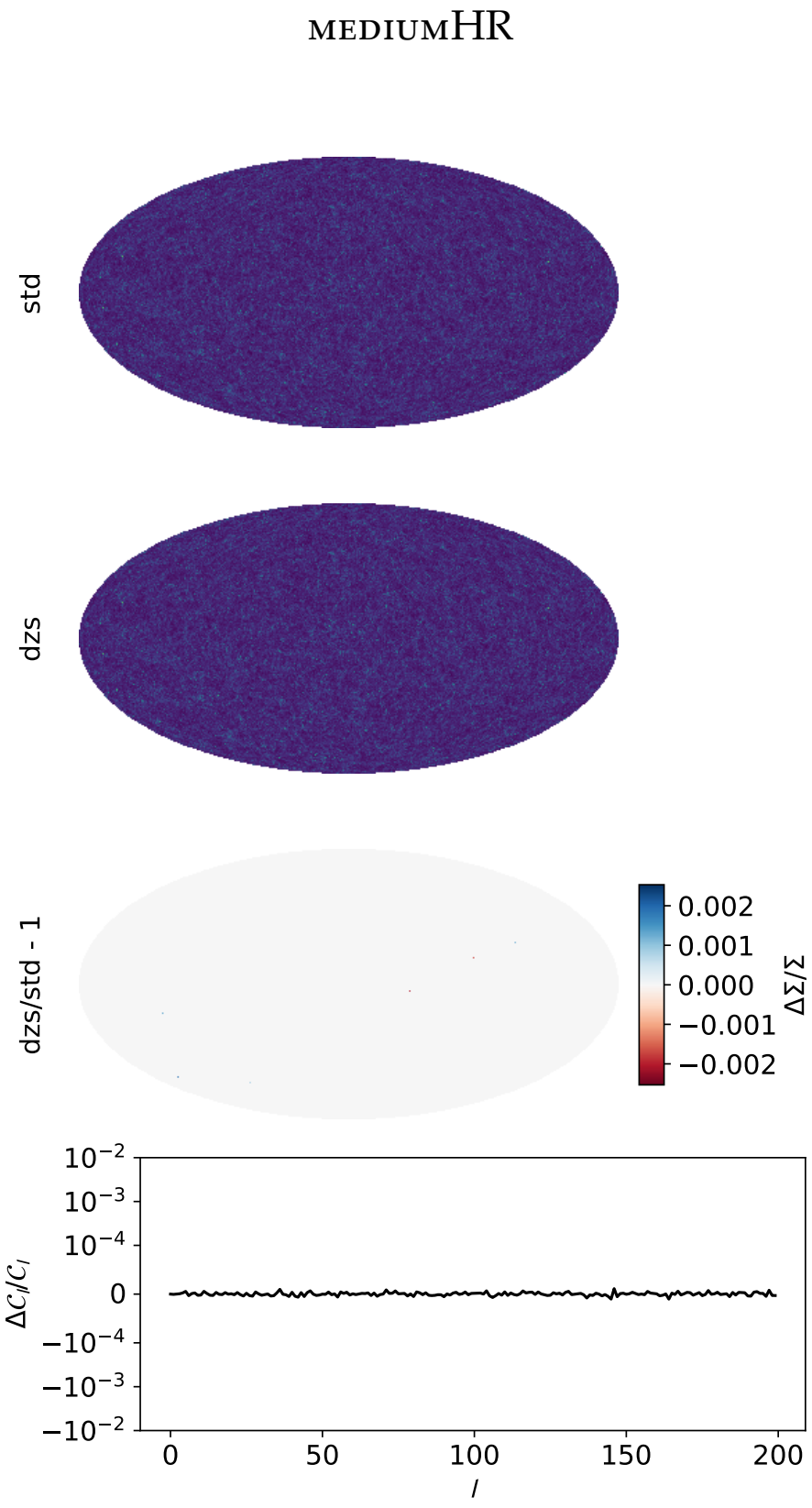


Figure 4.3: same as **fig. 4.2**, but for the MEDIUMHR runs.

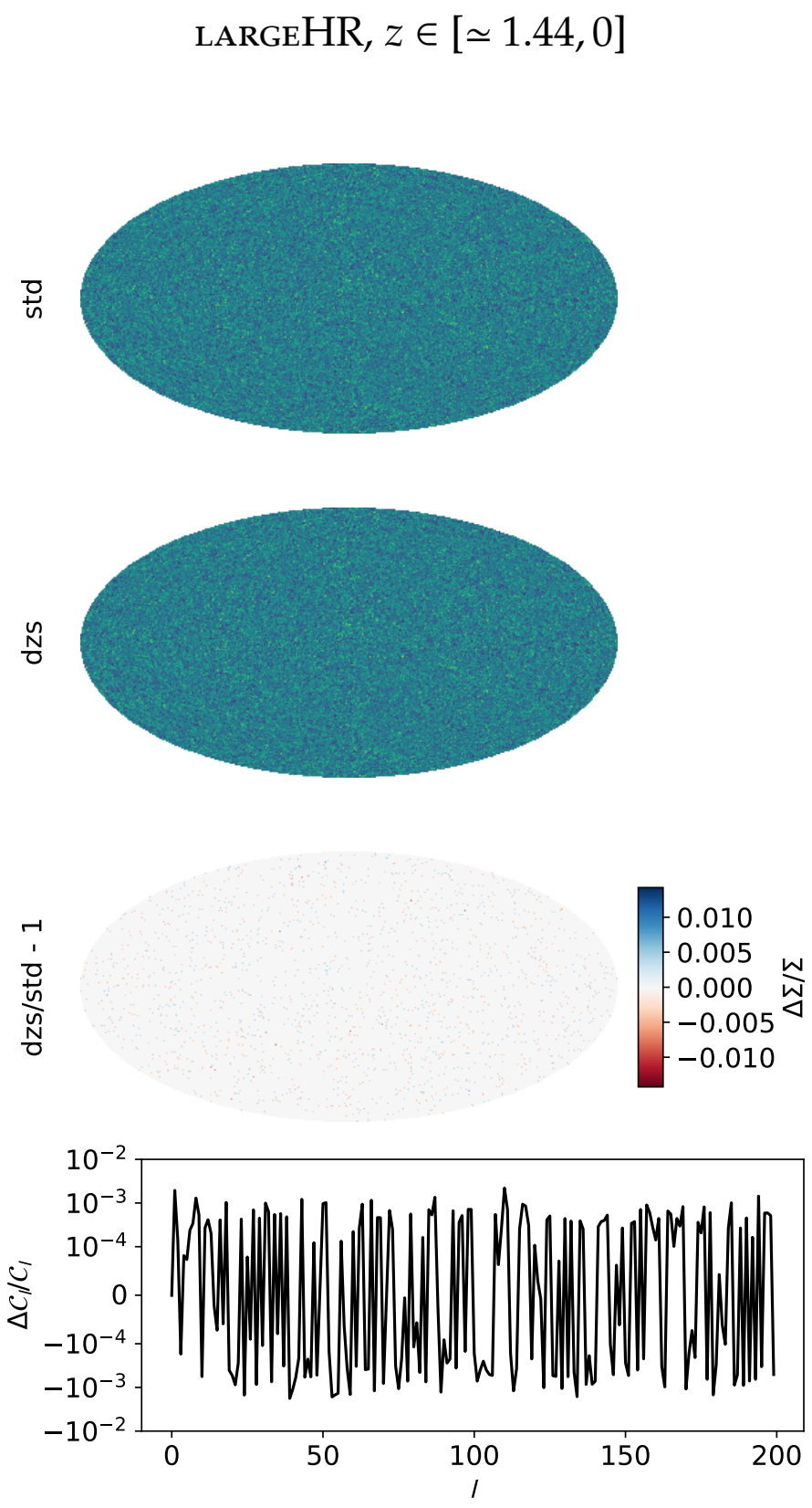


Figure 4.4: same as **fig. 4.2**, but for the LARGEHR runs from $z \simeq 1.44$ to $z = 0$.

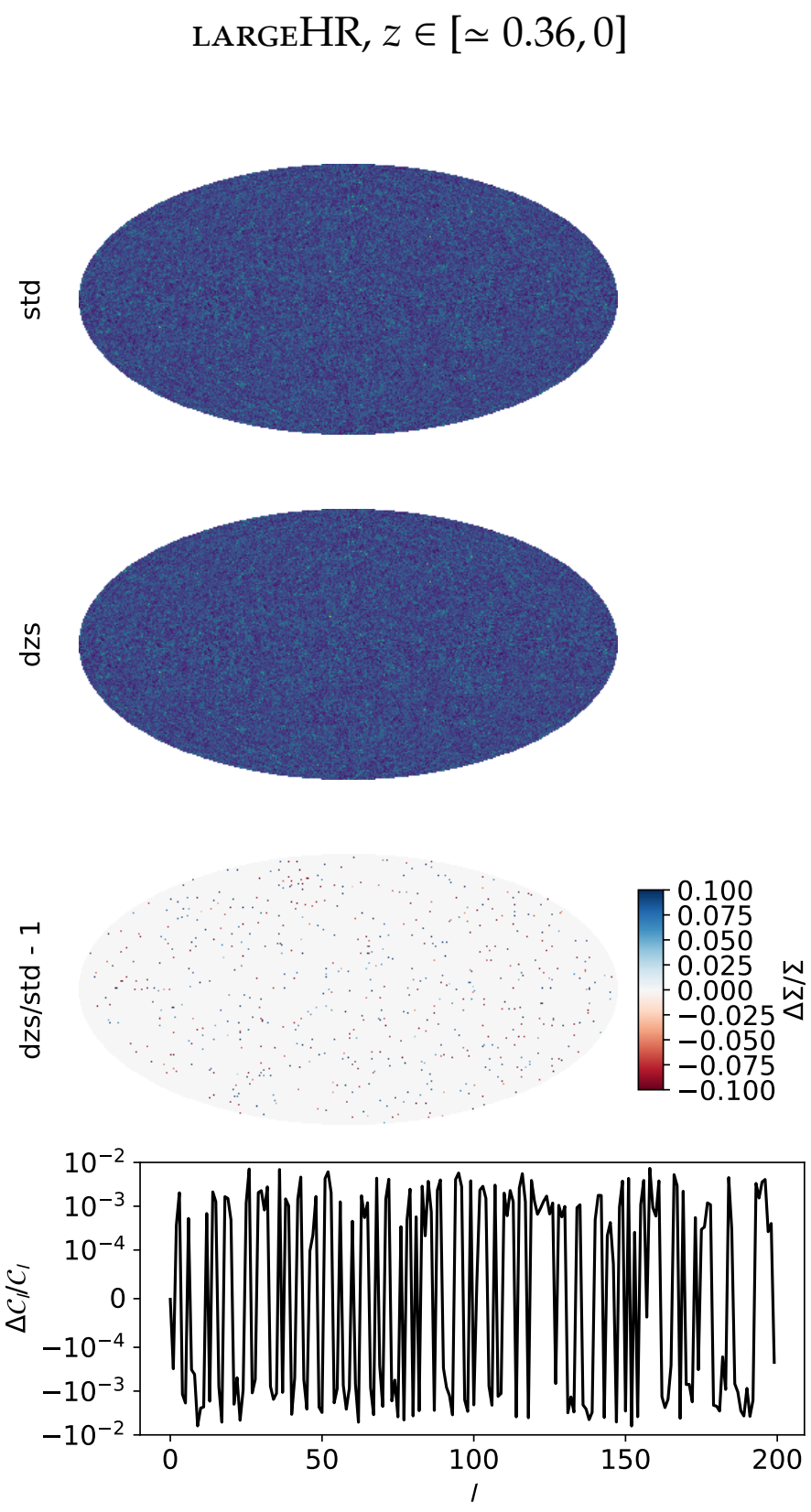


Figure 4.5: same as **fig. 4.2**, but for the LARGEHR runs from $z \simeq 0.36$ to $z = 0$.

with the lightcones spanning the same redshift range as in the previous subsection, namely $z \in [\simeq 0.36, 0]$. The figures also include a relative difference distribution similar to those seen in the previous subsection. At first glance, the dzs and std results appear very similar (if not identical) for both pairs of simulations, with the characteristic web-like arrangement of large-scale structures. A more quantitative inspection comes from the relative difference distribution, which reveals, for example, that in both figures non-zero differences arise in the inner portion of the conic slice. This result can be explained by considering the way in which lightcone shells are produced during the run: the innermost shells with low R_{lc} values are collected in later simulation steps with respect to the outer shells, which means that the particle content of the former shells has been affected by DZS-related modifications for a longer simulation time than the latter shells, hence the increase of relative deviations close to the observer.

As for the values of these deviations, they are mostly sub-percent in both the MEDIUM and MEDIUMHR simulations, with the latter showing very few non-zero spots in the relative difference distribution. This behaviour is coherent with the sky-projected results of the previous subsection, where the higher resolution of the MEDIUMHR runs led to fewer and smaller differences than in the MEDIUM case. In fact, this latter pair of simulations shows a significant number of non-zero spots in the relative difference distribution, which also has a couple of strange "artifacts" where an area with positive values is surrounded by a ring of negative values (or vice versa), such as the one highlighted in the zoom box of **fig. 4.6**. This effects is likely to be caused by the tools employed to build the smoothed density field from a particle distribution with a relatively small number of elements. In fact, when this technique is applied to simulations with lower particle counts (as the LARGE and LARGEHR runs, both having a worse resolution than that of the MEDIUM ones), the resulting density field closely mimics the distribution of actual particles without a significant smoothing effect, as a consequence of the lower particle density in the simulation domain.[1] Comparing these density fields in a relative difference map yields more ring-like artifacts, strenghtening the conclusion that the artifacts are related to a low resolution.

Overall, the error analysis has shown that the DZS algorithm is capable of reproducing the lightcone-like results of a standard simulation with high accuracy, which is expected to get even higher when dealing with a larger number of particles while keeping the box side fixed.

[1] It should be mentioned that this poor density field rendition is the reason why the LARGE and LARGEHR results are not included in the 3D lightcone analysis.

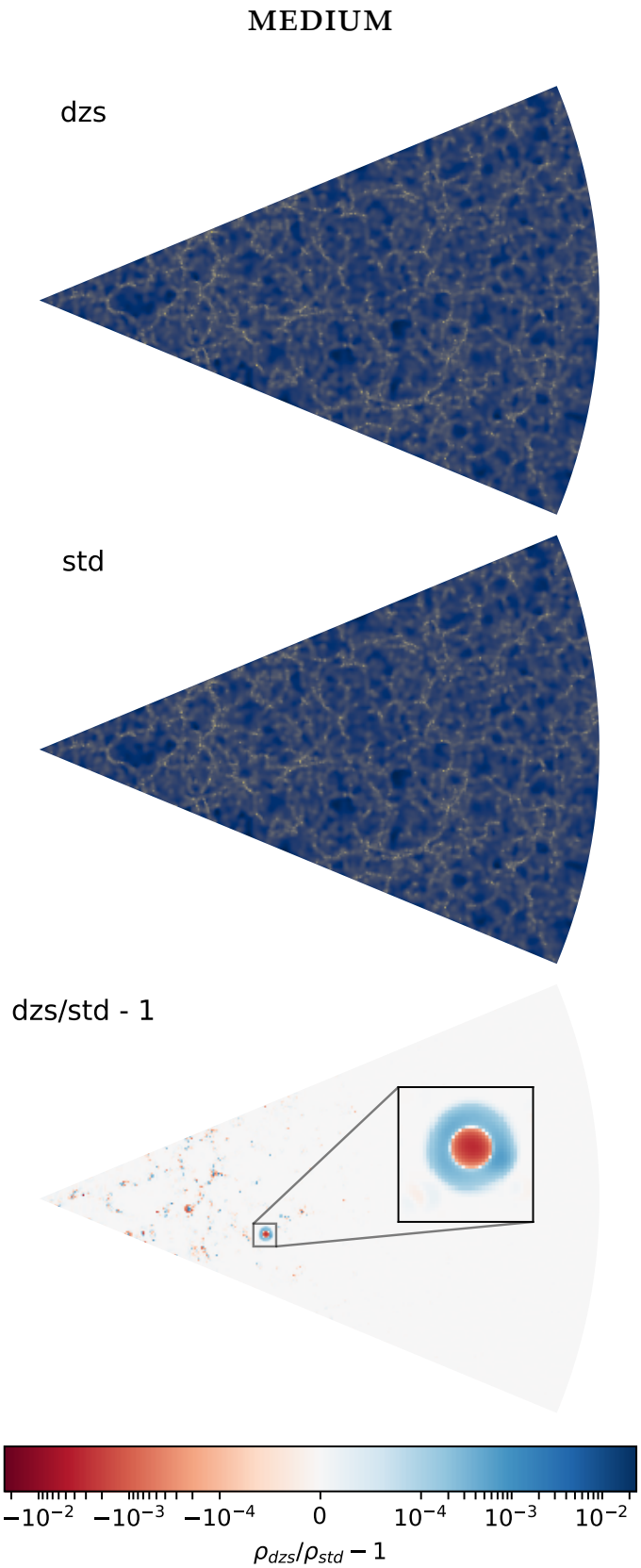


Figure 4.6: density distribution for a 2D conic subset of a thin slice (30 *Mpc* thick) obtained from the 3D lightcone of the dzs and std MEDIUM simulation (top two panels, arbitrary units, darker colors corresponding to lower pixel counts). Third panel: distribution of relative differences between the dzs and std results (the scale is symmetric with respect to the error with the highest absolute value). The zoom box highlights the presence of a ring-like artifact.

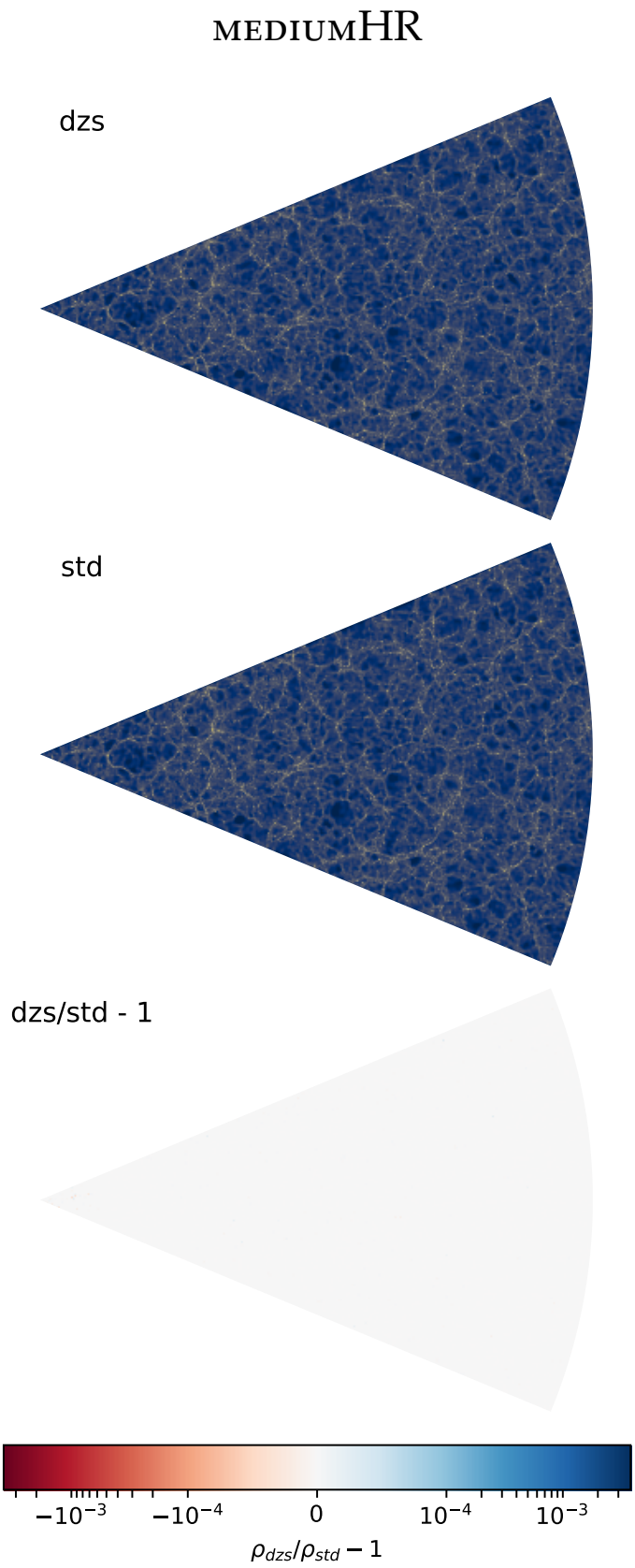


Figure 4.7: same as **fig. 4.6**, but for the MEDIUMHR runs and without any zoom boxes.

4.1.4 Particle displacements

While the above subsections have shown that the DZS algorithm yields very contained differnces with respect to a standard run and can be considered safe to use, a complete investigation should include the most strict comparison possible, i.e. that which is performed between pairs of single particles in a twin set of simulations. Specifically, particle pairs are constituted by the particles which start in the same exact position in twin simulations, and during a dzs run might end up in a differ-

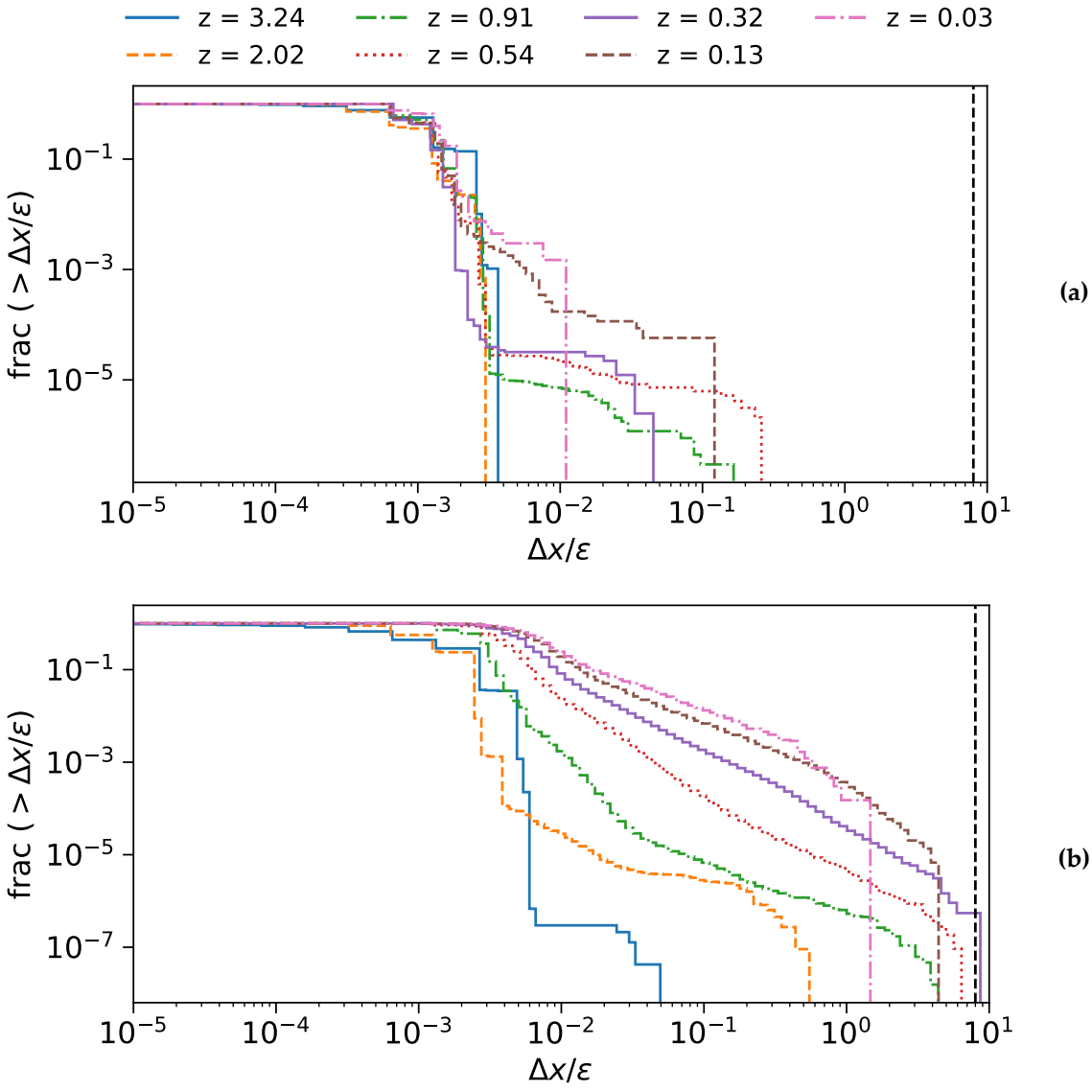


Figure 4.8: anticumulative distribution of particle displacements Δx calculated at various redshifts in the LARGE **(a)** and LARGEHR simulations **(b)**; the displacements are normalized to the softening length ε, equal to 0.4 *Mpc* in **(a)** and to 0.2 *Mpc* in **(b)**. The vertical black dotted line marks $\Delta x/\varepsilon = 8$.

	$z = 3.24$	$z = 2.02$	$z = 0.91$	$z = 0.54$	$z = 0.32$
LARGE	$1.08 \cdot 10^{-8}$	$6.22 \cdot 10^{-8}$	$1.0 \cdot 10^{-7}$	0.0	0.0
LARGEHR	0.0	$7.77 \cdot 10^{-9}$	$7.53 \cdot 10^{-8}$	$6.66 \cdot 10^{-7}$	$1.79 \cdot 10^{-6}$

	$z = 0.13$	$z = 0.03$
LARGE	0.0	0.0
LARGEHR	$2.23 \cdot 10^{-6}$	$1.50 \cdot 10^{-4}$

Table 4.2: number of particles without a match divided by the number of particles inside the lightcone in the std LARGE and LARGEHR simulations; the redshifts are equal to those depicted in **fig. 4.8**.

ent position with respect to the corresponding std run; such pairs are identified by matching particle IDs across snapshots performed in dzs and std runs. Quantifying the positional distance of each dzs particle from the corresponding std one (i.e. the displacement of that particle) gives a very precise idea of how the DZS algorithm only minimally affects the simulation domain. By construction, this particle displacement check can be performed only inside the lightcone (that is, the whole sphere with radius R_{lc}, not just a thin shell), where the dzs component of a particle pair cannot be removed or modified by the algorithm. For this reason, and to have the largest possible amount of particles to analyse as well as test the configuration with the largest expected DZS impact, displacements are calculated for the twin simulations LARGE and LARGEHR at different redshifts and shown in **fig. 4.8**. In fact, each displacement Δx is normalised by the softening length ε (see **section 2.1**), with the value $\Delta x/\varepsilon = 8$ being chosen as an upper limit for individual displacements which do not worsen the spatial reliability of a simulation (following Garaldi et al. 2020). In the LARGE simulations (**fig. 4.8a**), every displacement is way smaller than the upper limit, with the vast majority of particles having $\Delta x/\varepsilon \lesssim 10^{-2}$ in every examined redshift. There is, however, a minor fraction of particles ($\lesssim 10^{-3}$) with higher displacements which tend to get even higher with decreasing redshift, showing how the DZS-related approximations accumulate during a simulation. In fact, at $z = 3.24$ (i.e. shortly after the lightcone has entered the LARGE and LARGEHR simulation box) every particle has $\Delta x/\varepsilon < 10^{-2}$, but the tail of the distribution reaches higher displacements as z tends to zero, reaching its peak at $z = 0.54$. Afterwards, the maximum displacement becomes smaller as a result of the decreasing number of particles inside the lightcone.

The results of the LARGEHR runs (**fig. 4.8b**) follow a similar trend, but with much

higher displacements. Nonetheless, most particles are not displaced farther than 8 times the softening length, with the exception of a small fraction ($\approx 10^{-6}$) at $z = 0.32$; in fact, these few particles do not affect the overall visual differences between the std and the dzs run, as confirmed by **fig. 4.4**. Actually, the dependency on resolution observed in particle displacements, that is, a general increase of $\Delta x/\varepsilon$, seems to disagree with that of previous subsections. This discrepancy is indeed only apparent, because the HEALPix and 3D lightcone results focus on quantities integrated over the size of a pixel, where an increasing density of particles helps in making differences smaller; on the contrary, the present analysis takes into account the individual contributions of each particle, where a small fraction of values exceeding the set tolerance stands out among the vast majority of displacements. Furthermore, a lower simulation resolution implies an higher mass for tracer particles, leading to them having smaller peculiar velocities by construction, which by themselves limit the displacements.

Last but not least, it should be noted that the displacement of a particle can transport it inside or outside of the lightcone, possibly resulting in a number of particles without a matching companion. While this number is always a very small fraction of the whole content of the lightcone in the std runs, it is appropriate to include it in **tab. 4.2**, especially considering that particles without a companion cannot be depicted in **fig. 4.8**. The better resolution of the largeHR results in a very small fraction of particles inside the lightcone without a match in the dzs run, whereas the very few particles left inside R_{lc} at low redshift all have their companion in the large case.

4.2 Performance analysis

The previous section showed that runs performed with the DZS algorithm yield accurate output data when compared to simulations without it. While this is a reassuring result, it does not verify that Dynamic Zoom Simulations are indeed able to be carried out with a lower toll on computational resources than that of standard runs. In fact, this section aims to compare the run times and work-load balances of every pair of simulations depicted in **tab. 4.1**, to see, first of all, if there is an actual performance improvement and, second of all, how different boxes and resolutions affect that improvement. It should be noted that such an analysis ideally requires that both the std and dzs runs of a twin set of simulations are performed on the exact same computational units. In this framework, however, this requirement is not always met due to the large number of computational units, most of which are constantly in use, on the computing cluster the simulations have been performed on, that is, the SuperMUC-NG cluster of the Leibniz Supercomputing Centre. Nonetheless, all

the computational units employed have the same hardware and should thus be able to yield the same performance results. Another important thing to point out is that the following performance estimate comes from simulations which have performed output writing operations to create the data seen in the previous section; given that simulations including the DZS algorithm have a lower number of particles to save to disk, part of the observed performance gain can also be due to these output-related operations. In fact, this framework mimics a more realistic simulation scenario and highlights how the different operations of a code can benefit from a reduced number of particles.

4.2.1 Run time performance

The wall-clock time required to run simulations with and without dzs is depicted in **fig. 4.9**, both in a cumulative fashion and referred to the performance of individual simulation steps. In the top panel, independently of the specific twin simulations examined, the dzs and std runs show identical curves (with the exception of the MEDIUM runs in the first stages of the simulations, see below), but from the scale factor in which the first derefinement takes place (marked by the diamond symbols), the dzs runs gradually become more efficient, yielding a substantial flattening of the curves when the number of particles becomes half of the initial one (event marked by the circular symbols). In fact, this flattening results from the reduced run time per simulation step required by the dzs runs, highlighted in the bottom panel of **fig. 4.9**: as the volume outside of the lightcone becomes larger and more particles are eliminated, simulation steps take much less time to be carried out. Furthermore, the DZS-related performance gain also comes from the fact that the increased clustering of matter at low redshift leads to smaller particle timesteps and thus a more resource-hungry time evolution. Indeed, this performance analysis shows that having a small number of particles in this stage of a simulation helps in reducing the computational costs. The bottom panel of **fig. 4.9** also shows why the LARGE and LARGEHR runs have a better total run time saving (defined as $1-t^{dzs}_{wc}(a=1)/t^{std}_{wc}(a=1)$) than the MEDIUM and MEDIUMHR runs: the larger box allows the lightcone to enter the simulation domain earlier and node derefinements start to take place sooner, resulting in a more rapidly declining wall-clock time per simulation step and thus in a larger run time saving overall.

To discuss these run time savings in more detail, it is appropriate to point out a feature arising in **fig. 4.9**: in the top panel it is possible to notice a rapid increase in the cumulative run time at the end of each dashed curve (as shown by the smaller zoom

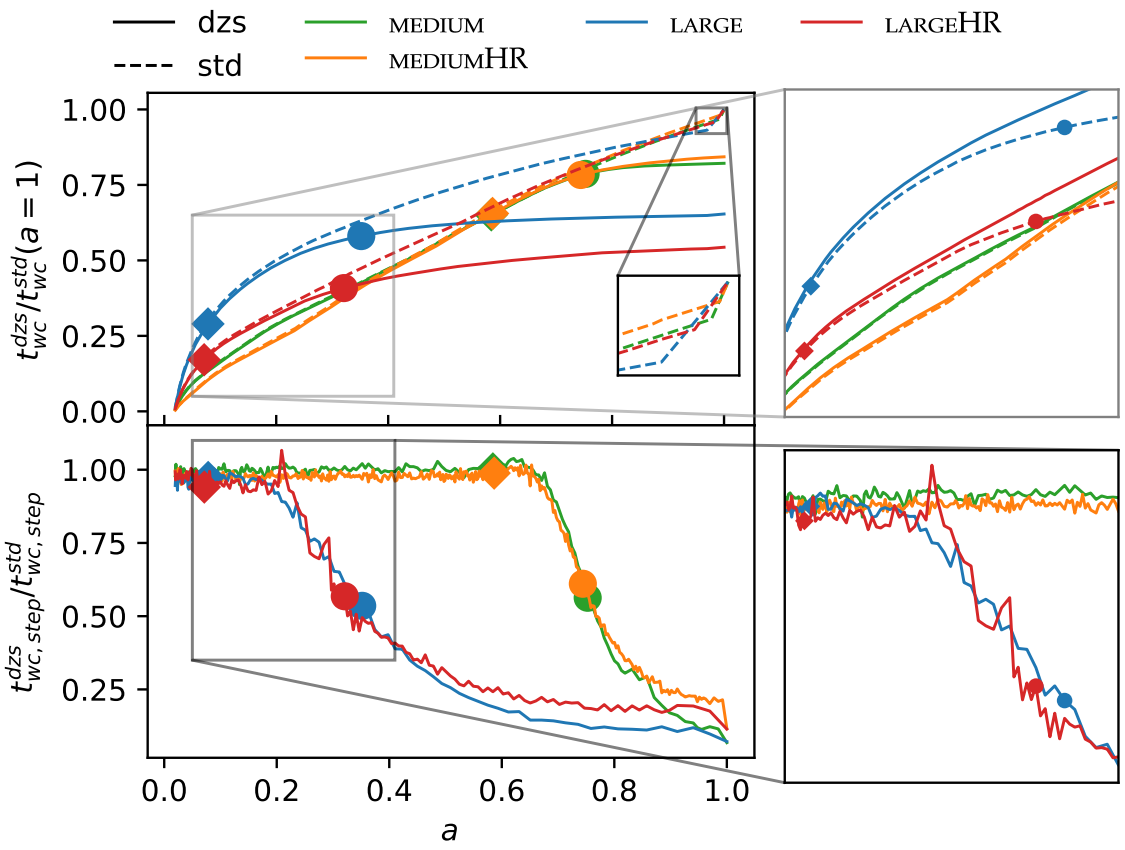


Figure 4.9: top panel: wall-clock time t_{wc}^{dzs} needed to run a dzs simulation up to a certain cosmic time (expressed through the scale factor a), as a function of a (solid curves); the curves are normalized to the total run time of the corresponding std runs (t_{wc}^{std} when $a = 1$) for comparison purposes, and the std runs are also plotted as a reference (dashed curves). The two zoom boxes highlight the early and last stages of the simulations. Bottom panel: ratio of the wall-clock time employed to perform a (global) simulation step in a dzs run ($t_{wc,step}^{dzs}$) and in the corresponding std one ($t_{wc,step}^{std}$), as a function of the scale factor each step was performed in. The zoom box highlights the early stages of the simulations. In both panels, diamonds identify the moment in which the first node is derefined, and circles mark the time at which the total number of particles becomes half of the initial one.

box), as if the final operations of std simulations required a substantial amount of computational resources. This happens because of the so-called restart files, which AREPO creates as a way to resume a simulation from the time in which it ended. In fact, these files contain the memory image of each MPI task, which makes their creation more significant in a std run with respect to a dzs run, due to the higher number of particles and data related to them. Indeed, this also works in favor of the DZS performance gain, and while creating the restart files might not seem like an operation directly related to the time evolution of tracer particles, is part of every realistic

	$t_{wc}^{dzs}(z=0)/t_{wc}^{std}(z=0)$	$N_{part}^{dzs}(z=0)/N_{part}^{dzs}(z=50)$
MEDIUM	0.82	0.046
MEDIUMHR	0.84	0.033
LARGE	0.65	0.045
LARGEHR	0.54	0.037

Table 4.3: total wall-clock time needed to complete a dzs simulation, normalized to the respective std total run time. For the sake of completeness, the second column of the table includes the final number of particles in each dzs run with respect to the initial number.

cosmological simulation setup that I chose to reproduce in this analysis.[2]

With that said, the run time savings yielded by the ICs with the same box size can now be compared: the MEDIUM and MEDIUMHR runs share a very similar result, showing that the two mass resolutions do not yield significant differences in the limited timeframe in which node derefinement takes place. On the other hand, this is not the case in the LARGE and LARGEHR runs: as can be seen in **tab. 4.3**, the increased resolution leads to a lower fraction of the total std run time being employed by the dzs simulation, and thus to a higher run time saving. This result likely comes from the aforementioned increased computational cost of low redshift simulation steps, whose performance impact is usually directly related to the resolution. Therefore, the performance gain provided by the DZS algorithm should also be more significant when the number of particles in a box is increased, which is indeed what happens in the LARGE and LARGEHR simulations. In fact, in the latter pair of simulations the dzs run has been completed in nearly half the run time taken by the std run, which is a very promising result for high-resolution simulations.

4.2.2 Work-load balance

Given the fact that the DZS algorithm heavily modifies the simulation domain and basically separates it into a high resolution and a low resolution volume, it is appropriate to see how this affects the work-load balance[3] during a run, with an emphasis

[2]Usually, cosmological simulations produce restart files at regular intervals (every few hours) as a safety measure against crashes and interruptions, to resume a run without having to repeat it from the start. The simulations analysed here, on the other hand, only produce restart files at the end of the run, because completing these simulations does not require more than a few hours in the first place. Nonetheless, it should be noted that the regular production of restart files during a simulation in a realistic setup makes the DZS algorithm even more efficient.

[3]As mentioned in **chapter 3**, the work-load balance is a property of every simulation step and is defined as the maximum wall-clock time spent on tree algorithm across every MPI task, divided by

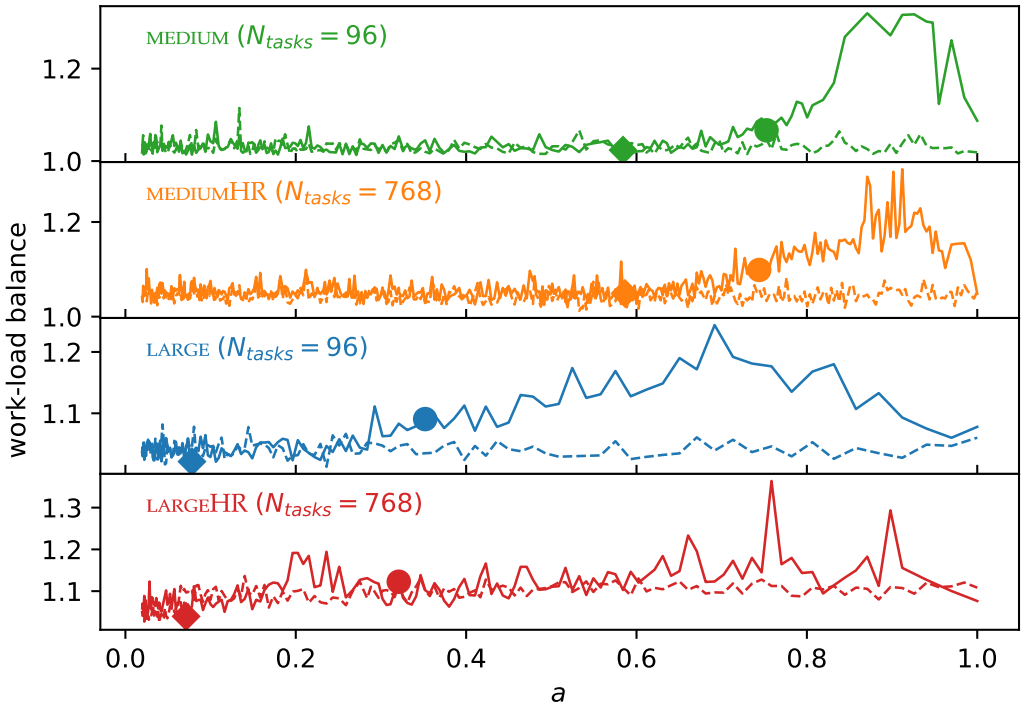


Figure 4.10: work-load balance of the dzs runs (solid lines) and of the corresponding std runs (dashed) lines. The quantity N_{tasks} represents the number of MPI tasks each pair of simulations was performed with. The role of diamond and circular symbols is identical to that which they have in **fig. 4.9**.

on large deviations from the ideal value of 1. The work-load balances of all simulations included in **tab. 4.1** are plotted against the scale factor in **fig. 4.10**; the plot also includes the number of MPI tasks employed in each pair of runs, because having a number of tasks which is too high compared to the size of the problem can largely affect the work-load balance. There are in fact DZS-related imbalances in every examined simulation, with peak values getting as high as 1.3. In every set of twin simulations, reducing the number of particles makes it harder for the domain decomposition to distribute them evenly across tasks. Eventually, however, all the dzs curves start decreasing towards $a = 1$ either gradually or abruptly, signaling that AREPO is finally managing to keep the imbalances in check. To understand why this is the case, it is possible to notice that the simulation stages with the highest imbalances are mostly dependent on the box size, being located around $a \simeq 0.7$ or 0.8 for the LARGE and LARGEHR runs and at $a \simeq 0.9$ for the MEDIUM and MEDIUMHR simulations. At these points of a dzs simulation, the clustering of matter, which is a source of imbalance in itself (because the clustering leads to tasks having fewer particles resort to communications more often and slowing down the workflow), occurs mostly

its average value.

inside of the lightcone, and the tasks working in this portion of the simulation volume will generally take a longer time to complete their operations with respect to the tasks working outside R_{lc}. As the lightcone shrinks towards the observer, most of the simulation domain adjusts to the same resolution and the imbalance becomes lower. In any case, the imbalance peaks show significant deviations from the std runs (whose work-load balance remains always reasonably close to 1), and while the performance gains depicted in **subsec. 4.2.1** are very promising, the results of **fig. 4.10** show that there is indeed some room for improvement (also because the work-load balances of dzs runs in `PGADGET-3` deviate from 1 at most by a few per cent, Garaldi et al. 2020). Some brief suggestions to achieve this improvements and to deal with imbalances in general can be found in the next and final chapter of this work, which also sums up the state of the current implementation and mentions some possible extensions.

5 | Summary, conclusions and future prospects

In this work, I have introduced the DZS framework and described its implementation in the code AREPO, highlighting the performance gain that can be obtained with this technique in a DM-only scenario. More specifically:

- in **chapter 1**, I have presented the cosmological framework and the theory of gravitational instabilities that leads to structure formation, highlighting the crucial role of dark matter. In addition, I have also described the main mechanisms that lead to the formation and evolution of stellar objects and galaxies. Last but not least, I have introduced cosmological simulations as a way to address the lack of a purely analytical framework for structure and galaxy formation, emphasizing the role of tracer particles and limitations on their number in high-resolution simulations;

- in **chapter 2**, I have described some techniques employed to make gravity and hydrodynamics calculations more efficient. Specifically, I have included the particle-mesh method and the hierarchical multipole method for the gravitational framework (as well as the combination of the two methods, which constitues the tree-PM approach), and the smoothed particle hydrodynamics and moving mesh methods to treat baryonic physics;

- in **chapter 3**, I have provided a general overview of the Dynamic Zoom Simulations algorithm as a way to overcome the aforementioned limitations in high-resolution simulations, and a detailed description of its implementation in the tree-PM moving mesh code AREPO, starting with the DM-only, tree-based approach. This consists in merging the particle content of tree nodes lying outside the past lightcone of a random observer into single, massive particles. The process includes a tree walk, in which different operations are performed on nodes, particles, pseudoparticles or imported points, the particle elimination and creation stage, and the way to insert the algorithm in the general code

workflow. I have also introduced a possible mesh-based approach for baryon derefinement, relatively easy to implement but still to be tested in terms of accuracy and performance gain;

- in **chapter 4**, I have tested the ability of the DM-only DZS algorithm to correctly produce lightcone-like output in a variety of simulation setups. Specifically, this output included the Lightcone Halo Mass Function, the sky-projected lightcone and the 3D lightcone, with DZS-related approximations being very contained (below 1% in most cases). Afterwards, I verified that the algorithm does indeed introduce a performance gain, and studied how the simulation resolution and the box size affect this gain in the employed setups. Specifically, the simulations that I have performed with the DZS algorithm need from 16% to 46% less time to be completed than the corresponding standard runs. These results are expected to get even better when employing higher resolutions, larger boxes, and/or including the more complex baryonic physics.

The current state of Dynamic Zoom Simulations in `AREPO`, while being self-consistent, allows a variety of extensions and improvements which is appropriate to mention here, starting with the suggestions regarding work-load balance mentioned at the end of **chapter 4**.

5.1 Work-load balance and DZS special stop

As seen in **subsec. 4.2.2**, DZS-related modifications on the simulation domain indeed take their toll on the work-load balance of global simulation steps. While this is expected due to the change in resolution and the clustering of matter, some adjustments could be made in order to reduce the imbalance and further improve performance when possible. For example, `AREPO` weighs the cost of the tree-related gravitational calculations of each particle through the real numbers `GravCost`; when a new particle is created from a tree node, however, the associated `GravCost` is set to zero, as well as that of every particle flagged for removal. This can make a proper work-load balance more difficult to achieve in the next domain decomposition and partly contribute to the peaks seen in **fig. 4.10**. I already started working in this direction by performing simulations where the `GravCost` of new particles was given by the maximum value across all particles inside the corresponding derefined node, but without any noticeable work-load balance improvement. It is possible that more sophisticated choices for the `GravCost` of new particles could help in reducing the observed imbalances.

In any case, it is foreseeable that the DZS-related change in resolution will eventually lead to a large imbalance (especially with a high number of particles at the start of a simulation), due to the lower and lower number of particles assigned to MPI tasks and the subsequent large use of communications. For this reason, the PGADGET-3 implementation of DZS allows to set a user-defined threshold for the maximum allowed imbalance or the minimum ratio of the current and the starting number of particles. If the simulation exceeds one of these limits, it stops as soon as possible in order to be resumed with more appropriate computational resources, namely fewer MPI tasks and memory. It is important to point out, however, that it is not possible to use the restart files mentioned in **subsec. 4.2.1** in this DZS "special stop" framework, because a run restarted with those files needs to have the same number of tasks as the files were created with. This restriction is also present in PGADGET-3, which is why the existing implementation of DZS special stop implies the generation of a snapshot when the run is stopped, as simulations can be restarded from snapshots without the limitations that apply to restart files. Snapshot files, however, tipically do not store data as precisely as restart files, which is why resuming a simulation with DZS special stop could lead to additional approximations with respect to a standard run, and testing in this sense will be required when the feature is added to AREPO.

5.2 Further validation and additional physics

Other than the tests required by the future implementation of DZS special stop, further analysis is needed to complement the work described in **chapter 4**. First of all, the DZS algorithm depends on a variety of parameters: the opening angle θ_{geom}, for example, or the quantity α of **eq. 3.2**, when the dynamical criterion is employed; the buffer and the maximum node size also play a key role in determining DZS-related approximations and performance gains. Therefore, a comprehensive parameter dependence analysis should be carried out for a quantitative understanding of how different parameters and/or criterion combinations affect both lightcone-like output and individual particle displacements, as well as the overall time saving of the algorithm. Such a testing procedure will also help to validate the DZS algorithm in the presence of a dynamical derefinement criterion (as all the runs discussed in **chapter 4** have been performed with a geometrical criterion).

Second of all, while the suite of simulations in this work offers a variety of different initial conditions, it would be appropriate to extend the validation to the large-scale and high-resolution simulations that the DZS algorithm has been designed for. This is especially true in the baryon derefinement framework (once ready),

whose validation simulations should also gradually include the additional physics described in **chapter 1**, such as cooling, star formation, magnetic fields and stellar and AGN feedback. In fact, the capabilities of the DZS algorithm can be extended even further, as it is virtually suited to every particle-based model of alternative physics; AREPO, for example, can already employ models such as modified gravity (Arnold et al. 2019), self-interacting DM (Chua et al. 2021), fuzzy DM (Mocz et al. 2019) and massive neutrinos (Adamek et al. 2022); all these modifications of AREPO, as well as other models not yet implemented, could be adapted to the DZS framework and benefit from its performance gain.

5.3 Conclusions

In this work I presented the current framework of cosmological simulations and proposed a method to make these simulations more efficient. My implementation of this method in a state of the art code for baryon physics simulations such as AREPO has yielded a safe, accurate and efficient algorithm capable of reducing the run time needed by a DM-only simulation almost by 50% in one of the tested pairs, with relative differences with respect to a standard run mostly below 1%. In fact, these results are compatible with the ones reported in Garaldi et al. (2020), and it can be foreseen that the two implementations will share a similar increase in performance gain when the algorithm is applied to the next generation of cosmological simulations, where the very large volumes and high mass resolutions will all play in favor of the DZS framework. As mentioned in the previous section, the algorithm is also very versatile, because its structure makes it adaptable to additional complex and non-standard physics in a relatively straightforward way.

In conclusion, with this work I hope to have paved the way to a wider and wider usage of the DZS algorithm with a variety of physical models and in increasingly complicated scenarios, which will all take advantage of the performance improvement needed to reach unprecedented levels of resolution and accuracy.

Bibliography

Adamek, J. et al. (Nov. 2022). "Euclid: Modelling massive neutrinos in cosmology – a code comparison". In: *arXiv e-prints*, arXiv:2211.12457, arXiv:2211.12457. DOI: 10.48550/arXiv.2211.12457. arXiv: 2211.12457 [astro-ph.CO].

Alimi, Jean-Michel et al. (June 2012). "DEUS Full Observable ΛCDM Universe Simulation: the numerical challenge". In: *arXiv e-prints*, arXiv:1206.2838, arXiv:1206.2838. arXiv: 1206.2838 [astro-ph.CO].

Angulo, R. E. et al. (Nov. 2012). "Scaling relations for galaxy clusters in the Millennium-XXL simulation". In: *Monthly Notices of the Royal Astronomical Society* 426.3, pp. 2046–2062. DOI: 10.1111/j.1365-2966.2012.21830.x. arXiv: 1203.3216 [astro-ph.CO].

Arnold, Christian, Matteo Leo, and Baojiu Li (July 2019). "Realistic simulations of galaxy formation in f(R) modified gravity". In: *Nature Astronomy* 3, pp. 945–954. DOI: 10.1038/s41550-019-0823-y. arXiv: 1907.02977 [astro-ph.CO].

Barnes, Josh and Piet Hut (Dec. 1986). "A hierarchical O(N log N) force-calculation algorithm". In: *Nature* 324.6096, pp. 446–449. DOI: 10.1038/324446a0.

Bennett, Charles L. et al. (Jan. 2003). "The Microwave Anisotropy Probe Mission". In: *The Astrophysical Journal* 583.1, pp. 1–23. DOI: 10.1086/345346. arXiv: astro-ph/0301158 [astro-ph].

Bertone, Gianfranco, Dan Hooper, and Joseph Silk (Jan. 2005). "Particle dark matter: evidence, candidates and constraints". In: *Physics Reports* 405.5-6, pp. 279–390. DOI: 10.1016/j.physrep.2004.08.031. arXiv: hep-ph/0404175 [hep-ph].

Binney, James and Scott Tremaine (2008). *Galactic dynamics*. 2nd ed. Princeton series in astrophysics. Princeton University Press.

Birnboim, Yuval and Avishai Dekel (Oct. 2003). "Virial shocks in galactic haloes?" In: *Monthly Notices of the Royal Astronomical Society* 345.1, pp. 349–364. DOI: 10.1046/j.1365-8711.2003.06955.x. arXiv: astro-ph/0302161 [astro-ph].

Blumenthal, George R. et al. (Oct. 1984). "Formation of galaxies and large-scale structure with cold dark matter". In: *Nature* 311.5986, pp. 517–525. DOI: 10.1038/311517a0.

Bode, Paul, Jeremiah P. Ostriker, and Neil Turok (July 2001). "Halo Formation in Warm Dark Matter Models". In: *The Astrophysical Journal* 556.1, pp. 93–107. DOI: 10.1086/321541. arXiv: astro-ph/0010389 [astro-ph].

Bond, J. Richard, Lev Kofman, and Dmitry Pogosyan (Apr. 1996). "How filaments of galaxies are woven into the cosmic web". In: *Nature* 380.6575, pp. 603–606. DOI: 10.1038/380603a0. arXiv: astro-ph/9512141 [astro-ph].

Braun, Robert et al. (2015). "Advancing Astrophysics with the Square Kilometre Array". In: *Proceedings of Advancing Astrophysics with the Square Kilometre Array — PoS(AASKA14)*. Vol. 215, p. 174. DOI: 10.22323/1.215.0174.

Chua, Kun Ting Eddie et al. (Jan. 2021). "The impact of inelastic self-interacting dark matter on the dark matter structure of a Milky Way halo". In: *Monthly Notices of the Royal Astronomical Society* 500.1, pp. 1531–1546. DOI: 10.1093/mnras/staa3315. arXiv: 2010.08562 [astro-ph.GA].

Cimatti, Andrea, Filippo Fraternali, and Carlo Nipoti (2020). *Introduction to galaxy formation and evolution: from primordial gas to present-day galaxies*. Cambridge University Press.

Ciotti, Luca (2021). *Introduction to stellar dynamics*. Cambridge University Press.

Clarke, Lyndon, Ian Glendinning, and Rolf Hempel (1994). "The MPI Message Passing Interface Standard". In: *Programming Environments for Massively Parallel Distributed Systems*. Ed. by Karsten M. Decker and René M. Rehmann. Birkhäuser Basel, pp. 213–218. DOI: 10.1007/978-3-0348-8534-8_21.

Coles, Peter and Francesco Lucchin (2002). *Cosmology: the origin and evolution of cosmic structure*. 2nd ed. John Wiley.

Condon, J J. and A. M. Matthews (July 2018). "ΛCDM Cosmology for Astronomers". In: *Publications of the Astronomical Society of the Pacific* 130.989, p. 073001. DOI: 10.1088/1538-3873/aac1b2. arXiv: 1804.10047 [astro-ph.CO].

Cooley, James W. and John W. Tukey (1965). "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of Computation* 19.90, pp. 297–301. DOI: 10.1090/S0025-5718-1965-0178586-1.

Cox, Donald P. and Wallace H. Tucker (Sept. 1969). "Ionization Equilibrium and Radiative Cooling of a Low-Density Plasma". In: *The Astrophysical Journal* 157, p. 1157. DOI: 10.1086/150144.

Croton, Darren J. (Oct. 2013). "Damn You, Little h! (Or, Real-World Applications of the Hubble Constant Using Observed and Simulated Data)". In: *Publications of the Astronomical Society of Australia* 30, e052, e052. DOI: 10.1017/pasa.2013.31. arXiv: 1308.4150 [astro-ph.CO].

D'Inverno, Ray (1992). *Introducing Einstein's Relativity*. Clarendon Press.

Davis, M. et al. (May 1985). "The evolution of large-scale structure in a universe dominated by cold dark matter". In: *The Astrophysical Journal* 292, pp. 371–394. DOI: 10.1086/163168.

Diemand, J. et al. (Aug. 2008). "Clumps and streams in the local dark matter distribution". In: *Nature* 454.7205, pp. 735–738. DOI: 10.1038/nature07153. arXiv: 0805.1244 [astro-ph].

Dubois, Y. et al. (Oct. 2014). "Dancing in the dark: galactic properties trace spin swings along the cosmic web". In: *Monthly Notices of the Royal Astronomical Society* 444.2, pp. 1453–1468. DOI: 10.1093/mnras/stu1227. arXiv: 1402.1165 [astro-ph.CO].

Einstein, Albert (Nov. 1915). "Zur allgemeinen Relativitätstheorie". In: *Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften*, pp. 778–786. "On the General Theory of Relativity". In: *The collected papers of Albert Einstein: writings, 1914-1917*. Ed. by Alfred Engel. Princeton University Press, 1997.

Einstein, Albert (Mar. 1916). "Die Grundlage der allgemeinen Relativitätstheorie". In: *Annalen der Physik* 354.7, pp. 769–822. DOI: 10.1002/andp.19163540702. Trans. as "The Foundation of the General Theory of Relativity". In: *The collected papers of Albert Einstein: writings, 1914-1917*. Ed. by Alfred Engel. Princeton University Press, 1997.

Einstein, Albert (Feb. 1917). "Kosmologische Betrachtungen zur allgemeinen Relativitätstheorie". In: *Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften*, pp. 142–152. "Cosmological Considerations in the General Theory of Relativity". In: *The collected papers of Albert Einstein: writings, 1914-1917*. Ed. by Alfred Engel. Princeton University Press, 1997.

Fabian, A. C. (Jan. 1994). "Cooling Flows in Clusters of Galaxies". In: *Annual Review of Astronomy and Astrophysics* 32, pp. 277–318. DOI: 10.1146/annurev.aa.32.090194.001425.

Fosalba, Pablo et al. (Nov. 2008). "The onion universe: all sky lightcone simulations in spherical shells". In: *Monthly Notices of the Royal Astronomical Society* 391.1, pp. 435–446. DOI: 10.1111/j.1365-2966.2008.13910.x. arXiv: 0711.1540 [astro-ph].

Friedmann, Alexander (Jan. 1922). "Über die Krümmung des Raumes". In: *Zeitschrift fur Physik* 10, pp. 377–386. DOI: 10.1007/BF01332580. Trans. as "On the Curvature of Space". In: *General Relativity and Gravitation* 31 (Dec. 1999), p. 1991. DOI: 10.1023/A:1026751225741.

Galli, Daniele and Francesco Palla (Aug. 2013). "The Dawn of Chemistry". In: *Annual Review of Astronomy and Astrophysics* 51.1, pp. 163–206. DOI: 10.1146/annurev-astro-082812-141029. arXiv: 1211.3319 [astro-ph.CO].

Gao, L. et al. (Sept. 2012). "The Phoenix Project: the dark side of rich Galaxy clusters". In: *Monthly Notices of the Royal Astronomical Society* 425.3, pp. 2169–2186. DOI: 10.1111/j.1365-2966.2012.21564.x. arXiv: 1201.1940 [astro-ph.CO].

Garaldi, Enrico, Matteo Nori, and Marco Baldi (Dec. 2020). "Dynamic zoom simulations: A fast, adaptive algorithm for simulating light-cones". In: *Monthly Notices of the Royal Astronomical Society* 499.2, pp. 2685–2700. DOI: 10.1093/mnras/staa2064. arXiv: 2005.05328 [astro-ph.CO].

Glowinski, Roland, Stanley J. Osher, and Wotao Yin (eds.) (2016). *Splitting Methods in Communication, Imaging, Science, and Engineering*. Scientific Computation. Springer.

Goldstein, Herbert, Charles P. Poole, and John L. Safko (2008). *Classical mechanics*. 3rd ed. Addison Wesley.

Górski, K. M. et al. (Apr. 2005). "HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere". In: *The Astrophysical Journal* 622.2, pp. 759–771. DOI: 10.1086/427976. arXiv: astro-ph/0409513 [astro-ph].

Gottlieb, J. J. and C. P. T. Groth (Oct. 1988). "Assessment of Riemann Solvers for Unsteady One-Dimensional Inviscid Flows for Perfect Gases". In: *Journal of Computational Physics* 78.2, pp. 437–458. DOI: 10.1016/0021-9991(88)90059-9.

Grand, Robert J. J. et al. (May 2017). "The Auriga Project: the properties and formation mechanisms of disc galaxies across cosmic time". In: *Monthly Notices of the Royal Astronomical Society* 467.1, pp. 179–207. DOI: 10.1093/mnras/stx071. arXiv: 1610.01159 [astro-ph.GA].

Haardt, F. and L. Maraschi (Oct. 1991). "A Two-Phase Model for the X-Ray Emission from Seyfert Galaxies". In: *Astrophysical Journal Letters* 380, p. L51. DOI: 10.1086/186171.

Harten, Amiram, Peter D. Lax, and Bram van Leer (1983). "On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws". In: *SIAM Review* 25.1, pp. 35–61. DOI: 10.1137/1025002. eprint: https://doi.org/10.1137/1025002. URL: https://doi.org/10.1137/1025002.

Hubble, Edwin (Mar. 1929). "A Relation between Distance and Radial Velocity among Extra-Galactic Nebulae". In: *Proceedings of the National Academy of Science* 15.3, pp. 168–173. DOI: 10.1073/pnas.15.3.168.

Iršič, Vid et al. (July 2017). "New constraints on the free-streaming of warm dark matter from intermediate and small scale Lyman-α forest data". In: *Physical Review D* 96.2, 023522, p. 023522. DOI: 10.1103/PhysRevD.96.023522. arXiv: 1702.01764 [astro-ph.CO].

Ivezić, Željko et al. (Mar. 2019). "LSST: From Science Drivers to Reference Design and Anticipated Data Products". In: *The Astrophysical Journal* 873.2, 111, p. 111. DOI: 10.3847/1538-4357/ab042c. arXiv: 0805.2366 [astro-ph].

Jeans, James H. (Jan. 1902). "The Stability of a Spherical Nebula". In: *Philosophical Transactions of the Royal Society of London Series A* 199, pp. 1–53. DOI: 10.1098/rsta.1902.0012.

Kereš, Dušan et al. (Oct. 2005). "How do galaxies get their gas?" In: *Monthly Notices of the Royal Astronomical Society* 363.1, pp. 2–28. DOI: 10.1111/j.1365-2966.2005.09451.x. arXiv: astro-ph/0407095 [astro-ph].

Khandai, Nishikanta et al. (June 2015). "The MassiveBlack-II simulation: the evolution of haloes and galaxies to z ∼ 0". In: *Monthly Notices of the Royal Astronomical Society* 450.2, pp. 1349–1374. DOI: 10.1093/mnras/stv627. arXiv: 1402.0888 [astro-ph.CO].

Klypin, Anatoly A., Sebastian Trujillo-Gomez, and Joel Primack (Oct. 2011). "Dark Matter Halos in the Standard Cosmological Model: Results from the Bolshoi Simulation". In: *The Astrophysical Journal* 740.2, 102, p. 102. DOI: 10.1088/0004-637X/740/2/102. arXiv: 1002.3660 [astro-ph.CO].

Landau, Lev D. and Evgenij M. Lifshitz (1959). *Fluid mechanics*. Course of theoretical physics. Pergamon Press.

Laureijs, Rene et al. (Oct. 2011). "Euclid Definition Study Report". In: *arXiv e-prints*. arXiv: 1110.3193 [astro-ph.CO].

Linde, Andrei D. (Mar. 1974). "Is the Lee constant a cosmological constant?" In: *Soviet Journal of Experimental and Theoretical Physics Letters* 19, p. 183.

Linde, Andrei D. (Aug. 1984). "The inflationary Universe". In: *Reports on Progress in Physics* 47, pp. 925–986. DOI: 10.1088/0034-4885/47/8/002.

Linde, Andrei D. (1990). *Particle physics and inflationary cosmology*.

Llinares, Claudio (Sept. 2017). "The shrinking domain framework I: a new, faster, more efficient approach to cosmological simulations". In: *arXiv e-prints*. arXiv: 1709.04703 [astro-ph.CO].

Marinoni, Christian, J. Bel, and A. Buzzi (Oct. 2012). "The scale of cosmic isotropy". In: *Journal of Cosmology and Astroparticle Physics* 2012.10, p. 036. DOI: 10.1088/1475-7516/2012/10/036.

Matteucci, F. and L. Greggio (Jan. 1986). "Relative roles of type I and II supernovae in the chemical enrichment of the interstellar gas". In: *Astronomy & Astrophysics* 154.1-2, pp. 279–287.

Mocz, Philip et al. (Oct. 2019). "First Star-Forming Structures in Fuzzy Cosmic Filaments". In: *Physical Review Letters* 123.14, 141301, p. 141301. DOI: 10.1103/PhysRevLett.123.141301. arXiv: 1910.01653 [astro-ph.GA].

Narayan, Ramesh and Insu Yi (May 1995). "Advection-dominated Accretion: Self-Similarity and Bipolar Outflows". In: *The Astrophysical Journal* 444, p. 231. DOI: 10.1086/175599. arXiv: astro-ph/9411058 [astro-ph].

Padmanabhan, T. (1993). *Structure Formation in the Universe*. Cambridge University Press.

Penzias, Arno A. and Robert W. Wilson (July 1965). "A Measurement of Excess Antenna Temperature at 4080 Mc/s." In: *The Astrophysical Journal* 142, pp. 419–421. DOI: 10.1086/148307.

Pier, Edward A. and Julian H. Krolik (Dec. 1992). "Infrared Spectra of Obscuring Dust Tori around Active Galactic Nuclei. I. Calculational Method and Basic Trends". In: *The Astrophysical Journal* 401, p. 99. DOI: 10.1086/172042.

Planck Collaboration (Sept. 2016). "Planck 2015 results. XIII. Cosmological parameters". In: *Astronomy & Astrophysics* 594, A13, A13. DOI: 10.1051/0004-6361/201525830. arXiv: 1502.01589 [astro-ph.CO].

Planck Collaboration (Sept. 2020). "Planck 2018 results. VI. Cosmological parameters". In: *Astronomy & Astrophysics* 641, A6, A6. DOI: 10.1051/0004-6361/201833910. arXiv: 1807.06209 [astro-ph.CO].

Rankine, William J. M. (Dec. 1870). "XV. On the thermodynamic theory of waves of finite longitudinal disturbance". In: *Philosophical Transactions of the Royal Society of London* 160, pp. 277–288. DOI: 10.1098/rstl.1870.0015.

Rees, Martin J. and Jeremiah P. Ostriker (June 1977). "Cooling, dynamics and fragmentation of massive gas clouds: clues to the masses and radii of galaxies and clusters." In: *Monthly Notices of the Royal Astronomical Society* 179, pp. 541–559. DOI: 10.1093/mnras/179.4.541.

Riess, Adam G. et al. (Sept. 1998). "Observational Evidence from Supernovae for an Accelerating Universe and a Cosmological Constant". In: *Astronomical Journal* 116.3, pp. 1009–1038. DOI: 10.1086/300499. arXiv: astro-ph/9805201 [astro-ph].

Robertson, B. E. et al. (Dec. 2022). "Discovery and properties of the earliest galaxies with confirmed distances". In: *arXiv e-prints*, arXiv:2212.04480, arXiv:2212.04480. DOI: 10.48550/arXiv.2212.04480. arXiv: 2212.04480 [astro-ph.GA].

Ryden, Barbara S. (2017). *Introduction to cosmology*. 2nd ed. Cambridge University Press.

Schaye, Joop et al. (Jan. 2015). "The EAGLE project: simulating the evolution and assembly of galaxies and their environments". In: *Monthly Notices of the Royal Astro-*

nomical Society 446.1, pp. 521–554. DOI: 10.1093/mnras/stu2058. arXiv: 1407.7040 [astro-ph.GA].

Shakura, Nikolai I. and Rašid A. Sunyaev (Jan. 1973). "Black holes in binary systems. Observational appearance." In: *Astronomy & Astrophysics* 24, pp. 337–355.

Shu, Frank H. (1992). *The physics of astrophysics. 2: Gas dynamics*. A series of books in astronomy. Univ. Science Books.

Skillman, Samuel W. et al. (July 2014). "Dark Sky Simulations: Early Data Release". In: *arXiv e-prints*, arXiv:1407.2600, arXiv:1407.2600. arXiv: 1407.2600 [astro-ph.CO].

Smoot, George F. et al. (Sept. 1992). "Structure in the COBE Differential Microwave Radiometer First-Year Maps". In: *Astrophysical Journal Letters* 396, p. L1. DOI: 10.1086/186504.

Spergel, David et al. (Mar. 2015). "Wide-Field InfrarRed Survey Telescope-Astrophysics Focused Telescope Assets WFIRST-AFTA 2015 Report". In: *arXiv e-prints*. arXiv: 1503.03757 [astro-ph.IM]. Preprint.

Springel, Volker (Dec. 2005). "The cosmological simulation code GADGET-2". In: *Monthly Notices of the Royal Astronomical Society* 364.4, pp. 1105–1134. DOI: 10.1111/j.1365-2966.2005.09655.x. arXiv: astro-ph/0505010 [astro-ph].

Springel, Volker (Jan. 2010). "E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh". In: *Monthly Notices of the Royal Astronomical Society* 401.2, pp. 791–851. DOI: 10.1111/j.1365-2966.2009.15715.x. arXiv: 0901.4107 [astro-ph.CO].

Springel, Volker (Jan. 2016). "High Performance Computing and Numerical Modelling". In: *Saas-Fee Advanced Course* 43, p. 251. DOI: 10.1007/978-3-662-47890-5_3. arXiv: 1412.5187 [astro-ph.GA].

Springel, Volker et al. (Mar. 2018). "First results from the IllustrisTNG simulations: matter and galaxy clustering". In: *Monthly Notices of the Royal Astronomical Society* 475.1, pp. 676–698. DOI: 10.1093/mnras/stx3304. arXiv: 1707.03397 [astro-ph.GA].

Starobinskii, Alexei A. (June 1983). "The Perturbation Spectrum Evolving from a Nonsingular Initially De-Sitter Cosmology and the Microwave Background Anisotropy". In: *Soviet Astronomy Letters* 9, pp. 302–304.

Tucker, Wallace H. and Laurence P. David (July 1997). "A Feedback Model for Radio Sources Fueled by Cooling Flows". In: *The Astrophysical Journal* 484.2, pp. 602–607. DOI: 10.1086/304345.

van Albada, Tjeerd S. et al. (Aug. 1985). "Distribution of dark matter in the spiral galaxy NGC 3198." In: *The Astrophysical Journal* 295, pp. 305–313. DOI: 10.1086/163375.

Vogelsberger, Mark et al. (Jan. 2020). "Cosmological simulations of galaxy formation". In: *Nature Reviews Physics* 2.1, pp. 42–66. DOI: 10.1038/s42254-019-0127-2. arXiv: 1909.07976 [astro-ph.GA].

Von Neumann, John and Robert D. Richtmyer (Mar. 1950). "A Method for the Numerical Calculation of Hydrodynamic Shocks". In: *Journal of Applied Physics* 21.3, pp. 232–237. DOI: 10.1063/1.1699639.

Walker, Arthur G. (Jan. 1937). "On Milne's Theory of World-Structure". In: *Proceedings of the London Mathematical Society* 42, pp. 90–127. DOI: 10.1112/plms/s2-42.1.90.

Weinberger, Rainer, Volker Springel, and Rüdiger Pakmor (June 2020). "The AREPO Public Code Release". In: *The Astrophysical Journal Supplement Series* 248.2, 32, p. 32. DOI: 10.3847/1538-4365/ab908c. arXiv: 1909.04667 [astro-ph.IM].

Wetzel, Andrew R. et al. (Aug. 2016). "Reconciling Dwarf Galaxies with ΛCDM Cosmology: Simulating a Realistic Population of Satellites around a Milky Way-mass Galaxy". In: *Astrophysical Journal Letters* 827.2, L23, p. L23. DOI: 10.3847/2041-8205/827/2/L23. arXiv: 1602.05957 [astro-ph.GA].

Zhang, Tianchi et al. (July 2019). "The optimal gravitational softening length for cosmological N-body simulations". In: *Monthly Notices of the Royal Astronomical Society* 487.1, pp. 1227–1232. DOI: 10.1093/mnras/stz1370. arXiv: 1810.07055 [astro-ph.CO].

Zheng, Wei et al. (Sept. 2012). "A magnified young galaxy from about 500 million years after the Big Bang". In: *Nature* 489.7416, pp. 406–408. DOI: 10.1038/nature11446.

Zubovas, Kastytis and Andrew King (Feb. 2012). "Clearing Out a Galaxy". In: *Astrophysical Journal Letters* 745.2, L34, p. L34. DOI: 10.1088/2041-8205/745/2/L34. arXiv: 1201.0866 [astro-ph.GA].

Zwicky, Fritz (Jan. 1933). "Die Rotverschiebung von extragalaktischen Nebeln". In: *Helvetica Physica Acta* 6, pp. 110–127. Trans. as "Republication of: The redshift of extragalactic nebulae". In: *General Relativity and Gravitation* 41.1 (Jan. 2009), pp. 207–224. DOI: 10.1007/s10714-008-0707-4.