Alma Mater Studiorum - Università di Bologna

School of Science
Department of Physics and Astronomy
Master Degree Programme in Astrophysics and Cosmology

Cosmological Simulations on GPUs in the Era of High-Precision Cosmology

Graduation Thesis

March 27, 2026

Presented by:
Marianna Aiello

Supervisor:
Prof. Lauro Moscardini

Co-supervisor:
Prof. Klaus Dolag
Dott. Geray Karademir

Academic year 2024-2025
Graduation date V

È così bello fissare il cielo e accorgersi di come non sia altro che un vero e proprio immenso laboratorio di fisica che si srotola sulle nostre teste.

Margherita Hack

Abstract

In recent years, the role of Graphics Processing Units (GPUs) has become increasingly relevant in the world of high-performance scientific computing. In the context of large cosmological simulations, where several million hours of computational time are needed, the use of GPUs enables running bigger simulations at higher resolution.

In this master's thesis, we study the current implementation of the gravity solver in the cosmological simulation code OpenGadget3 (Ragagnin et al., 2020), with the goal of assessing its numerical accuracy across different hardware architectures and simulation setups. Using dark-matter-only simulations, using initial conditions from the Magneticum simulations (Dolag et al., 2025), we analyse a set of key cosmological observables, including the matter power spectrum, the halo mass function, and the internal properties of virilised haloes.

Particular attention is devoted to a detailed investigation of the treePM (Springel, 2005) scheme employed by OpenGadget3. By exploring the effects of different configuration flags and numerical parameters, we quantify their impact on cosmological observables at the percent-level.

Sommario

Le Graphics Processing Units (GPU) hanno assunto negli ultimi anni un ruolo sempre più centrale nel calcolo scientifico ad alte prestazioni (*high-performance computing, HPC*). Nell'ambito delle simulazioni cosmologiche, che richiedono spesso diversi milioni di ore di tempo computazionale, l'impiego delle GPU può migliorare significativamente le prestazioni, consentendo di simulare volumi di universo più estesi e con una risoluzione più elevata.

In questo lavoro di tesi magistrale analizziamo l'implementazione del gravity solver nel codice per le simulazioni cosmologiche OpenGadget3, con l'obiettivo di valutarne l'accuratezza numerica su diverse architetture hardware e in diverse configurazioni di simulazione. A tal fine utilizziamo simulazioni dark-matter-only, generate a partire da condizioni iniziali tratte dalla campagna di simulazioni Magneticum (Dolag et al., 2025). Su tali simulazioni analizziamo alcuni fondamentali osservabili cosmologici, tra cui il lo spettro di potenza (matter power spectrum), la funzione di massa degli aloni (*halo mass function*) e le proprietà degli aloni virializzati.

Particolare attenzione è dedicata a un'analisi dettagliata dello schema numerico TreePM (Springel, 2005) implementato in OpenGadget3. Esplorando l'effetto di diversi *Flag* di compilazione e di parametri numerici, quantifichiamo il loro impatto sugli osservabili cosmologici in termini percentuali.

Contents

Aim of the thesis

Our understanding of the Universe has advanced dramatically over the past few decades, culminating in the concordance cosmological model, the ΛCDM paradigm. ΛCDM is built on General Relativity and the cosmological principle, and presents a framework in which the Universe comprises, in order of abundance today, dark energy, represented by the cosmological constant Λ, cold dark matter (CDM), baryonic matter, and radiation. This model has been remarkably successful in explaining the accelerated expansion of the Universe (Perlmutter et al., 1999), the Cosmic Microwave Background (CMB) anisotropies (Ade et al., 2014), and the distribution of large-scale structures.

Despite the overall success of this model in explaining a wide range of observations, important questions remain regarding the nature of dark matter and dark energy, and about some of the tensions in ΛCDM (Planck Collaboration et al., 2020; Di Valentino et al., 2021; Perivolaropoulos & Skara, 2022). Addressing these questions motivates the next generation of high-precision surveys, such as Euclid (Mellier et al., 2025) and DESI (Aghamousa et al., 2016), which aim to map the large-scale structure of the Universe with unprecedented accuracy. The interpretation of these results requires simulations that can reliably simulate large volumes of the Universe at appropriate resolution and with sufficient statistical consistency. The numerical challenge, therefore, is to push the limits of efficiency, enabling larger and higher-resolution simulations while quantifying accuracy and convergence.

Recent advances in GPU-accelerated cosmological codes have enabled large, high-resolution N-body simulations. This work focuses on testing the numerical implementation of the gravity solver in OpenGadget3, with particular focus on the GPU porting of the algorithms devoted to the computation of gravitational forces (Ragagnin et al., 2020).

The structure of the thesis is as follows:

- In Chapter 1 we introduce the theoretical background of the thesis. We present the fundamental equations of cosmology in a homogeneous and isotropic Universe,

the ΛCDM concordance model, the linear theory of structure formation, the statistical properties of the Universe, and analytical models describing structure formation beyond the linear regime, motivating the need for numerical simulations in cosmology.

- In Chapter 2 we introduce the N-body method for cosmological simulations, and we describe the numerical techniques used to compute gravitational interactions, highlighting their features and drawbacks.
- In Chapter 3, we introduce fundamental concepts of parallel computing on CPUs and GPUs, starting with an overview of hardware features and moving on to how these translate into programming models. We highlight how parallelisation on CPU and GPU is achieved.
- In Chapter 4 we present the cosmological simulation code OpenGadget3. Building on the details provided in the literature (Springel, 2005), we present a novel description of the implementation and parallelisation of the tree in the gravity solver and the different flow of the algorithm depending on compilation configuration and numerical setup, focusing on MPI, OpenMP parallelisation, and GPU offloading.
- In Chapter 5 we present the validation strategy for the gravity solver on GPU, comprising tests on the matter power spectrum, the halo mass function, and the radial profiles of virialised haloes.
- In Chapter 6 we extend our analysis of the gravity solver to the setup of the hybrid TreePM method, focusing on how the numerical setup impacts convergence and reliability in the dark matter power spectrum.
- Finally, we draw our conclusions and discuss future prospects.
- Appendix A complements the discussion of parallel programming models by presenting an example of the order-of-magnitude speed-up provided by GPUs for embarrassingly parallel problems. It shows performance tests of CAM(L)AMBERT, a parallel solver designed for Monte Carlo simulations of the Lambert problem, developed during an internship at the European Space Agency.
- Appendix B provides additional plots supporting the analysis presented in the main chapters.

I

Simulations in the Era of High-Precision Cosmology

Chapter 1

The ΛCDM model and Structure Formation

Cosmology studies the properties and evolution of the Universe on the largest scales. In this chapter we present the fundamental theoretical concepts underlying modern cosmology and structure formation, mainly following Coles & Lucchin (1995). We begin by presenting the cosmological principle and the Friedmann–Lemaître–Robertson–Walker (FLRW) metric and by showing that the evolution of the Universe is described by the Friedmann equations. We discuss how the energy densities of the different cosmological components evolve in an expanding Universe. After introducing the main features of the ΛCDM model, currently the most accepted cosmological model, we turn to the theory of structure formation and examine how present-day galaxies and galaxy clusters originate from primordial density fluctuations.

On large cosmological scales, the dominant interaction is gravity. The currently accepted theory describing gravitational phenomena is Einstein's theory of General Relativity (GR). According to GR, gravity is a manifestation of spacetime curvature. Spacetime is modelled as a four-dimensional manifold where points are called *events*. Events in spacetime are described by a temporal and three spatial coordinates. The geometry of spacetime is fully described by the metric tensor $g_{\mu\nu}$, which is a symmetric rank-2 tensor. In four dimensions, this symmetry implies 10 independent components.

The metric tensor allows us to define invariant distances between events through the line element:

$$ds^2 = g_{\mu\nu}\, dx^\mu dx^\nu, \tag{1.1}$$

Once the metric is specified, all geometric quantities, such as curvature, can be derived from it.

Most modern cosmological models are based on the *cosmological principle*, which states that the Universe is homogeneous and isotropic on sufficiently large scales. This assumption is not valid today on small scales (below roughly hundreds of Mpc_s), where gravitational instability has led to the formation of highly non-uniform structures such as

galaxies and clusters of galaxies, known as the *Large Scale Structure* (LSS) of the Universe. Vice versa, when averaged over sufficiently large volumes, the matter distribution appears statistically uniform. Strong observational support for this hypothesis comes from the discovery of the Cosmic Microwave Background (CMB) radiation (Penzias & Wilson, 1965; Dicke et al., 1965), and from its measurement with increasing accuracy by missions such as COBE (Mather et al., 1990) WMAP (Bennett et al., 2013) and Planck (Ade et al., 2014). These observations show that, approximately 3×10^5 years after the Big Bang, the Universe was extremely homogeneous and isotropic, with relative density fluctuations of order:

$$\frac{\delta\rho}{\rho} \sim 10^{-5}. \tag{1.2}$$

These perturbations constitute the seeds from which present-day structures eventually formed through gravitational collapse.

The symmetries imposed by the cosmological principle strongly constrain the possible form of the metric tensor. The most general metric compatible with these symmetry requirements is the Friedmann-Lemaître-Robertson-Walker (FLRW) metric. In this metric, the invariant line element can be written as, assuming from now on $c = 1$:

$$ds^2 = -dt^2 + a^2(t)\left[\frac{dr^2}{1-kr^2} + r^2\left(d\theta^2 + \sin^2\theta\, d\phi^2\right)\right]. \tag{1.3}$$

Here, t is the cosmic time, defined as the proper time measured by comoving observers, r, θ, ϕ are comoving spatial polar coordinates; $a(t)$ is the scale factor, describing the expansion (or contraction) of the Universe; and k is the spatial curvature parameter, which takes the values, depending on the geometry of the universe:

$$k = \begin{cases} +1 & \text{spherical geometry,} \\ 0 & \text{flat geometry,} \\ -1 & \text{hyperbolic geometry.} \end{cases} \tag{1.4}$$

1.1 Friedmann equations

In the introduction, we mentioned that the scale factor $a(t)$ evolves in a homogeneous and isotropic universe. Einstein Field equations describe this evolutionion:

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

where T_{ij} is the *energy-momentum tensor* , which for a perfect fluid takes the form:

$$T^{\mu}_{\ \nu} = \begin{pmatrix} -\rho & 0 & 0 & 0 \\ 0 & p & 0 & 0 \\ 0 & 0 & p & 0 \\ 0 & 0 & 0 & p \end{pmatrix}. \tag{1.6}$$

Typically, the Universe is treated as a perfect fluid, with mass-energy density ρ and pressure p, and thermal conduction and viscous processes are neglected. Using the FLRW metric and the energy-momentum tensor for a perfect fluid, the Einstein Field equations give rise to the Friedmann Equations:

$$\begin{aligned} \dot{a}^2 + k &= \frac{8}{3}\pi G\rho a^2, \\ \ddot{a} &= -\frac{4}{3}\pi G\left(\rho + p\right)a, \end{aligned} \tag{1.7}$$

where the dot represents the derivative with respect to the proper time. We can rearrange the terms in the first Friedmann equation to isolate the curvature

$$\frac{k}{a^2} = \left(\frac{\dot{a}}{a}\right)^2 \left(\frac{\rho}{\rho_c} - 1\right), \tag{1.8}$$

and defining the critical density ρ_c as the density required for the spatial curvature to vanish ($k = 0$):

$$\rho_c = \frac{3}{8\pi G}\left(\frac{\dot{a}}{a}\right)^2 \tag{1.9}$$

If $\rho > \rho_c$, then $\frac{\rho}{\rho_c} - 1 > 0$, implying $k > 0$ and the Universe has positive spatial curvature and is said to be *closed*. For $\rho < \rho_c$, then $\frac{\rho}{\rho_c} - 1 < 0$, implying $k < 0$ and the Universe has negative spatial curvature and is referred to as *open*. If $\rho = \rho_c$, then $k = 0$, and the Universe is spatially flat.

It is convenient to introduce the *Hubble parameter*, which measures the expansion rate of the Universe at cosmic time t:

$$H(t) \equiv \frac{\dot{a}(t)}{a(t)}, \tag{1.10}$$

so that the first Friedmann equation (1.7) can be rewritten as:

$$H^2(t) + \frac{k}{a^2} = \frac{8\pi G}{3}\rho. \tag{1.11}$$

The current value of the Hubble parameter, i.e. measured at the present cosmic time t_0, is called the *Hubble constant* H_0 and is parametrised by a dimensionless parameter h:

$$H_0 \equiv H(t_0) = 100 \text{ h } kms^{-1}Mpc^{-1}. \tag{1.12}$$

The parameter h is observationally constrained to values between 0.67 and 0.73, depending on the measurement method. Local distance–ladder measurements generally favour higher values, while estimates derived from Cosmic Microwave Background observations within the ΛCDM framework yield lower values. This discrepancy is commonly referred to as the *Hubble tension*.

A direct observable consequence of cosmic expansion is the cosmological redshift. Consider two comoving observers separated by some distance in the expanding Universe. Suppose that one of them emits a monochromatic electromagnetic wave with wavelength λ_e, where the subscript "e" denotes the emitted quantity. The signal is received by the second observer at the present time t_0 with wavelength λ_o.

In a relativistic framework, electromagnetic radiation propagates along null geodesics, for which the space-time interval satisfies:

$$ds^2 = 0. \tag{1.13}$$

Imposing this condition on the FLRW metric shows that the time interval between the arrival of two successive wave crests at the observer, δt_o, is related to the time interval between their emission, δt_e, through

$$\frac{\delta t_\mathrm{o}}{\delta t_\mathrm{e}} = \frac{a_0}{a_\mathrm{e}}, \tag{1.14}$$

where a_0 and a_e are the values of the scale factor at the time of observation t_0 and at the time of emission t_e, respectively. In an expanding Universe $a_0 > a_\mathrm{e}$, implying $\delta t_\mathrm{o} > \delta t_\mathrm{e}$. Since the wavelength of the radiation is proportional to the period of the wave, $\lambda \propto \delta t$, the observed wavelength is stretched with respect to the emitted one.

This leads to the definition of the cosmological redshift,

$$z \equiv \frac{\lambda_\mathrm{o} - \lambda_\mathrm{e}}{\lambda_\mathrm{e}} = \frac{a_0}{a_\mathrm{e}} - 1. \tag{1.15}$$

It is customary to normalise the present-day scale factor to unity ($a_0 = 1$), so that the relation simplifies to:

$$a = \frac{1}{1+z}. \tag{1.16}$$

1.1.1 Friedmann equations with a Cosmological Constant

From the second Friedmann equation, it can be observed that the second derivative over proper time of the scale factor ($\ddot{a}(t)$) is always non-zero, unless the following condition is met:

$$\rho = -3\rho. \tag{1.17}$$

This means that density and pressure must be opposite in sign for the Universe to be static. The conviction that the Universe must be static is the original reason why Einstein proposed a modified version of his field equations:

$$R_{ij} - \frac{1}{2}g_{ij}R - \Lambda g_{ij} = 8\pi T_{ij} \tag{1.18}$$

where Λ is known as the cosmological constant, and populated the geometrical side of the equation. Since the discovery of the expansion of the Universe in the 1920s, research for a static solution to the field equations had ceased. More recently, the cosmological constant has assumed a different role in modern cosmology, as the most widely accepted model for *Dark Energy*. In this context, it is usually written on the source term side of the equations, giving rise to the modified energy-momentum tensor :

$$\tilde{T_{ij}} = T_{ij} + \frac{\Lambda}{8\pi G}g_{ij} = -\tilde{p}g_{ij} + (\tilde{p} + \tilde{\rho})U_iU_j \tag{1.19}$$

The Friedmann equations can now be rewritten with the modified pressure and density arising from the presence of the cosmological constant as:

$$\begin{aligned} \ddot{a} &= -\frac{4}{3}\pi G(\rho + p)a + \frac{\Lambda a}{3} \\ \dot{a}^2 + k &= \frac{8}{3}\pi G\rho a^2 + \frac{\Lambda a^2}{3} \end{aligned} \tag{1.20}$$

Analogous arguments on the geometry of the Universe and the critical density can be repeated in this case using the effective pressure and density.

1.1.2 A multi-component universe

From the Friedmann equation, and using the adiabaticity principle $d(\rho a^3) = -p\, d(a^3)$, we can compute the time evolution of the density once the equation of state is specified. The equation of state links pressure to density through the *equation of state parameter* ω:

$$p = \omega\rho \tag{1.21}$$

For a single-component universe one obtains:

$$\rho = \rho_0 \left(\frac{a}{a_0}\right)^{-3(1+\omega)} \tag{1.22}$$

We call *dust* models that are pressureless and have $\omega = 0$. This is usually a good approximation for non-relativistic particles, since their thermal energy is much smaller than their rest mass. Ultrarelativistic particles and radiation, on the other hand, have $\omega = \frac{1}{3}$.

For a *radiation-dominated* universe we therefore obtain:

$$\rho_r = \rho_{0,r}(1+z)^4 \tag{1.23}$$

while for a *matter-dominated* universe:

$$\rho_m = \rho_{0,m}(1+z)^3 \tag{1.24}$$

For the cosmological constant the density is constant:

$$\rho_\Lambda = \frac{\Lambda}{8\pi G} = \text{const},$$

and the corresponding equation of state parameter is $\omega = -1$.

The temporal evolution of the energy densities is illustrated in Fig. 1.1. Each component undergoes a period of dominance during cosmic history. At very early times the Universe can be approximated as radiation-dominated.

The *matter–radiation equivalence* occurs when the densities of matter and radiation are equal. From the relations above we obtain the corresponding redshift:

$$\rho_{0,m}(1+z_{\text{eq}})^3 = \rho_{0,r}(1+z_{\text{eq}})^4 \;\Rightarrow\; z_{\text{eq}} = \frac{\rho_{0,m}}{\rho_{0,r}} - 1 \tag{1.25}$$

After this epoch the Universe becomes matter-dominated until the *matter–dark energy equivalence*, when the densities of matter and dark energy are the same:

$$\rho_{0,m}(1+z_{\text{eq},\Lambda})^3 = \rho_\Lambda \;\Rightarrow\; z_{\text{eq},\Lambda} = \left(\frac{\rho_\Lambda}{\rho_{0,m}}\right)^{1/3} - 1 \tag{1.26}$$

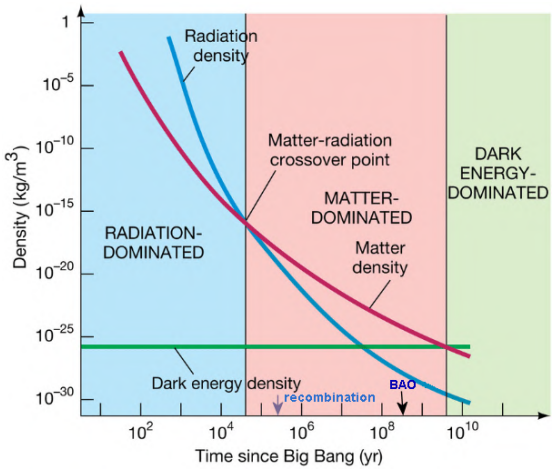


Figure 1.1. Evolution of the energy density of the different components of the Universe with cosmic time, showing the transitions between the radiation-, matter-, and dark-energy-dominated eras. (from : https://universe-review.ca/I02-19-CosmicEra.png)

This transition occurred relatively recently in cosmic history. While for $z \gg z_{\rm eq}$ the Universe can be approximated as radiation-dominated, and for intermediate redshifts it is often convenient to treat it as matter-dominated (and it is often referred to as *Einstein-de-Sitter* or *EdS* universe), today we are not sufficiently far from the dark energy–matter equality to neglect the contribution of dark matter. Therefore, a complete description of the expansion of our multi-component Universe requires accounting for all components simultaneously.

We define the density parameter for a generic *i-th* component as $\Omega_i = \frac{\rho_i}{\rho_c}$. This allows us to rewrite the first Friedmann Equation using the density parameters of every component at *present-day*:

$$\left(\frac{\dot{a}}{a_0}\right)^2 = H_0^2 \left[\sum_i \Omega_{0,i} \left(\frac{a_0}{a}\right)^{1+3w_i} + \left(1 - \sum_i \Omega_{0,i}\right)\right]. \tag{1.27}$$

1.2 The ΛCDM paradigm

The ΛCDM model, or Standard Cosmological Model, is currently the most successful framework for describing the Universe. It considers a flat Universe ($\Omega_0 = 1$) in an expansion state starting from an initial singularity, the Big Bang, characterised by divergent temperature and density. During the expansion history of the Universe,

the temperature progressively decreased, allowing the different components of the mass-energy fluid, initially coupled, to decouple at different cosmic times.

The model is named after its two dominant(today) components:

- *Dark Energy (Λ):* modelled as a cosmological constant, is responsible for the today accelerated expansion of the Universe, it contributes $\sim 68\%$ of the total mass-energy, with a density $\rho_\Lambda \sim 7 \times 10^{-30}\,\mathrm{g\,cm^{-3}}$.

- *Cold Dark Matter (CDM):* Accounting for $\sim 27\%$, this non-relativistic, collisionless component interacts only through gravity. CDM is the main responsible for the formation of cosmic structures such as galaxies and clusters, and ist's presence is inferred from galaxy rotation curves, cluster dynamics, gravitational lensing, and large-scale structure (LSS).

The remaining $\sim 5\%$ is *baryonic matter*, primarily light elements like Hydrogen and Helium. Additionally, the Standard Model predicts relic *radiation* (CMB) and the *Cosmic Neutrino Background*, contributing only for the 10^{-5} of the total budget.

The ΛCDM model is capable of accurately reproducing and predicting properties of our universe using six cosmological parameters, listed in Tab. 1.1.

Parameter	Description	Value (Planck 2020)
$\Omega_{0,m}$	Total matter density parameter	$\Omega_{0,m}h^2 = 0.143 \pm 0.001$
$\Omega_{0,b}$	Baryonic matter density parameter	$\Omega_{0,b}h^2 = 0.0224 \pm 0.0001$
H_0	Hubble constant	$67.4 \pm 0.5\ \mathrm{km\,s^{-1}\,Mpc^{-1}}$
A_s	Primordial power spectrum amplitude	$\ln(10^{10}A_s) = 3.04 \pm 0.01$
n_s	Spectral index of primordial power spectrum	0.965 ± 0.004
τ	Reionisation optical depth	0.054 ± 0.007

Table 1.1. Fundamental cosmological parameters of the ΛCDM model measured by Planck (Planck Collaboration et al., 2020).

Despite the many successes of ΛCDM, there is currently no fully established theoretical framework to explain the fundamental nature of dark matter and dark energy. Together with the unknown dark components, the ΛCDM model faces several unresolved questions and observational tensions, which may hint at new physics. Among the most prominent are:

- the Hubble tension (H_0): a discrepancy between the expansion rate measured locally via distance ladders and the value inferred from the CMB under ΛCDM ;

- the S_8 tension: a mismatch in the amplitude of matter clustering between weak lensing surveys and CMB predictions;
- small-scale structure issues, such as the core–cusp problem in galaxy haloes and the "too-big-to-fail" problem.

Moreover, a potential tension could arise from recent high-redshift galaxy surveys that find galaxies too massive or too bright too early, potentially challenging standard structure formation scenarios.

These tensions and anomalies have been discussed in detail in recent reviews (Perivolaropoulos & Skara, 2022; Di Valentino et al., 2021; Planck Collaboration et al., 2020) highlighting the limits of ΛCDM and motivating explorations of extensions or modifications to the standard model.

1.3 Linear Regime for Structure formation

To study the growth of structures, we investigate the effect of self-gravity on overdensities in an otherwise homogeneous fluid. This problem was first addressed in 1902 by Jeans in his study of gravitational instability. He demonstrated that small fluctuations in density and velocity could evolve under the effect of self-gravity and found a characteristic length scale over which these overdensities grow, known today as *Jeans' length* λ_J. To obtain an order of magnitude estimate of the *Jeans length* we consider a spherical inhomogeneity of radius λ containing a small positive density fluctuation $\delta\rho$ of mass M, in a background fluid of mean density ρ. The overdensity will grow if the self-gravitational force per unit mass exceeds the pressure force per unit mass:

$$F_g \simeq \frac{GM}{\lambda^2} \simeq \frac{G\rho\lambda^3}{\lambda^2} > \frac{p\lambda^2}{\rho\lambda^3} \simeq \frac{v_s^2}{\lambda} \tag{1.28}$$

This order-of-magnitude estimate could also be obtained by comparing the self-energy per unit mass with the kinetic energy per unit mass, or by requiring that the free-fall time be shorter than the hydrodynamic time. In all cases, this leads to an estimate of the Jeans length of $\lambda_J \simeq v_s(G\rho)^{-1/2}$. Density fluctuations with a lengthscale $\lambda > \lambda_J$ will collapse under the effect of self-gravity.

This picture gets more complicated in the context of cosmological structure formation, since we would need to study the formation of structures in a multi-component expanding universe.

A full treatment of the evolution of density perturbations in a cosmic fluid requires solving the collisionless Boltzmann equation coupled to the linearised Einstein equations.

This approach is complex and beyond the scope of this work. Therefore, we restrict ourselves to the *linear Jeans analysis* of a single-component, non-relativistic fluid in an expanding universe. This treatment is limited to scales in causal contact, which motivates defining the *particle horizon* at cosmic time t as the maximum distance from which light emitted at the Big Bang ($t = 0$) could have reached an observer by time t. Since photons move along null geodesics the line element is $ds = 0$. Considering only radially travelling photons ($d\theta = d\phi = 0$), we find that the cosmological comoving horizon is defined as:

$$\eta(t) = \int_0^t \frac{dt'}{a(t')} \tag{1.29}$$

In this section, we develop a theoretical description of structure formation through linear perturbation theory and beyond, and we aim to determine its limits, thereby clarifying the role of simulations in modern cosmology.

1.3.1 Gravitational instability in an expanding universe

We are interested in studying density perturbations in a fluid of Cold Dark Matter (CDM). We follow the evolution of the system using the Euler and continuity equations, coupled with the Poisson equation for gravity. These equations capture the essential dynamics of the system in the linear regime, as they represent the first moments of the collisionless Boltzmann equation (see Chap. 2). We write fluid equations as:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) &= 0 \\ \frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} + \frac{1}{\rho}\nabla p + \nabla \phi &= 0 \\ \nabla^2 \phi &= 4\pi G \rho \end{aligned} \tag{1.30}$$

where ρ is the density, p the pressure, v the velocity and ϕ the gravitational potential. We are neglecting viscosity and thermal conductivity. For this reason, we pair this with the equation for conservation of entropy per unit mass. In an expanding universe, it is convenient to use comoving coordinates $\boldsymbol{x}$, defined by $\boldsymbol{r} = a(t)\boldsymbol{x}$, and to decompose the velocity as $\boldsymbol{v} = H\boldsymbol{r} + a\boldsymbol{v}$, where $\boldsymbol{v} = \dot{\boldsymbol{x}}$ is the peculiar velocity and $H\boldsymbol{r}$ is the Hubble flow. Gradients transform as $\nabla_{\boldsymbol{r}} = \nabla_{\boldsymbol{x}}/a$. For the partial derivative of time, we look at how the Lagrangian derivative along the fluid flow transforms in comoving coordinates:

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \frac{1}{a}\boldsymbol{v} \cdot \nabla, \tag{1.31}$$

The fluid equations then become:

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + 3H\rho + \frac{1}{a}\nabla \cdot (\rho \boldsymbol{v}) = 0, \\
&\frac{\partial \boldsymbol{v}}{\partial t} + H\boldsymbol{v} + \frac{1}{a}(\boldsymbol{u} \cdot \nabla)\boldsymbol{v} = -\frac{1}{a\rho}\nabla p - \frac{1}{a}\nabla \phi, \\
&\nabla^2 \phi = 4\pi G a^2 (\rho - \rho_b).
\end{aligned} \tag{1.32}$$

The system is supplemented by the conservation of entropy along the flow. Having established the fluid equations in an expanding universe, we now consider *small perturbations* around a homogeneous background,

$$\rho(\boldsymbol{x}, t) = \bar{\rho}(t) + \delta\rho(\boldsymbol{x}, t), \qquad \boldsymbol{v}(\boldsymbol{x}, t) = \delta\boldsymbol{v}(\boldsymbol{x}, t), \qquad \phi(\boldsymbol{x}, t) = \bar{\phi}(t) + \delta\phi(\boldsymbol{x}, t), \tag{1.33}$$

where $\bar{\rho}(t)$ and $\bar{\phi}(t)$ are the background density and potential, and $\delta\rho, \delta\boldsymbol{v}, \delta\phi$ are small perturbations. Assuming these perturbations are sufficiently small, the continuity, Euler, and Poisson equations can be *linearised*, expanding them to the first order around a solution for the homogeneous background:

$$\begin{aligned}
\frac{\partial \delta\rho}{\partial t} + \rho_0 \nabla \cdot \delta\mathbf{v} = 0, \\
\frac{\partial \delta\mathbf{v}}{\partial t} + \frac{1}{\rho_0}\left(\frac{\partial p}{\partial \rho}\right)_s \nabla\delta\rho + \frac{1}{\rho_0}\left(\frac{\partial p}{\partial s}\right)_\rho \nabla\delta s + \nabla\delta\phi = 0, \\
\nabla^2\delta\phi - 4\pi G\delta\rho = 0.
\end{aligned} \tag{1.34}$$

Since these equations are linear in the perturbations we can decompose any arbitrary density fluctuation into a superposition of plane waves using Fourier analysis. In this linear approximation, each Fourier mode evolves independently, without coupling to other modes. Therefore, it is sufficient to study the evolution of a single mode, $\delta\rho \propto \delta_k(t) e^{i\boldsymbol{k}\cdot\boldsymbol{x}}$. Doing so, the equations reduce to the following relation:

$$\ddot{\delta}_k + 2H\dot{\delta}_k + \delta_k\left(k^2 c_s^2 - 4\pi G\rho_b\right) = 0, \tag{1.35}$$

where ρ_b is the background density and c_s is the effective sound speed. The first derivative term is known as *Hubble friction*, as it is the term that slows down the accretion due to the *Hubble flow*.

In a collisionless cold dark matter fluid, particles do not experience true thermodynamic pressure. However, the random motions of particles within a small volume give

rise to an effective velocity dispersion, described by the tensor:

$$\sigma_{ij}^2 = \langle v_i v_j \rangle - \langle v_i \rangle \langle v_j \rangle.$$

For large-scale, isotropic perturbations (which we can assume given the cosmological principle) we assume the tensor to be isotropic:

$$\sigma_{ij}^2 \approx c_s^2 \, \delta_{ij},$$

where c_s is the effective sound speed that we saw in Eq. 1.35.

We get for the *Jeans wavelength*:

$$\lambda_J = \frac{2\pi}{k_J} = c_s \sqrt{\frac{\pi}{G\rho_b}}. \tag{1.36}$$

Perturbations with $\lambda < \lambda_J$ oscillate as sound waves, while for scales over the Jeans ($\lambda > \lambda_J$) but still within the particle horizon, the dispersion relation leads to a growing and a decaying solution.

Assuming a matter-dominated EdS universe we obtain:

$$\begin{cases} \delta_+(t) \propto a \propto t^{2/3} \\ \delta_-(t) \propto a^{-2/3} \propto t^{-1} \end{cases} \tag{1.37}$$

The decaying solution over time decreases in amplitude; we are just interested in the growing mode.

During the radiation-dominated era, the growth of cold dark matter perturbations within the horizon scale is strongly suppressed by the Hubble friction in eq. 1.35. For sub-horizon modes in this regime, the evolution equation for matter perturbations leads to a growing solution of the form:

$$\delta_+ \propto \ln a, \tag{1.38}$$

instead of the linear growth $\delta_m \propto a$ in matter domination. This represents the asymptotic behaviour at $z >> z_{eq}$. As first derived in Meszaros (1974), evaluating the full solution at $a = a_{\rm eq}$ shows that sub-horizon perturbations have grown only by a finite factor of 5/2 during radiation domination. This suppression of growth during radiation domination is known as the Mészáros effect.

1.3.2 An approximate treatment for super-horizon scales

As previously mentioned, above the scales of the horizon the Newtonian Jeans analysis is no longer valid, since causal processes cannot operate on super-horizon scales. A fully relativistic treatment is required. An alternative, physically intuitive way to obtain an approximate description of the evolution of such perturbations is to treat the perturbation as a closed universe embedded in a single-component background. This construction is justified by Birkhoff's theorem, which states that the gravitational field inside a spherically symmetric shell depends only on the mass enclosed within it. Writing the Friedmann equation for the background (EdS) and for the overdensity one finds:

$$H_p^2 = \frac{8\pi G}{3}\rho_p, H_b^2 = \frac{8\pi G}{3}\rho_b \tag{1.39}$$

where the subscript p denotes the perturbation, and b the background. The scale factors are initially the same, so we can equate the above relations and find:

$$\delta = \frac{\rho_p - \rho_b}{\rho_b} = \frac{3}{8\pi G \rho_b a^2} \tag{1.40}$$

where we define δ as the *density contrast*. The relation above implies that the evolution of the density contrast depends on the background density. We can distinguish two behaviours for the evolution of the overdensities on super-horizon scales, depending on whether we are in a radiation-dominated or a matter-dominated era:

$$\begin{cases} \delta = \delta_r \propto a^2 \propto t \text{ for } z > z_{eq} \\ \delta = \delta_m \propto a \propto t^{2/3} \text{ for } z < z_{eq} \end{cases} \tag{1.41}$$

Since the components of the fluid are paired through gravity, the behaviour of the dominant component determines the behaviour of the other components.

1.4 Statistical properties of the universe

We have seen where the analytical approach to structure formation stems from and summarised the growth rate of structures before and after the radiation-matter equivalence both in and outside of the horizon scale, and we are now interested in seeing how we can connect it to primordial perturbations.

1.4.1 Correlation function and the Power Spectrum

Today's Universe exhibits significant inhomogeneity on scales below several hundred Mpc. This is because, as the process of structure formation proceeds, the density contrast moves out of the linear regime, and gravitational effects lead to the formation of structures. Today we know, thanks to Planck Collaboration et al. (2020), that the amplitude of the perturbations in the temperature field at the time of recombination is of the order of $\frac{\delta T}{t} \simeq 10^{-5}$, which also reflects in the density contrast. These CMB anisotropies are our means of measuring the amplitude of primordial perturbations at the time of recombination, which then grew into today's Large Scale Structure (LSS). To keep track of the growth we need to adopt a statistical approach. Because the Universe is assumed to be statistically homogeneous and isotropic, in accordance with the cosmological principle, we can invoke the ergodic hypothesis and replace ensemble averages with spatial averages over sufficiently large volumes. This allows us to characterise cosmological perturbations statistically. Inside a volume V, the density perturbation can be decomposed in Fourier modes:

$$\delta(\mathbf{x}) = \int \frac{d^3k}{(2\pi)^3} \delta(\mathbf{k})\, e^{i\mathbf{k}\cdot\mathbf{x}}. \tag{1.42}$$

where k is the wave vector of x, and $\delta(k)$ is the Fourier transform of the perturbation density in the Fourier space and can be written as:

$$\delta(\mathbf{k}) = \int d^3x\, \delta(\mathbf{x})\, e^{-i\mathbf{k}\cdot\mathbf{x}} \tag{1.43}$$

Since $\delta(\mathbf{x})$ is real, its Fourier transform satisfies the condition:

$$\delta^*(\mathbf{k}) = \delta(-\mathbf{k}). \tag{1.44}$$

The first statistical quantity we can consider is the two-point correlation function. This function describes the auto-correlation of the field δ with itself in positions separated by a distance r

$$\xi(\mathbf{r}) \equiv \langle \delta(\mathbf{x})\, \delta(\mathbf{x}+\mathbf{r}) \rangle\,. \tag{1.45}$$

The Power Spectrum is defined via:

$$\langle \delta(\mathbf{k})\, \delta^*(\mathbf{k}') \rangle = (2\pi)^3 \delta_D(\mathbf{k}-\mathbf{k}')\, P(k) \tag{1.46}$$

where δ_D is the Dirac delta function. Substituting the Fourier transform into the definition of ξ gives:

$$\xi(\mathbf{r}) = \int \frac{d^3k}{(2\pi)^3} P(k) e^{i\mathbf{k}\cdot\mathbf{r}}. \tag{1.47}$$

The power spectrum is the Fourier transform of the correlation function:

$$P(k) = \int d^3r\, \xi(r) e^{-i\mathbf{k}\cdot\mathbf{r}} \tag{1.48}$$

Moreover, we can connect the variance of the density field to the power spectrum. Starting from the definition of variance:

$$\sigma^2 \equiv \left\langle \delta^2(\mathbf{x}) \right\rangle \tag{1.49}$$

and using the expression in Fourier modes:

$$\delta(\mathbf{x}) = \int \frac{d^3k}{(2\pi)^3} \delta(\mathbf{k}) e^{i\mathbf{k}\cdot\mathbf{x}} \tag{1.50}$$

we obtain for the variance:

$$\sigma^2 = \left\langle \int \frac{d^3k}{(2\pi)^3} \int \frac{d^3k'}{(2\pi)^3} \delta(\mathbf{k}) \delta^*(\mathbf{k}') e^{i(\mathbf{k}-\mathbf{k}')\cdot\mathbf{x}} \right\rangle \tag{1.51}$$

and since:

$$\left\langle \delta(\mathbf{k}) \delta^*(\mathbf{k}') \right\rangle = (2\pi)^3 \delta_D(\mathbf{k} - \mathbf{k}') P(k) \tag{1.52}$$

we obtain:

$$\sigma^2 = \int \frac{d^3k}{(2\pi)^3} P(k) \tag{1.53}$$

assuming isotropy (thanks to the cosmological principle), we can rewrite the variance as a one-dimensional integral:

$$\sigma^2 = \frac{1}{2\pi^2} \int_0^\infty dk\, k^2 P(k) \tag{1.54}$$

It is common to see it written in terms of the *dimensionless power spectrum* ($\Delta^2(k)$) as:

$$\Delta^2(k) = \frac{k^3}{2\pi^2} P(k) \tag{1.55}$$

$$\sigma^2 = \int_0^\infty \frac{dk}{k} \Delta^2(k) \tag{1.56}$$

From a practical perspective the density contrast is not measured with infinite

resolution. It is therefore useful to write the variance of the density contrast smoothed on a scale R as:

$$\sigma^2(R) \equiv \left\langle \delta_R^2(\mathbf{x}) \right\rangle \tag{1.57}$$

where the smoothed density contrast is

$$\delta_R(\mathbf{x}) = \int d^3x' \, W_R(\mathbf{x} - \mathbf{x}') \, \delta(\mathbf{x}') \tag{1.58}$$

and W_R is a window function. Using the Fourier transform, the smoothed field becomes

$$\delta_R(\mathbf{k}) = \delta(\mathbf{k}) W_R(\mathbf{k}) \tag{1.59}$$

and the variance smoothed at length R can be written as:

$$\sigma_R^2(R) = \frac{1}{2\pi^2} \int_0^\infty dk \, k^2 P(k) |W_R(k, R)|^2 \tag{1.60}$$

1.4.2 The Matter Power spectrum

According to current cosmological models, the formation of primordial perturbations in the metric is attributed to a phase of accelerated expansion, known as *inflation*. Inflation refers to a very brief period in the early Universe during which the scale factor increased exponentially. It was originally proposed to address several fundamental issues of the standard Big Bang model: it explains the observed large-scale homogeneity and isotropy of the Universe (the horizon problem), accounts for the near-flat spatial geometry (the flatness problem), and suppresses the abundance of unwanted relics such as magnetic monopoles (the monopole problem). Furthermore, quantum fluctuations generated during inflation provide the seeds for the formation of cosmic structures observed today. These primordial fluctuations are predicted to be nearly Gaussian and almost scale invariant, with no preferred length scale. This translates into a power-law primordial power spectrum:

$$P_{in}(k) = A_s(k)^{n_s} \tag{1.61}$$

where As and ns are referred to as the scalar amplitude and spectral index, respectively. Several inflationary models predict a scale-independent amplitude of primordial potential fluctuations; this is realised for $n_s = 1$, and the corresponding power spectrum is referred to as the Harrison-Zel'dovich power spectrum (Harrison, 1970; Zeldovich, 1972).

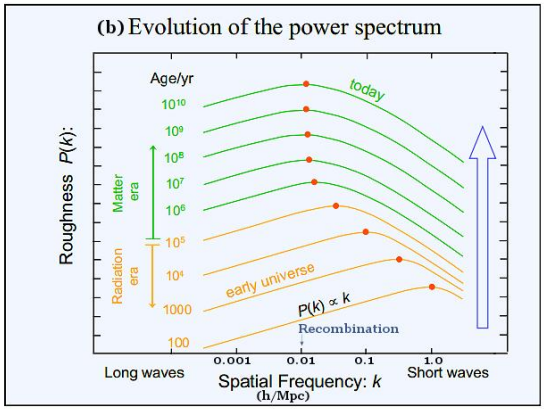


Figure 1.2. Evolution of the matter power spectrum at different cosmic times. The peak progressively shifts toward larger scales as the Universe evolves, approaching the characteristic scale set by matter–radiation equality. (from : https://universe-review.ca/R05-04-powerspectrum.htm)

On the one hand, we have a primordial power spectrum that is scale-independent; on the other hand, a process of structure formation that depends on the size of the perturbations. For this reason it is useful to write the Power Spectrum as a function of the scale factor, within linear theory, as:

$$P_{lin}(k, a) = D^2(a)T^2(k)P_{in}(k) \tag{1.62}$$

The primordial power spectrum is modified by the *linear growth factor* $D^2(a)$ and *the transfer function* $T^2(k)$, which accounts for the scale dependence of structure formation.

Figure 1.2 shows the linear matter power spectrum, as it evolves through cosmic history. On large scales (small k), the linear power spectrum preserves the primordial logarithmic slope n_s, while on small scales (large k), it decreases with slope $n_s \approx 3$ (this value for a Harrison-Zel'dovich primordial power spectrum). These two regimes are separated by a maximum at $k_{eq} = 2\pi\lambda_{eq} \approx 2 \cdot 10^{-2}$ hMpc^{-1} where λ_{eq} is the size of the comoving particle horizon at the time of matter-radiation equivalence, occurred at redshift $z_{eq} \approx 3400$. Modes with k >keq entered the horizon at some time during radiation domination in this period; the sub-horizon growth of the amplitude of matter perturbations was strongly suppressed due to the Mészáros effect. Modes with larger k entered the horizon earlier, suffering for a longer time from stagnation, so that for increasing $k > k_{eq}$ the power spectrum decreases. Modes with $k < k_{eq}$ did not enter the horizon before equivalence. Being their growth factor scale-independent, the power

spectrum preserved the primordial slope. For $z < z_{eq}$, during matter domination, both super-horizon and sub-horizon fluctuations are characterised by the same growth factor, and the linear power spectrum evolution simply corresponds to a k-independent rescaling. Several phenomena change the shape of the power spectrum as perturbations evolve. Modes with large $k = 2\pi/R$ are the first to become nonlinear, and this manifests with an increase of power at large k.

1.5 Beyond Jeans linear theory

1.5.1 Spherical Collapse

As the density contrast approaches unity, the fluctuation distribution ceases to be Gaussian, and the linear regime is no longer valid. In this section, we describe the spherical collapse model (Gunn & Gott, 1972), an analytical approach relevant to the nonlinear nature of structure formation. In this model, we treat the overdensity that will lead to a virialised structure as a closed universe embedded in a flat, single-component background. Since we must work in a single-component universe, the treatment begins well after the epoch of equivalence, in a matter-dominated universe. While a spherical overdensity is not representative of cosmological applications, given that real fluctuations are irregular, this model still provides useful insights.

We follow the evolution of a sphere whose initial amplitude is still in the linear regime. The spherical perturbation of constant density, has an initial amplitude : $\delta_i > 0$ at time $t_i \gg t_{\rm rec}$, and $|\delta_i| \ll 1$. The perturbation is assumed to expand initially with zero peculiar velocity at its edge. For an Einstein-de-Sitter background, the density contrast evolves, as:

$$\delta(t) = \delta_+(t_i)\left(\frac{t}{t_i}\right)^{2/3} + \delta_-(t_i)\left(\frac{t}{t_i}\right)^{-1}, \tag{1.63}$$

From the derivative of the density contrast, we find the velocity:

$$V = \frac{1}{k_i t_i}\left[\frac{2}{3}\delta_+(t_i)\left(\frac{t}{t_i}\right)^{-1/3} - \delta_-(t_i)\left(\frac{t}{t_i}\right)^{-4/3}\right]. \tag{1.64}$$

and by imposing the aforementioned boundary condition $V_i = 0$ we obtain a constraint on the amplitude of the growing mode:

$$\delta_+(t_i) = \frac{3}{5}\delta_i \tag{1.65}$$

We assume that the decaying mode becomes negligible. The effective density parameter of the perturbation at t_i :

$$\Omega_p(t_i) = \Omega(t_i)(1+\delta_i). \tag{1.66}$$

The structure will collapse if at some time t_m the spherical region will stop expanding with the background and will then collapse. This requires $\Omega_p(t_i) > 1$. This condition can be re written as a function of the of the present value of the density parameter Ω:

$$\delta_+(t_i) > \frac{3}{5}\frac{1-\Omega}{\Omega(1+z_i)}. \tag{1.67}$$

This shows us that for universes with $\Omega \geq 1$ the overdensity will always collapse, while on the other hand Ω would have to exceed a critical value. The perturbation evolves as a separate Friedmann model:

$$\left(\frac{\dot{a}}{a_i}\right)^2 = H_i^2\left[\frac{\Omega_p(t_i)}{a/a_i} + 1 - \Omega_p(t_i)\right] \tag{1.68}$$

The perturbation will reach maximum expansion at t_m:

$$t_m = \frac{\pi}{2H_i}\frac{\Omega_p(t_i)}{[\Omega_p(t_i)-1]^{3/2}} \tag{1.69}$$

for an Einstein-de Sitter universe the overdensity at turnaround is:

$$\chi \equiv \frac{\rho_p(t_m)}{\rho(t_m)} = \left(\frac{3\pi}{4}\right)^2 \simeq 5.6. \tag{1.70}$$

which corresponds to a perturbation $\delta_+ \simeq 4.6$. It is also interesting to extrapolate from the linear growth law what the density contrast would be at the same time when predicted from linear theory:

$$\delta_+(t_m) = \delta_+(t_i)(\frac{t_m}{t_i})^{2/3} = \frac{3}{5}\left(\frac{3\pi}{4}\right)^{2/3} \simeq 1.07. \tag{1.71}$$

At a time of the order of $2t_m$, ignoring pressure forces, the overdensity would have reached infinite density at the centre. This is not true, since pressure gradients and shocks would convert kinetic energy into heat. The system would therefore virialise. Virialisation occurs at $t_{\rm vir} \approx 3t_m$ according to numerical simulations. Using the virial theorem, one can also find a relation between the radius and the maximum expansion and the virialised radius.

$$R_m = 2R_{\rm vir}.$$

The ratio between the overdensity at collapse and at virialisation are:

$$\frac{\rho_p(t_c)}{\rho(t_c)} = 2^2 8\chi \simeq 180 \tag{1.72}$$

$$\frac{\rho_p(t_{\rm vir})}{\rho(t_{\rm vir})} = 3^2 8\chi \simeq 400 \tag{1.73}$$

respectively. Linear extrapolation gives the critical collapse threshold:

$$\delta_+(t_c) \simeq 1.686 \rightarrow \frac{\rho_p(t_c)}{\rho(t_c)} \simeq 2.68 \tag{1.74}$$

$$\delta_+(t_{vir}) \simeq 2.20 \rightarrow \frac{\rho_p(t_{\rm vir})}{\rho(t_{\rm vir})} \simeq 3.20 \tag{1.75}$$

So we find that at the time of collapse the density contrast from the spherical collapse model is $\delta sph(t_c) \simeq 180$, while the extrapolation of linear perturbation theory would give $\delta_{lin}(t_c) \simeq 1.686$.

1.5.2 Press-Schechter formalism

A fundamental application for this linear extrapolation is the Press-Schechter model for the mass function n(M) (Press & Schechter, 1974). The mass function is defined by :

$$dN = n(M)dM \tag{1.76}$$

which gives the number of structures per unit volume with mass in the interval $[M, M + dM]$. In their approach, they propose a simple method based on the following assumptions: the initial fluctuations are Gaussian and the overdensities collapse with spherical symmetry (the spherical-collapse model applies).

One has to find the probability that at some point the fluctuation δ_M exceeds a critical value δ_c. The probability of having a perturbation of amplitude larger than this threshold, for fluctuations smoothed on a mass scale M, is

$$P_{>\delta_c}(M) = \int_{\delta_c}^{\infty} \frac{1}{\sqrt{2\pi}\,\sigma(M)} \exp\left[-\frac{\delta^2}{2\sigma^2(M)}\right] d\delta = \mathrm{erfc}\left(\frac{\delta_c}{\sqrt{2}\,\sigma(M)}\right), \tag{1.77}$$

which gives the fraction of regions in which the density contrast exceeds the critical threshold for gravitational collapse and therefore forms bound objects of mass M. The chosen threshold value is chosen to be $\delta_c = 1.686$, which is the value extrapolated from linear theory from the value obtained in the spherical collapse model. This equation

does not account for the so-called *cloud-in-cloud problem*, namely the possibility that at a given instant some object, nonlinear on a mass scale M, can later be contained within another object on a larger mass scale. One can come up with a way around it by taking the limit of $M \to \infty$, for which the variance diverges, and the probability is expected to approach unity. This formalism, however, predicts a value of 1/2, suggesting that only half of it is locked up in collapsed structures. This is addressed by introducing an adjustment factor of 2 into the probability, which corrected the predicted value to align with expectations:

$$F(> M) = 2P(\delta_M > \delta_c). \tag{1.78}$$

While this formalism yields the halo mass function and its predictions are validated through numerical simulations, it remains merely a statistical description of structure formation. In particular, the introduction of the factor of two required to recover the correct normalisation is somewhat ad hoc, and the formalism does not, by itself, describe the assembly history of individual haloes.

This connection is made via merger trees, which trace how smaller progenitor haloes merge over time to build larger structures. In the Extended Press–Schechter theory, the original Press–Schechter formalism is generalised to compute conditional mass functions that describe the probability that a halo of a given mass at one epoch had progenitors of various masses at earlier times (Lacey & Cole, 1993).

Chapter 2

Numerical Cosmology

In Chapter 1 we saw how analytical methods can help us building intuition and formulate a theory for structure formation. Linear theory based on the Jeans instability provides only a limited description of structure formation. More advanced analytical approaches, such as Lagrangian perturbation theory, extend this treatment but still break down once shell crossing occurs. Since the growth of cosmic structures eventually enters a strongly non-linear regime where analytical perturbative methods break down, modelling these scales requires alternative approaches capable of following the full dynamical evolution.

This chapter is devoted to introducing the concept of cosmological dark-matter-only simulations, and in particular algorithms to compute the gravitational interactions within said simulations. The macroscopic collisionless evolution of non-relativistic self-gravitating classical matter in an expanding universe is governed by the cosmological Vlasov–Poisson (VP) set of equations, describing the time evolution of the phase-space distribution function $f(\boldsymbol{x}, \boldsymbol{p}, t)$:

$$\frac{\partial f}{\partial t} + \frac{\mathbf{p}}{ma^2}\nabla f - m\nabla\Phi\frac{\partial f}{\partial \boldsymbol{p}} = 0 \tag{2.1}$$

Coupled with the Poisson equation for the gravitational potential:

$$\nabla^2\Phi(\boldsymbol{x}, t) = 4\pi G a^2[\rho(\boldsymbol{x}, t) - \rho_b(t)] \tag{2.2}$$

where $\boldsymbol{x}, \boldsymbol{p}$ are, respectively, the position and the momentum, Φ is the gravitational potential and ρ_b is the background density. The proper mass density is given by the integral of the phase space distribution over the comoving momenta $\boldsymbol{p} = -ma^2\dot{\boldsymbol{x}}$:

$$\rho(\boldsymbol{x}, t) = \int f(\boldsymbol{x}, \boldsymbol{p}, t)d^3p \tag{2.3}$$

These equations describe the evolution of the system in six-dimensional phase space over time. The most known discrete approach to VP dynamics is by using a set of N particles

to trace the underlying matter distribution, and is known under the name of N-body simulations. These simulations have been employed in astrophysics since the early 1960s, here, in order to provide a historical picture of how this method became standard practice in cosmology, we follow the timeline in Angulo & Hahn (2022). N-body simulations were introduced as a numerical tool to study the Hamiltonian dynamics of gravitationally bound systems such as star and galaxy clusters (von Hoerner, 1960; Aarseth, 1963; Henon, 1973) by Monte-Carlo sampling the phase space of the system. They started being used to study cosmological structure formation in the early 1970s (Peebles, 1971; Miyoshi & Kihara, 1975). A notable example is Press & Schechter (1974), where they derived an analytical model based on statistical arguments applied to Gaussian density fluctuations, and used numerical studies to support it. This was followed by an explosion of the field in the first half of the 1980s (Doroshkevich et al., 1980; Klypin & Shandarin, 1983; Centrella & Melott, 1983; Fukushige et al., 1991; Miller, 1983). These works demonstrated the web-like structure of the distribution of matter in the Universe and established that cold dark matter reasonably provides the missing component to the mass budget of the universe. Once the dynamic range increased sufficiently Navarro et al. (1997) studied the internal structure of dark matter haloes, and proposed a model for their universal halo profiles, now known as Navarro-Frenk-White (NFW) profiles. The N-body method is now being used in all large state-of-the-art cosmological simulations as the method of choice to simulate the gravitational evolution of CDM. These simulation implement additional physics such as hydrodynamics (Vogelsberger et al. (2020) for a review) for the collisional baryonic component, and implement sub-grid models to simulate phenomena below the resolution scale of the simulation that are relevant for structure formation.

In N-body simulations the particles are sampling the underlying matter distribution, and they can be considered a Monte Carlo realisation obtained from such distribution. The particles are discrete tracers of the matter distribution rather than individual physical particles, and their mass sets the mass resolution of the simulation. Since one aims to simulate an infinite volume of universe through a finite box the approach of *periodic boundary conditions* is often used. This consist in considering infinite copies of the fundamental cubical box and setting initial conditions appropriately. The N-body method is possible because the Vlasov equation is the continuum version of the Hamiltonian equations of motion, which implies that phase-space density is conserved along Hamiltonian trajectories. The non-linear coupling in Hamilton's equations arises through the coupling of particles with gravity via Poisson's equation, which is only sourced by the density field. Therefore, as long as a finite number of N particles is able

to fairly sample the density field, the evolution of the system can be approximated by these discrete trajectories.

This chapter introduces some of the standard methods used for gravity computations in gravity solvers, so later on this knowledge can be used to interpret the analysis carried out in this work. Computing the gravitational interactions is an extremely computationally expensive problem in the context of cosmological simulations. In our simulations there will be N particles, sampling of the underlying density distribution. In a system of N particles the most straightforward way of computing the potential in a point is using the superposition principle:

$$\Phi(x) = -G \sum_j \frac{m_j}{|r - r_j|} \tag{2.4}$$

where the sum is made over all particles. Equation 2.4 represents the exact Newtonian potential which determines the particle acceleration. In the context of Simulations the gravitational potential is most commonly seen written as:

$$\Phi(x) = -G \sum_j \frac{m_j}{(|r - r_j|^2 + \epsilon^2)^{1/2}} \tag{2.5}$$

where ϵ denotes the gravitational softening length. In its absence, the denominator of Eq. 2.5 could become arbitrarily small when particles approach each other closely, causing the gravitational potential to diverge. Gravitational softening therefore regularises the force at small separations by effectively smoothing the mass distribution over a scale ϵ, which can be interpreted as giving particles a finite effective size.

This modification also suppresses artificial two-body scattering. Since N-body simulations aim to approximate the collisionless Vlasov–Poisson dynamics of dark matter, such collisional encounters are non-physical and should be minimised. However, softening also alters the gravitational force on small scales and can therefore affect the growth of structure if chosen too large. The softening length must therefore be kept well below the physical scales of interest, representing a compromise between reducing unphysical two-body interactions and preserving small-scale clustering. A detailed analysis on the impact of gravitational softening on halo properties across different cosmological simulation codes is presented in Power et al. (2003).

The most straightforward way to account for the contribution of all particles would be, at every timestep, for all N particles, to compute the contribution due to the remaining N-1 particles in the simulation. This leads to a very expensive computational setup, which scales like $\mathcal{O}(N^2)$ with the particle number, numerically executing the summation

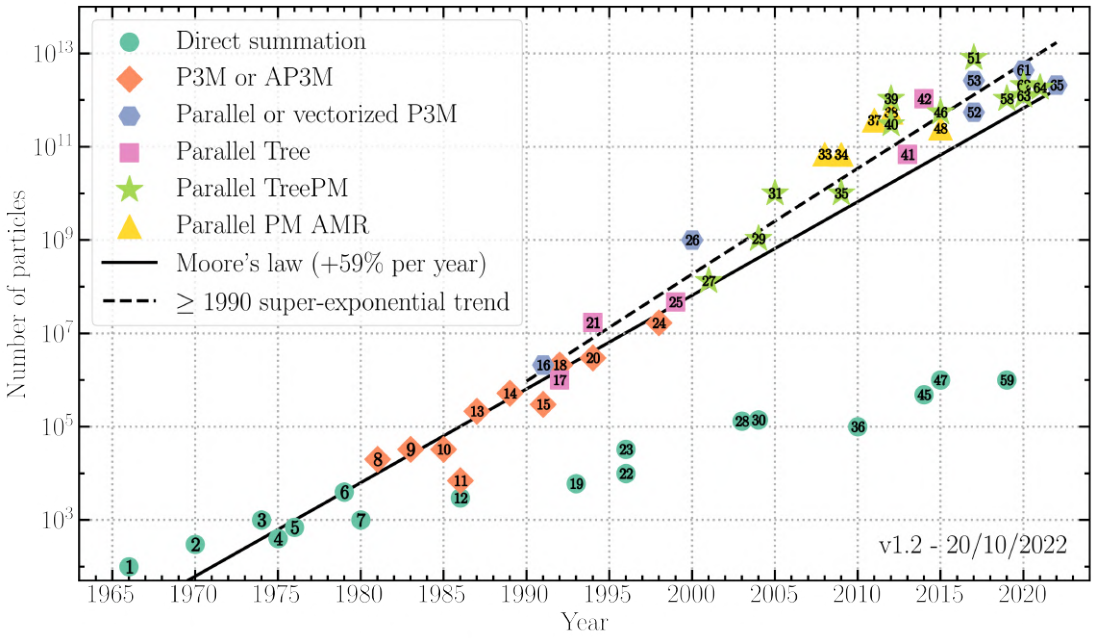


Figure 2.1. Evolution of the number of particles used in N-body simulations as a function of year of publication. The symbols and colours indicate the technique used for gravity. Hydrodynamic simulations are represented in a black square. Figure and dataset selection by Leclercq (2020).

in equation 2.5. This approach is know as *Direct Summation* or *particle-particle* method.

Over the last decades, many algorithms have been developed to make gravitational computations less expensive, while still retaining sufficient accuracy. These methods aim to reduce the number of operations required for the force computation by employing different criteria and algorithmic approaches.

Figure 2.1 illustrates how these advancements have enabled progressively larger simulations over time. Here, "larger" refers to an increasing number of tracer particles, allowing simulations to probe larger volumes at fixed resolution, achieve higher resolution within the same volume, or both. The solid black line represents Moore's law for hardware performance, which states that the number of transistors on a microprocessor—and roughly, computational performance—doubles approximately every 18–24 months (on a reprint, Moore (2006)). We can see that the size of simulations performed with direct summation does not follow Moore's law: this is understandable, considering the N^2 computational cost of calculating all pairwise gravitational forces. In the upper-right region of the figure, we see that the largest simulations ever performed employ hybrid schemes such as Parallel TreePM, P3M, or PM with AMR. Thanks to these methods, the trend of growth of simulations is super-exponential.

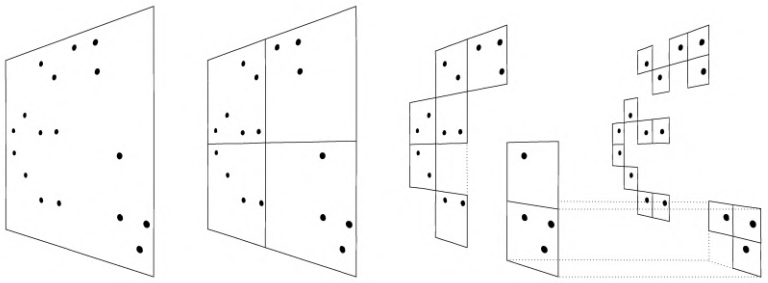


Figure 2.2. Schematic illustration of the recursive procedure to build the tree in two dimension, in this case a Barnes and Hut oct-tree (Barnes & Hut (1986)), in two dimensions. The particles are first enclosed in a square (root node). This square is then iteratively subdivided in four squares of half the size, until exactly one particle is left in each final square (leaves of the tree). In the resulting tree structure, each square can be progenitor of up to four siblings. Note that empty squares need not to be stored, from Springel et al. (2001)

In order to clarify how these computational gains are achieved, we now briefly describe the underlying principles behind these methods (see Dolag et al. (2008), Moscardini & Dolag (2011) for a review on the topic).

2.1 Tree Methods

Tree algorithms aim to reduce the number of pairwise interactions computed per timestep, in order to reduce the overall computational cost. This is achieved grouping particles far from the target and approximating their contributions to the gravitational potential through their multiple expansions, truncated to a certain order. This implies having to make a criterion for what "far" means in this context. In order to obtain the particle grouping needed for the algorithm, the simulation volume is divided recursively, this procedure is illustrated in fig. 2.2. A hierarchical tree is built so that the lowest-level nodes contain only one particle.

When the force needs to be computed, the tree needs to be walked. Starting from the root node, the code evaluates whether the multiple expansion of a node provides an accurate enough force. The opening of a lower-level node is based on an opening criterion. If the opening criterion is not met, then the force is computed with the multiple expansion of the particles in the node. Advantages of this method include:

- The dynamical range is not restricted, higher density regions end up being more resolved;

- The accuracy can be increased by setting a more constraining opening criterion.

One of the main disadvantages of this method is that it gets very computationally expensive for more homogeneous distributions, such as those at early phases of cosmological simulations. In these scenarios, in fact, many forces would cancel out due to symmetry reasons; with the tree method, however, the full tree walk has to be completed regardless of this fact. When compared to the direct summation, this method scales as NlogN.

2.2 PM methods

Particle-Mesh methods (Hockney & Eastwood, 1981) have, of course, the same aim: computing the accelerations on particles due to the gravitational potential. The path to these results, however, differs considerably. In this case we want to compute the gravitational potential by solving the Poisson equation for the gravitational potential Φ.

$$\nabla^2\Phi = 4\pi G\rho(x) \tag{2.6}$$

where $\rho(x)$ is the matter density. In real space, the gravitational potential can be written as the convolution of the density field with the Green function g:

$$\Phi(x) = \int g(x - x')\rho(x)dx \tag{2.7}$$

This convolution becomes a simple multiplication in the Fourier space, and the PM method was designed to make use of this property to reduce the computational cost.

$$\hat{\Phi}(k) = \hat{g}(k) \cdot \hat{\rho}(k) \tag{2.8}$$

This is possible also because Fast-Fourier-Transform (FFT) libraries provide an efficient way to compute the Fourier transform of our density field.

The Green function employed for periodic boundary condition simulations is either the continuum Green function in eq. 2.9 (known as the "Poor man's Poisson Solver" Bagla & Padmanabhan (1997)) or by the "Periodic Green's Function".

$$g(k) = -\frac{1}{k^2} \tag{2.9}$$

The former has the advantage of being the true Green's function, the latter is of easier implementation, since it comes from the Fourier transform of a discrete realisation of the Laplacian.

The main steps of the PM method are three:

- Computing the Fourier transform of the density field
- Multiplying the density field in Fourier space by the Green's function
- Computing the inverse Fourier transform of the potential to real space

However, since we are simulating a system of N discrete particles, this is not so straightforward, as we do not have a density field in the first place. To these steps, we then need to add a step 0: computing the density field, by assigning particles to cells of a grid.

Moreover, after having obtained the potential, it has to be differentiated to get the force, and then the force needs to be assigned back to the particles, following the same criterion with which the particles were assigned to the grid in the first place.

A schematic representation of this procedure is illustrated in figure 2.3

To assign the particles to grid cells we need to decide which fraction of the particle's mass needs to be assigned to the adjacent cells. We could assign the entire mass of every particle to the grid cells where the point is. This would be straightforward, however it would lead to a force field piecewise constant in cells, but discontinuous at cells edges. For this reason the standard practice is to associate a shape function to the particles. The shape function is constant within a defined volume, and zero elsewhere. The weight with which a particle in x_p contributes to a grid cell with centre x_{ijk} can be written as:

$$W(x_p - x_{ijk}) = \int_{x_{ijk}-\Delta x/2}^{x_{ijk}+\Delta x/2} S(x_p - x')\, dx' \tag{2.10}$$

where $S(x_p - x')$ is the shape function. So that the density can be written as:

$$\rho_{ijk} = \sum_{p=1}^{N_p} m_p\, W(\mathbf{x}_p - \mathbf{x}_{ijk}) \tag{2.11}$$

In order of increasing accuracy, in terms of continuity of the resulting force field, the most commonly used interpolation functions are:

- *Nearest grid point* (NGP): The shape function is Dirac delta. In this case we are back to assigning the entire mass of the particle to the grid cell it belongs to.:

$$S(x) = \frac{1}{\Delta x}\,\delta\!\left(\frac{x}{\Delta x}\right) \tag{2.12}$$

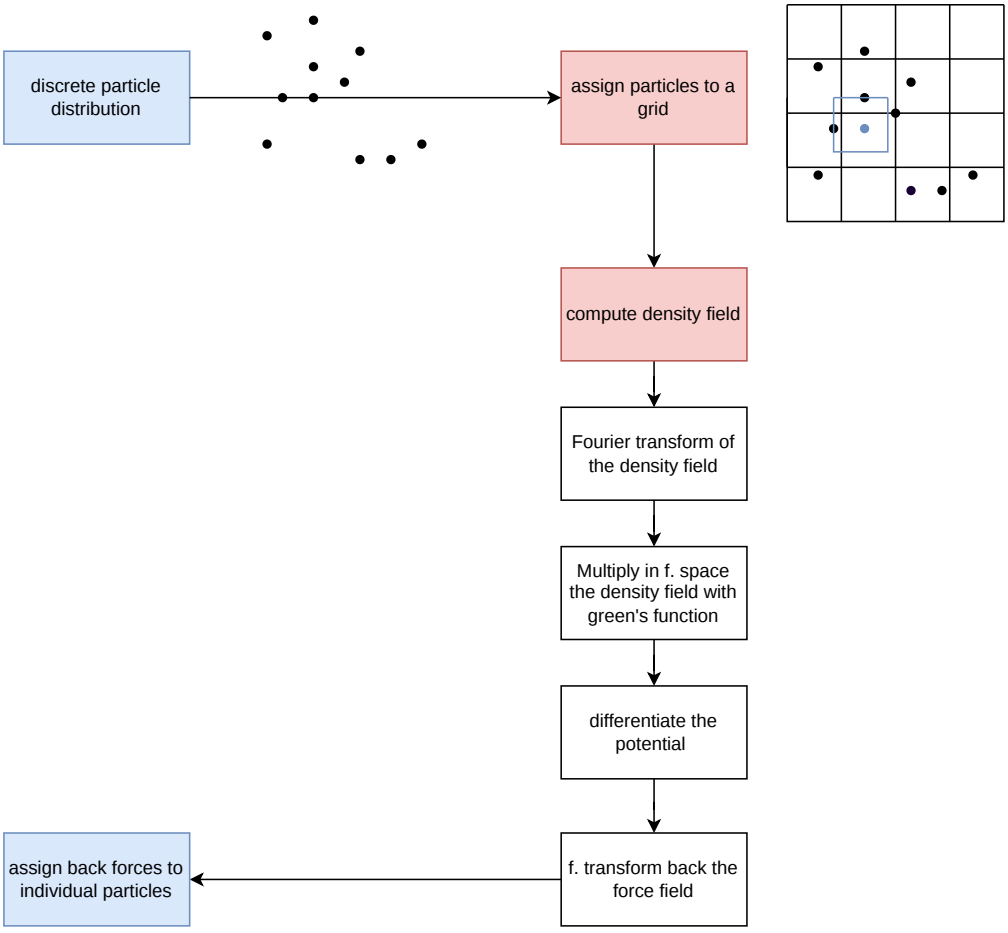


Figure 2.3. Schematic representation of the particle-mesh method. The blue items represent the starting and ending points of the procedure.

- *Cloud in cells* (CIC): the particle's mass is distributed uniformly over a cube of side length h. This cubic volume spans over 8 adjacent cells, and leads to a continuous force field. This is the method illustrated in Fig. 2.3, where it is illustrated in two dimensions.

$$S(x) = \frac{1}{\Delta x} \begin{cases} 1 & \text{if } |x| < 0.5\,\Delta x \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

- *Triangular-shaped-cell* (TSC): the mass decomposition involves three (i.e. $3^3 = 27$ in the 3D case) nearest points. This choice of interpolating function leads to a force field with continuous first derivative.

$$S(x) = \frac{1}{\Delta x} \begin{cases} 1 - \dfrac{|x|}{\Delta x} & \text{if } |x| < \Delta x \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

The interpolating functions could be extended to higher orders, this is rarely done due to the increase in computational cost.

The same weights used in this step to assign particles to the PM-grid will also be used to assign the forces back to the particles.

To obtain the force, the potential can be differentiated in the Fourier Space, and then the components of the force can be Fourier transformed back to real space.

Some of the main disadvantages of the Particle Mesh methods include the fact that the dynamical range is set by the number of grid cells. In our scenarios, when structures start to form many particles will be within one grid cell, so we completely lose the resolution at small scales. Moreover at every timestep the force of all particles must be recomputed, and therefore no individual time steps are allowed. On the other hand, for tree methods some hierarchical time integration techniques can be implemented, as it will be further explained in Sect. 4.3. One of the main advantages is that this method has no downsides when working with homogeneous distributions, and scales as $N_p + N_g log N_g$, with N_p being the usual number of particles and N_g the number of grid cells.

Adaptive Mesh Refinement (AMR) methods were developed to face the aforementioned issues of PM methods. The idea behind AMR is to redefine the mesh where density exceeds a threshold, and to solve Poisson's equation on the adaptive mesh hierarchy, so that the resolution is no longer limited to the size of the original PM grid (for a review see Norman (2004)). Many cosmological simulation codes employ AMR for cosmological and hydrodinamical simulations, such as ENZO (Bryan et al., 2014) and RAMSES (Teyssier, 2002).

2.3 Hybrid methods

The two previously mentioned methods have opposite strengths, for this reason, many modern cosmological codes employ hybrid methods. Most modern codes, as also seen in Fig. 2.1, employ a hybrid method known as TreePM. This method was introduced in Gadget2 (Springel, 2005) and it is meant to allow to run bigger simulations by splitting the force computation between two regimes.

When computing gravitational interaction the gravitational potential is split into two terms in the Fourier space:

$$\Phi_k = \Phi_k^{short} + \Phi_k^{long} \tag{2.15}$$

where the short range term is computed using the tree.

$$\Phi_k^{short} = \Phi_k[1 - e^{(-k^2 r_{cut}^2)}] \tag{2.16}$$

And the long range contribution with the PM grid:

$$\Phi_k^{long} = \Phi_k e^{(-k^2 r_{cut}^2)} \tag{2.17}$$

where r_{cut} is the scale of the force split, and should be chosen to minimise force errors (Springel, 2005).

Before the introduction of the TreePM method in the GADGET family of codes, there had been earlier attempts to combine the efficiency of Particle-Mesh (PM) methods with better dynamical range. The first such attempt was the so-called P^{3}M code (Hockney & Eastwood, 1981), which combines the precision of direct summation (particle–particle) methods with the efficiency of PM grids to account for long-range forces. An improved version of this approach incorporates adaptive mesh refinement (AMR) and is known as AP^{3}M (Kravtsov et al., 1997).

2.4 GRAPE: a Hardware-Based Approach to the N^2 Bottleneck

As part of this historical overview, it is worth mentioning an interesting approach to overcome the $O(N^2)$ bottleneck of gravitational force calculations using specialised hardware. Introduced in 1990 (Makino et al., 1990), GRAPE (GRAvity PipE) is a special-purpose computer designed for N-body simulations. The first implementation of GADGET (Springel et al., 2001) featured the option to run direct summation on GRAPE hardware.

With the advent of programmable GPUs, many of the performance gains originally achieved by GRAPE hardware can now be realised using commodity graphics processors. For example Zwart et al. (2007) demonstrated that GPUs can rival GRAPE-6 performance for direct N-body force calculations in terms of runtime and speedup.

Chapter 3

Parallel computing for scientific applications

In the previous section we presented some of the methods used in cosmological simulations, and concluded with a discussion of specialised hardware such as GRAPE (GRAvity PipE), which was designed to accelerate gravitational N-body calculations. Modern Graphics Processing Units (GPUs) can be used to implement acceleration on a general purpose hardware that today is already available in most of the top computing centres. According to Top500 (2025a) " A total of 255 systems on the list are using accelerator/co-processor technology". Table 3.1 shows the top 10 most common GPU accelerators used in HPC systems. Before the advent of GPUs, cosmological simulations were designed to run in parallel on CPUs due to the high computational cost of evaluating gravitational interactions and the large memory requirements of the simulations.

Rank	Accelerator	Count	Share (%)	Rmax (TFlops)	Rpeak (TFlops)
1	NVIDIA H100 SXM5 80GB	26	5.2	855,772	1,297,807
2	NVIDIA A100	22	4.4	302,864	460,443
3	NVIDIA H100	17	3.4	939,587	1,428,714
4	NVIDIA GH200 Superchip	17	3.4	2,268,950	2,967,609
5	NVIDIA H200 SXM5 141GB	16	3.2	630,283	909,833
6	NVIDIA A100 SXM4 40GB	15	3.0	178,250	250,224
7	AMD Instinct MI250X	14	2.8	2,402,071	3,459,745
8	NVIDIA H100 SXM5 94GB	14	2.8	219,048	345,378
9	NVIDIA Tesla V100	14	2.8	114,492	166,870
10	NVIDIA H100 80GB	12	2.4	288,283	419,860

Table 3.1. Most common GPU accelerators in leading HPC systems, including system count, share, and performance metrics (Rmax and Rpeak) (Top500, 2025b).

Parallel computing addresses this challenge by decomposing the problem into smaller tasks that can be executed concurrently on multiple processing units, while also providing more memory for the simulations across multiple *nodes*.

The most intuitive form of parallelism is Single Instruction, Multiple Data (SIMD),

where the same operation is applied simultaneously to different elements of a dataset. SIMD is particularly effective for embarrassingly parallel problems, in which tasks are independent and require no communication, such as Monte Carlo realisations. Modern CPUs also allow *thread-based parallelism*, where a single core can execute multiple threads concurrently, overlapping instruction execution to improve utilisation of its resources.

In large simulations, however, particles may be distributed across different nodes of a computing cluster, introducing challenges related to data locality, due to the long-range nature of gravitational interactions. As a result, modern simulation codes must combine multiple forms of parallelism, carefully balancing computation, communication, and memory distribution to achieve both efficiency and scalability.

In this chapter we introduce the fundamental concepts of parallel computing on CPU and GPU together with examining different programming paradigms to exploit this hardware.

3.1 CPUs vs GPUs: Differences at hardware level

Central processing units (CPUs) and graphics processing units (GPUs) represent two complementary design points in modern high-performance computing. CPUs are optimised for low-latency execution on a wide variety of tasks, with a small number of complex cores and sophisticated control logic that excel on sequential and branch-heavy workloads. GPUs, in contrast, follow a throughput-oriented design: they integrate hundreds to thousands of simpler cores and very high memory bandwidth, which allows them to execute the same operation on massive datasets in parallel. This architectural difference translates into different capabilities and limitations of the two processor types. From a performance perspective, GPUs can deliver an order-of-magnitude speed-up for highly data-parallel kernels (see Appendix A). This is particularly true on embarrassingly parallel problems where the algorithm is optimised for GPU use.

Although a detailed discussion of hardware is beyond the scope of this work, it is important to note that the characteristics of GPU programming are closely tied to their architecture. This paragraph, therefore, briefly introduces these features and contrasts them with those of CPUs. Figure 3.1 shows the difference between a CPU and GPU architecture: in GPUs more transistors are dedicated to data processing, while on CPU many are dedicated to control flow and memory management.

GPUs consist of hundreds or thousands of simpler cores organised into *streaming multiprocessors* (SMs, also called compute units), which execute many threads in parallel

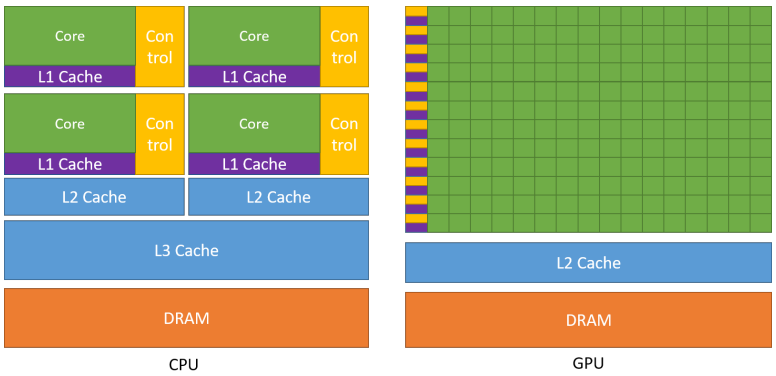


Figure 3.1. CPU and GPU cores. Elements highlighted in the same colour serve the same purpose, the GPU devotes more cores to data processing (from Nvidia (2025))

using a SIMD model. In the following, the terminology will be mostly NVIDIA-specific. There is no universal nomenclature across vendors; for example, AMD refers to these units as *Compute Units (CUs)*, while Intel uses the term *Execution Units (EUs)*.

On the other hand a CPU is optimised for low-latency execution of sequential and control-intensive tasks. It typically contains a small number of powerful cores, each with sophisticated control logic, branch prediction, and out-of-order execution. CPUs rely heavily on a hierarchical cache system (L1, L2, and often shared L3 cache) to reduce memory access latency, with L1 requiring the least clock cycles to be accessed, and the main memory (DRAM) the most. This design makes CPUs well suited for workloads with irregular memory access patterns and frequent branching. These differences in an heterogeneous CPU/GPU architecture are illustrated in Fig. 3.2, providing a system diagram for the architecture. This scheme also highlights that the two do not share the same global memory (physically) and in terms of variable space, therefore it is responsibility of the programmer to allocate memory on the GPU and transfer data.

It is also worth examining the memory organisation of a GPU, shown in Fig. 3.2. A GPU typically consists of off-chip global memory, an L2 cache, and a unified data cache for each streaming multiprocessor, which is shared between the L1 cache and shared memory.

The *Global memory* is the main memory of the GPU. It is physically located off-chip. All threads can access global memory. It provides very high bandwidth, meaning it can transfer a large amount of data per second. However, the latency of a single access is relatively high, often hundreds of clock cycles. This means that if a thread reads data from global memory, it may need to wait many cycles before the data becomes available. Because of this high latency, GPU performance depends heavily on accessing global

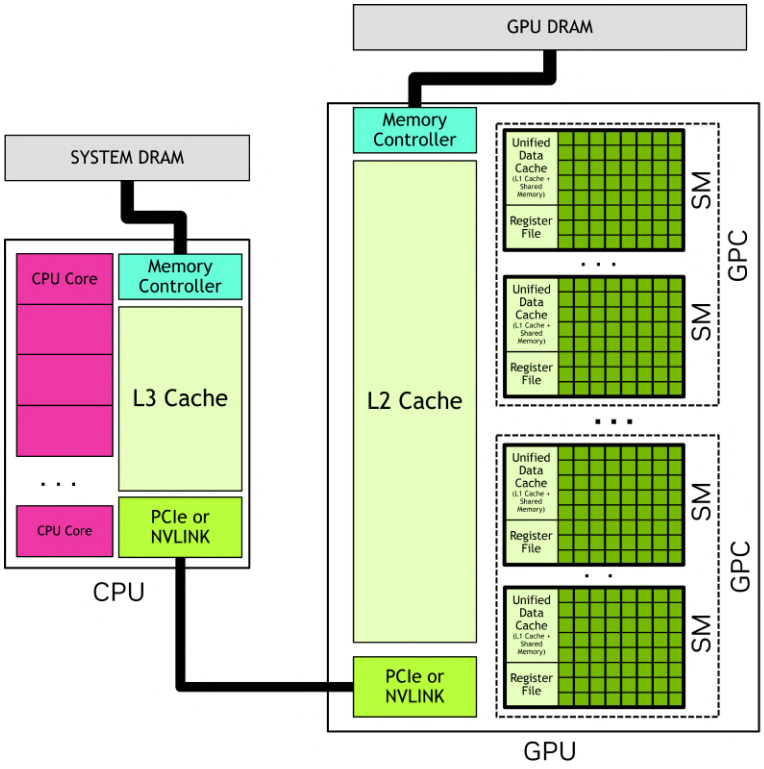


Figure 3.2. CPU and GPU cores. Elements highlited in the same colour serve the same purpose, the GPU devotes more cores to data processing (from Nvidia (2025)).

memory efficiently. In particular, when consecutive threads access consecutive memory addresses the hardware can combine these requests into fewer memory transactions, improving effective bandwidth usage. This access pattern is known as *coalesced access*. *Shared memory*, in contrast, is an on-chip memory located within each SM. It is much smaller than global memory but significantly faster, with latency comparable to cache access on a CPU. More details will be discussed later when introducing the programming paradigms that govern how shared memory is accessed.

3.2 Parallelism on CPU

This section is dedicated to the description of parallel programming on CPU. From the generic picture of CPU architecture portrayed in the previous section we now introduce the concept of High Performance Computing. According to IBM (2025) "HPC is a technology that uses clusters of powerful processors that work in parallel to process massive, multidimensional data sets and solve complex problems at extremely high

speeds". An HPC cluster comprises multiple high-speed computer servers networked with a centralised scheduler that manages the parallel computing workload. The computers, called nodes, use high-performance multi-core CPUs (and today also GPUs). A single HPC cluster can include 100,000 or more nodes.

Since computing clusters are such complex systems, with different nodes connected through a network, one has to keep in mind that not all cores will share the same variable space. This setup is referred to as *distributed memory system*. In this section we will discuss programming standards to enable parallelism on CPU level both at shared and distributed memory level.

3.2.1 Shared memory parallelism: OpenMP

On multi-core CPU systems, all cores within a single node can directly access the same memory space. This is referred to as a shared memory environment. To exploit this architecture efficiently, programmers can use OpenMP (OpenMP, 2026), a widely adopted standard for shared memory parallelism. OpenMP uses compiler directives to annotate existing code, allowing threads to execute regions of code in parallel. One of the challenges of shared memory parallelisation is ensuring a correct access pattern to memory. This can be achieved using *barriers* or *synchronisation points*. Some of the key constructs in openMP are:

- Parallel regions: Sections of code executed concurrently by multiple threads (#pragma omp parallel).
- Parallel loops: Loops divided among threads (#pragma omp for) to perform the same operation on different data elements.
- Vectorisation: Loops divided among threads (#pragma omp for simd) explicitly requiring vectorisation, given a correct data layout provides speedup even on a single OMP thread.
- Barriers: #pragma omp barrier ensures all threads reach the same point before continuing.
- Critical sections: #pragma omp critical allows one thread at a time to execute a block of code.
- Atomic operations: #pragma omp atomic guarantees atomic read-modify-write for shared variables.

- Reductions: Aggregate values (addition, max, min) safely across threads using reduction clauses.

Some of these aspects will be further discussed in the Sect. 3.3 when discussing GPU parallelisation.

Even on single nodes, some disadvantages of using shared memory can arise, this is particularly true for NUMA (non uniform memory architecture) where the access time to memory is uneven for different cores, and data locality becomes a relevant topic.

3.2.2 Parallelisation on distributed memory: MPI

When scaling beyond a single node, each node maintains its own private memory. This requires explicit communication between nodes, which is handled through the Message Passing Interface (MPI, The Open MPI Community (2026)), the standard framework for distributed-memory parallel programming in HPC. In MPI programs, the computational work is divided among different *MPI ranks*, each of which can be assigned a specific role. For example, one rank (often rank 0) may act as a coordinator or handle serialised I/O operations.

An important aspect of MPI programming is *workload management*. Ensuring that tasks are distributed evenly across ranks helps prevent processors from remaining idle and wasting computing resources. However, balancing the workload typically requires more frequent communication and synchronisation between nodes, which introduces communication overhead. For this reason, an efficient MPI implementation must carefully balance load distribution against the cost of communication.

Key MPI features include:

- Point-to-point communication: MPI_Send / MPI _Recv for transferring data between two processes. Non-blocking variants (MPI_Isend, MPI_Irecv) allow overlapping communication and computation.

- Collective communication: Operations involving all processes in a group such us MPI_Bcast – broadcast a value to all processes, MPI_Reduce / MPI_Allreduce – compute global reductions, MPI_Scatter / MPI_Gather – distribute or collect data across processes

- Synchronisation: MPI_Barrier ensures all processes reach the same point before proceeding. Non-blocking operations require MPI_Wait or MPI_Test to ensure completion.

3.3 Parallelism on GPU, Programming Models and Portability Challenges

GPU programming models are closely related to the chip architecture discussed in Sec. 3.1. At present, there is no universal industry standard for GPU programming, which represents one of the main challenges in developing portable code. In this section we discuss several programming models, starting with low-level approaches such as CUDA, which provide a clearer view of the relationship between hardware and software architecture, and then moving to directive-based models such as OpenMP and OpenACC.

3.3.1 CUDA

In this section we provide an overview of the CUDA language and API. A more detailed description can be found in the NVIDIA CUDA Programming Guide (Nvidia, 2025). While this work focuses on CUDA, as it provides a clear link between GPU hardware and the programming model, it is important to note that several alternative frameworks also exist. HiP (Heterogeneous-computing Interface for Portability, AMD (2026a)) is an API developed by AMD that is meant for the same uses as CUDA, but works on both AMD and Nvidia architectures. The HiP documentation also provides useful tools to port CUDA code to HiP (AMD, 2026b) and to compare functions across the two APIs (AMD, n.d.). Other frameworks aimed at heterogeneous computing, such as SYCL (The Khronos Group, 2026) and Kokkos (Contributors to the Kokkos project, 2026), attempt to address cross-vendor portability by providing higher-level abstractions that can target multiple hardware backends. These approaches sit conceptually between low-level APIs such as CUDA/HiP and directive-based approaches like those discussed in Sect. 3.3.2. Since the goal here is to illustrate the fundamental concepts of GPU programming and their connection to hardware architecture, CUDA provides a convenient and widely used reference model, and the discussion will therefore focus on this framework. CUDA supports programming on heterogeneous architectures, and assumes that the system includes both CPUs and GPUs.

In CUDA terminology, the CPU is referred to as *host* and its memory as *host memory* while the GPU is referred as *device* and its memory as *device memory*. An application executing on the GPU is referred to as *device code*, and a function that is invoked for execution on the GPU is, for historical reasons, called a *kernel*. Kernels can be *launched*, this is the technical term for starting (or running) the kernel. In CUDA an execution hierarchy, illustrated in Fig. 3.3, is defined to help structure large parallel

computations. At the lowest level are *threads*, which represent the smallest units of parallel work. Threads are grouped into *thread blocks*, which are collections of threads that can cooperate and synchronise with each other using shared memory. Thread

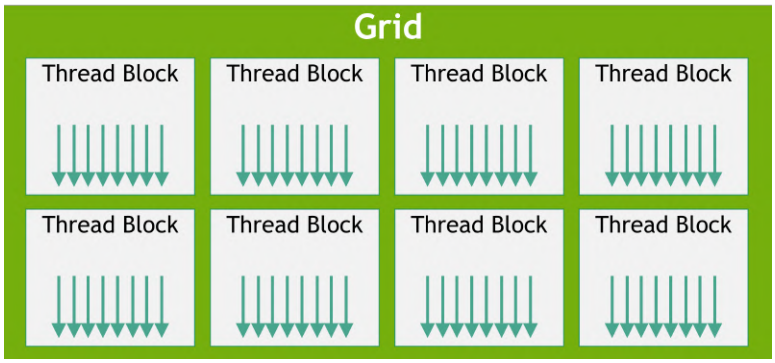


Figure 3.3. CUDA execution hierarchy, threads in blocks, launched in a grid (Nvidia, 2025)

blocks are lunched in a *grid*. When a kernel is launched, the programmer specifies both the *block size* and *grid dimensions*, allowing a large grid of thread blocks to be dispatched across the device. Thread blocks and grids may have up to three dimensions, which simplifies the mapping of individual threads to units of work or data items. For example, when operating on a matrix, threads can be naturally mapped to its elements. Threads are also aware of their index within the containing block, as well as the position of the block within the grid. This information is typically used to determine which portion of the dataset a thread will process.

Within a thread block, threads are further scheduled by the hardware into *warps*, fixed-size groups of threads (typically 32) that execute the same instruction in lockstep under the Single-Instruction Multiple Threads (SIMT) model. The CUDA runtime maps each thread block to a single SM for (usually) its lifetime, ensuring that all threads within the block have access to the same on-chip shared memory and can synchronise efficiently. All threads in the warp execute the same instruction simultaneously. If some threads within a warp follow a control flow branch in execution while others do not, the threads which do not follow the branch will be masked off while the threads which follow the branch are executed. This organisation in threads, warps, blocks is a software abstraction, however it is based on the underlying hardware execution units.

Moving into memory access patterns, we have seen that accessing the global memory has higher latency. In modern GPU architectures, global memory accesses are performed at the granularity of a warp rather than an individual thread. The memory subsystem collects addresses and groups them into one or more *memory transactions*. Instead of

fetching data for each thread independently, the hardware attempts to combine accesses that fall within the same aligned memory segment. In order to use this feature for increased computational performance, the data layout is fundamental. The addresses requested by the threads must be contiguous. Another key consideration in shared memory is *bank conflicts*. If multiple threads in a warp access different addresses within the same bank simultaneously, the accesses are *serialised*, reducing effective performance, this is another reason for which it is important to avoid randomised access pattern to memory.

Finally, it is worth mentioning the concept of *race conditions*. When multiple threads write to the same memory location simultaneously, a *race condition* can occur, producing non-deterministic results. To avoid them, CUDA provides *atomic operations*, which guarantee that a read-modify-write sequence is executed as a single, indivisible step. Atomics are also widely used in *parallel reductions*, where threads collaboratively compute sums, minima, maxima, or other operations over elements of an array. Table 3.2 highlights some of the features of GPU programming and the challenges and advantages they bring to software development.

3.3.2 Directive Based Approaches

As we have seen, GPU programming comes with high complexity and often requires completely restructuring the existing codebase that one aims to offload to the GPU. Directive-based programming models are designed to hide some of these complexities from the developer, allowing to accelerate code for GPUs using compiler directives into C, C++ or Fortran code. These directives act as annotations that instruct the compiler where and how to offload computation and data movement to an accelerator without requiring the programmer to explicitly write device kernels or manage low-level thread organisation. On the other hand, they provide less flexibility and control, and often do not allow to achieve the same speed-up. Even in this case, there is no industry standard, the two most used directive based approaches are OpenACC and OpenMP offloading, designed respectively for Nvidia and Intel architectures.

- *OpenACC* (Nvidia, 2026) is a high-level directive standard developed for Nvidia hardware. It focuses on simplicity and minimal code changes. Parallelism is achieved using pragmas such as `#pragma acc parallel`, `#pragma acc kernels`, and `#pragma acc data` to define regions of code to be executed on an accelerator and to describe data movement between the host and device. The compiler then generates the appropriate parallelised GPU code.

Feature	Impact on Software Development
Many threads at low cost	Enables efficient execution of SIMD/SIMT workloads by leveraging massive parallelism.
Warps execute in lockstep	Thread divergence should be minimised or managed, since divergent paths reduce hardware utilisation.
High-latency global memory	Minimise global memory transactions.
Coalesced memory access	Proper data layout is critical to achieve maximum memory bandwidth and efficiency.
Shared memory	Use shared memory to store frequently reused data and reduce global memory access.
Race conditions	Occur when multiple threads write to the same memory location simultaneously. Use atomic operations or algorithm redesign to prevent non-deterministic results.
Shared memory bank conflicts	If multiple threads in a warp access different addresses in the same bank simultaneously, accesses are serialised, reducing performance. Adjust memory layout and indexing to distribute accesses across banks.
Atomic operations	Ensure read-modify-write operations on memory are executed atomically, preventing race conditions. Used in parallel reductions, their use should be cointained as they hinder performance
Parallel reductions	Combine data from multiple threads (e.g., sum, min, max) using shared memory and atomics. Proper tree-based reduction patterns minimise serialisation and maximise performance.

Table 3.2. Key GPU architectural features and their impact on software development.

- *OpenMP target offloading* Intel (2021) extends the established OpenMP standard for CPU previously discussed to heterogeneous systems, allowing `#pragma omp target` and related constructs (e.g., `teams`, `distribute`, `parallel`) to specify loops or functions that should be compiled for and executed on a GPU. While developed for intel hardware, according to the developers it provides compatibility with other vendors, even if performance on cross vendor architecture is still being studied.

Figure 3.4 provides a summary of common openACC pragmas and their openMP counterpart, together with common use scenarios.

Directive based approach improves portability, although performance tuning may still require attention to data transfers and parallelisation granularity.

OpenACC Pragma	OpenMP Pragma	Intent
`#pragma acc parallel`	`#pragma omp target teams`	Create a team of threads on the GPU, with the master thread in each team executing the region
`#pragma acc parallel loop gang worker vector collapse(2)`	`#pragma omp target teams distribute parallel for simd collapse(2)`	Parallel computation, distribute work across GPU hardware threads
`#pragma acc kernels loop reduction(+:norm)`	`#pragma omp parallel for reduction(+:norm)`	Parallel reduction computation
`#pragma acc data copy(A[0:Sz])`	`#pragma omp target data map(tofrom: A[0:Sz])`	Copy data to/from the target device
`#pragma acc update host(A[0:Sz])`	`#pragma omp target update from(A[0:Sz])`	Update data on the host from the device
`#pragma acc data copyin(A[0:Sz])`	`#pragma omp target data map(alloc:A[0:Sz])`	Allocate memory on the device
`#pragma acc update device(X)`	`#pragma omp target update to(X[0:Sz])`	Update data on the device from the host
`#pragma acc loop vector`	`#pragma omp simd`	Vectorization
`#pragma acc atomic update`	`#pragma omp atomic update`	Atomically update a memory location

Figure 3.4. Common OpenACC pragmas and their OpenMP equivalents, from Intel (2021)

II

OpenGadget: from CPU to GPU

Chapter 4

The Cosmological Simulatation Code OpenGadget3

This chapter presents the cosmological simulation code OpenGadget3 (Ragagnin et al. (2020), Dolag et al. (n.d.) in preparation), used throughout this thesis, with a focus on the gravity solver and its parallelisation. The discussion builds on the concepts introduced in Chapters 2 and 3.

While the general structure of the solver has been described in the literature, in this chapter, we provide a novel, detailed mapping of the control flow under different compilation and runtime configurations, obtained directly from the source code. Sections 4.4 and 4.5 present this new analysis, highlighting how the code behaves in GPU-enabled versus CPU-only runs. This original contribution is critical for understanding the numerical behaviour observed in the validation tests of the subsequent chapters.

OpenGadget3 is an improved version of (P-Gadget3-XXL), an extension of (Springel et al., 2001; Springel, 2005), featuring several modules for additional physics, such as Cooling, Star Formation, Stellar Feedback, Chemical Enrichment, Black Hole physics and more recently an implementation for self-interacting dark matter (Fischer et al., 2026). This thesis deals with dark-matter-only simulations assuming CDM, this chapter describes the simulation workflow, emphasising the stages of the algorithm that are critical for performance. A simplified flux diagram for the code can be seen in Fig. 4.1. In this chapter, we will discuss the initial conditions for cosmological simulations and then analyse the gravity solver and the time integration. Some time will also be spent on discussing the halo finder algorithm , since it is heavily used for the analysis of our simulations discussed in Chapters 5 and 6.

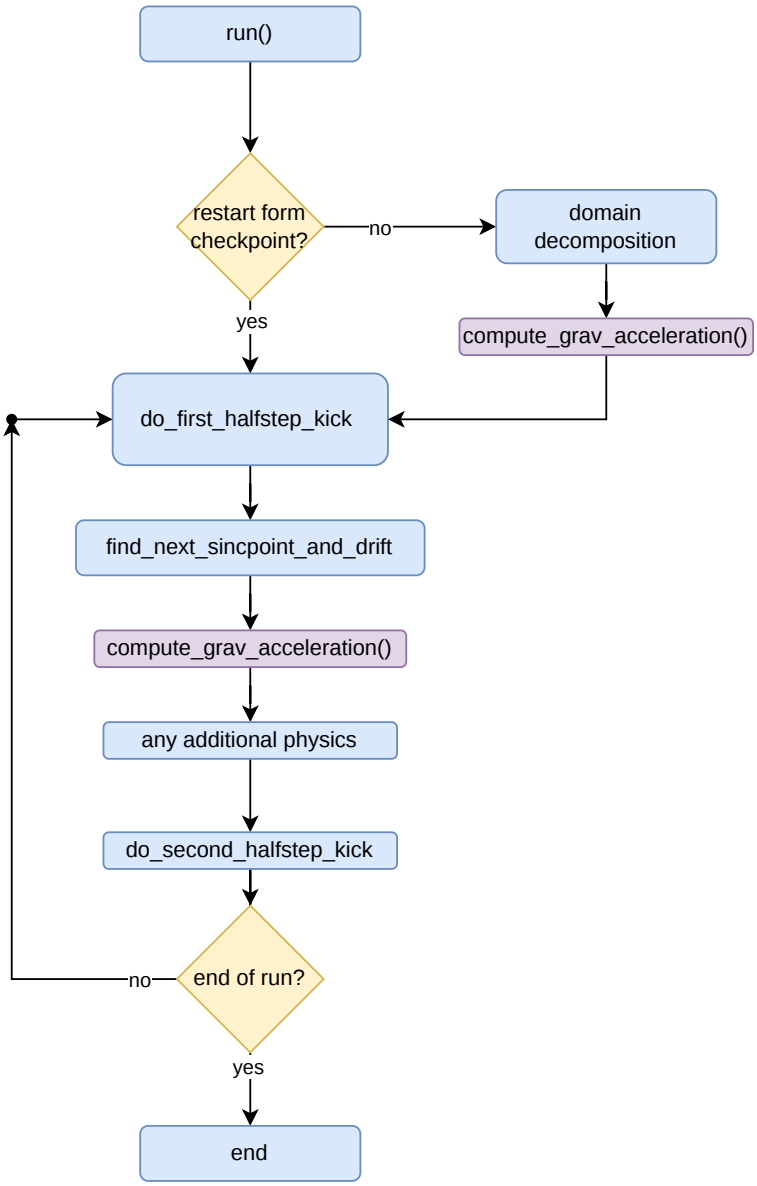


Figure 4.1. Simplified schematic of a dark-matter-only simulation run, emphasising the stages relevant to this thesis. Gravitational acceleration computations are highlighted in purple.

4.1 Cosmological Boxes

A cosmological box is a simulation of a finite volume of the universe, achieved by simulating a cube with periodic boundary conditions. The initial conditions for the simulation are generated at high redshift using linear perturbation theory. First, a Gaussian random density field consistent with the matter power spectrum is sampled, and particle displacements are then computed using linear perturbation theory. The modes that such simulations can represent are constrained on large scales by the size of the box, and on small scales by the resolution of the simulation. In a dark-matter-only simulation using LCDM as a cosmological framework, the simulation evolves under the effect of the collisionless Boltzmann equation for DM, the Poisson equation for the gravitational potential, and the Friedmann equations for the expansion of the universe.

4.2 Parallelism and Domain Decomposition

Cosmological simulation codes are massively parallel and, in their usual application, run on several cores or, more commonly, on multiple nodes of computing clusters.

In Chapter 2 we have portrayed a picture in which every particle in the simulation has to "know" the effect of every other particle on the gravitational field, regardless of the method used to compute the potential. Due to the way data structures are implemented in Gadget (arrays of large structures), the code does not benefit from vectorisation. Easy parallelisation with minimal algorithmic changes can be achieved using compiler directives, as described in Sect. (3.2.1). This approach, however, has two limitations: first, depending on the data access pattern and the compiler used, it might not produce optimal results; second, and more importantly, it works only if all cores share the same memory, i.e., in a shared memory architecture. This is never the case if we are working across multiple nodes of a computational cluster, and even on shared memory architectures, it can backfire if the system is NUMA (Non-Uniform Memory Access).

This is why the main parallelisation in Gadget is achieved with MPI (Sect.3.2.2). Originally, Gadget2 came with CPU-only MPI parallelisation. In this approach, memory is distributed, and communication among MPI processes is explicitly handled by the developer.

Simulations of this size are typically memory-bound, making domain decomposition a fundamental part of the solver. In Gadget, the domain decomposition is performed to ensure that tree-force calculations (Sec. 2.1) are independent of the number of processors. This is achieved by mapping the 3D simulation volume onto a 1D Peano–Hilbert curve,

which is then split into M segments containing an equal number of particles, where M is the number of MPI ranks. Details on the construction of these curves can be found in the original Gadget-2 paper (Springel, 2005).

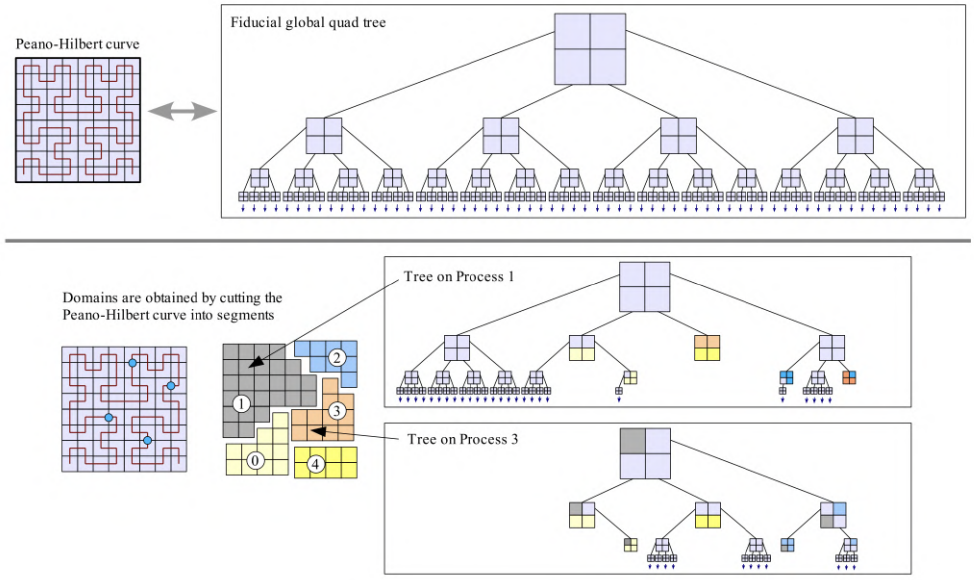


Figure 4.2. Illustration of the relation between the BH oct-tree and a domain decomposition based on a Peano–Hilbert curve. For clarity, the sketch is drawn in two dimensions. The fiducial Peano curve associated with the simulation volume visits each cell of a regular mesh exactly once. The simulation volume is cut into domains by segmenting this curve at arbitrary intermediate points on cell boundaries. This generates a rule for distributing the particle set onto individual processors. Because the geometric structure of the BH tree is commensurable with the mesh, each mesh cell corresponds to a certain branch of a fiducial global BH tree. These branches then reside entirely on single processors. In addition, each processor constructs a 'top-level tree' where all nodes at a higher level are represented. The missing data on other processors is marked using 'pseudo-particles' in this tree, from Springel (2005).

The interesting property that makes this method suitable for domain decomposition is that points close on the curve are also close in 3D space. Such domain decomposition preserves, therefore, locality, a fundamental property to minimise communication costs among processors handling neighbouring domains. As illustrated by figure 4.2, this choice of domain decomposition also aligns with the construction of the gravity tree: in the solver different sections of the domain belong to different MPI ranks, each processor has a branch of the tree and constructs a *top-level tree* where all nodes at higher level are represented. In these nodes the missing data on other processors is marked using *pseudo-particles*. We will see in Sect. 4.4 where the link between the tree algorithm and

the domain decomposition becomes relevant.

4.3 The gravity solver in OpenGadget3

The gravity solver implemented in OpenGadget3 is a TreePM algorithm (Sec. 2.3). This section builds on the description of treePM algorithms given in Sect. 2.3, focusing on implementation details specific to OpenGadget3. A more detailed description can be found in Springel (2005).

For the short-range force, the algorithm uses a Barnes–Hut oct-tree (Barnes & Hut, 1986), in which nodes are constructed by recursively subdividing the computational domain into a hierarchy of cubic cells. As in the original implementation in Gadget2, two different opening criteria can be selected. The first criterion is purely geometrical

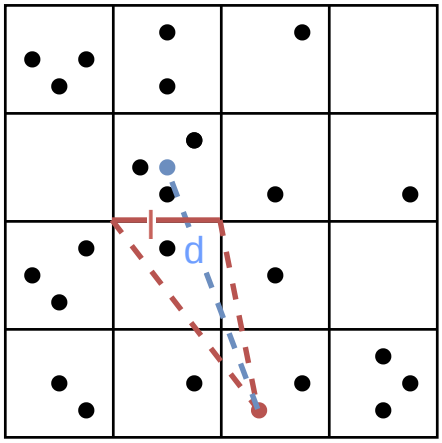


Figure 4.3. 2D Illustration of the geometrical opening criterion, d is the distance of the target particle from the centre of mass of the node, l the size of the node

and is known as the *opening angle criterion*. A node is opened if the angle it subtends at the target particle, θ_p, exceeds the chosen opening angle θ. Figure 4.3 illustrates this logic, where d is the distance between the target particle and the centre of mass of the node, and l is the size of the node. The subtended angle can be approximated as:

$$\theta_p \simeq \frac{l}{d}\,, \tag{4.1}$$

Otherwise, the gravitational contribution of all particles within the node is computed using their multipole expansion. Typical values adopted in the literature for the opening angle are around $\theta \simeq 0.45$, representing a compromise between force accuracy and computational cost.

The alternative (and currently default) opening criterion in OpenGadget3 is based on force accuracy rather than geometry alone. In this case, force contributions from a node of mass M and size l are computed only if the node satisfies a relative error condition of the form

$$\frac{GM\,l^2}{r^4} < \alpha\,|\mathbf{a}|, \tag{4.2}$$

where r is the distance between the particle and the node's centre of mass, $\mathbf{a}$ is the particle's acceleration from the previous time-step, and α is a tolerance parameter that controls the allowed relative force error.

In contrast to the short-range forces handled by the tree, the long-range component in a TreePM solver is computed on a PM grid. The grid size is typically chosen such that the number of cells is of the same order as the number of particles in the simulation. This ensures that the mesh resolves the large-scale density field with sufficient accuracy, while avoiding an unnecessary increase in memory usage and FFT cost, which scale with the total number of grid cells. Since the PM component is responsible for the long-range forces, its resolution only needs to capture scales larger than those treated by the tree.

As discussed in the section on hybrid methods (Sect. 2.3), the potential is split at a characteristic scale $R_{\rm cut}$. The value of $R_{\rm cut}$ is determined empirically from force-accuracy tests and is typically of the order of, but somewhat larger than, the PM grid spacing. Tests on force accuracy performed on Gadget2 to determine this setup are available in Springel (2005).

4.4 The Gravity solver on GPU

While aspects of this topic have been discussed in the literature, this section provides original contributions by tracing the source code directly, highlighting the main function calls, and describing the different implementations depending on compilation options.

Figure 4.4 illustrates how the PM and tree components are linked in the gravity solver; this flow chart corresponds to the function highlighted in purple (compute_grav_acceleration() in the previous flow chart (Fig. 4.1).

In the code configuration file, one can specify whether to compile the code with or without the PM grid. If the PM grid is enabled, the long-range forces are first

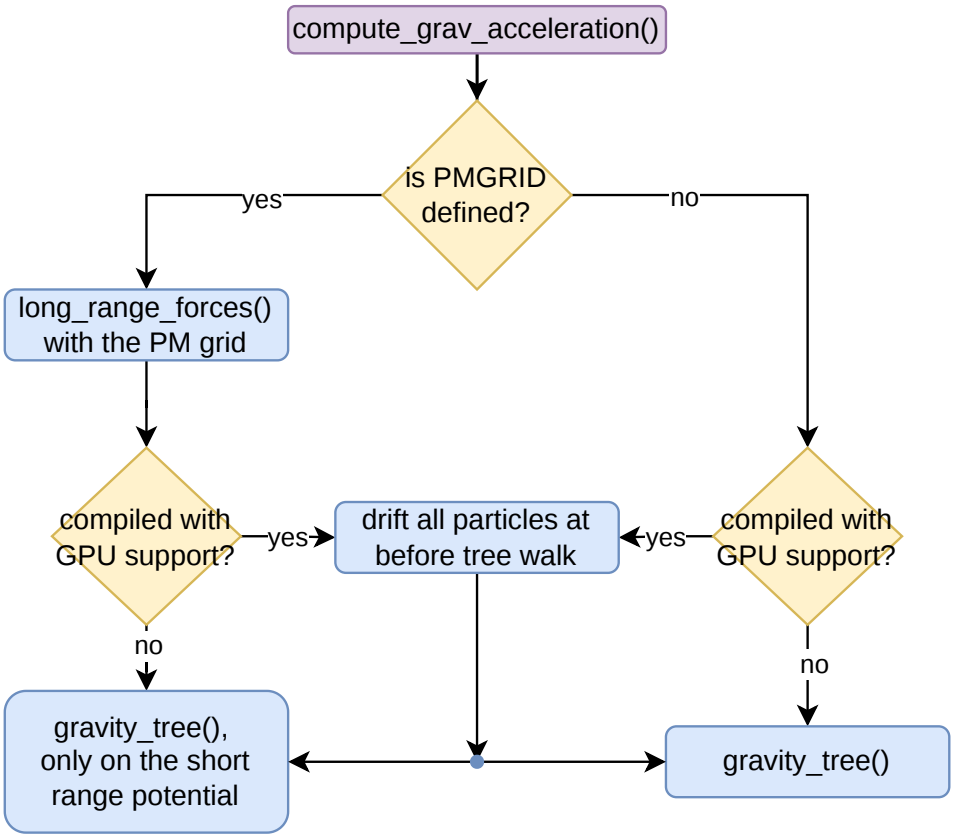


Figure 4.4. Flow chart for computing gravitational interactions

computed on the PM grid, followed by the short-range forces with the tree, which can optionally use GPU support as described later in this section. If the code is run in tree-only mode, the tree computes the full gravitational potential instead of just the short-range component. The portion of the gravity solver that has been ported to the GPU corresponds to the local tree computation of gravitational interactions (Ragagnin et al., 2020). In this section, we highlight the performance-critical steps of the solver and indicate where MPI, OpenMP, and GPU acceleration contribute. This level of detail is particularly important because the classical parallelisation in Gadget-2 does not include OpenMP support, and understanding these aspects is necessary to fully appreciate the scaling capabilities of OpenGadget3.

Execution Flow with Hybrid CPU/GPU Parallelisation

Figure 4.5 presents the execution flow of the `gravity_tree()` routine for a single MPI rank. The described logic is executed independently by each rank in the distributed-memory configuration. Within each rank, shared-memory parallelism is exploited via OpenMP on the CPU, while optional GPU acceleration is enabled through OpenACC.

At the beginning of the routine, it is determined whether the code was compiled with GPU support. If GPU acceleration is unavailable, the computation proceeds entirely on the CPU. If GPU support is enabled, a runtime condition checks whether the number of active particles on that rank exceeds a predefined threshold (10^3 particles). This avoids inefficient GPU launches for small workloads. Only when this condition is satisfied are the relevant particle data offloaded to the accelerator. As highlighted in Sec. 4.2, every MPI rank owns a section of the domain. The tree walk happens on every MPI rank, and the flow chart represents the control flow per MPI rank.

Primary Loop. The primary loop computes gravitational interactions for local particles (meaning particles whose domain belongs to that MPI rank) through a tree walk. On the CPU path, this loop is parallelised using OpenMP within each MPI rank, distributing particles across threads while ensuring thread-safe updates where necessary. On the GPU path, the primary loop is executed on the device asynchronously, allowing overlap between computation and communication. In the meantime, the CPU also does a tree walk, but exclusively to add to the export buffer the particles requiring interactions with domains belonging to other MPI ranks. Once the buffer is filled, MPI send/receive operations are triggered. In the GPU-enabled configuration, communication may proceed concurrently with device execution, enabling overlap between GPU kernels and MPI data transfer.

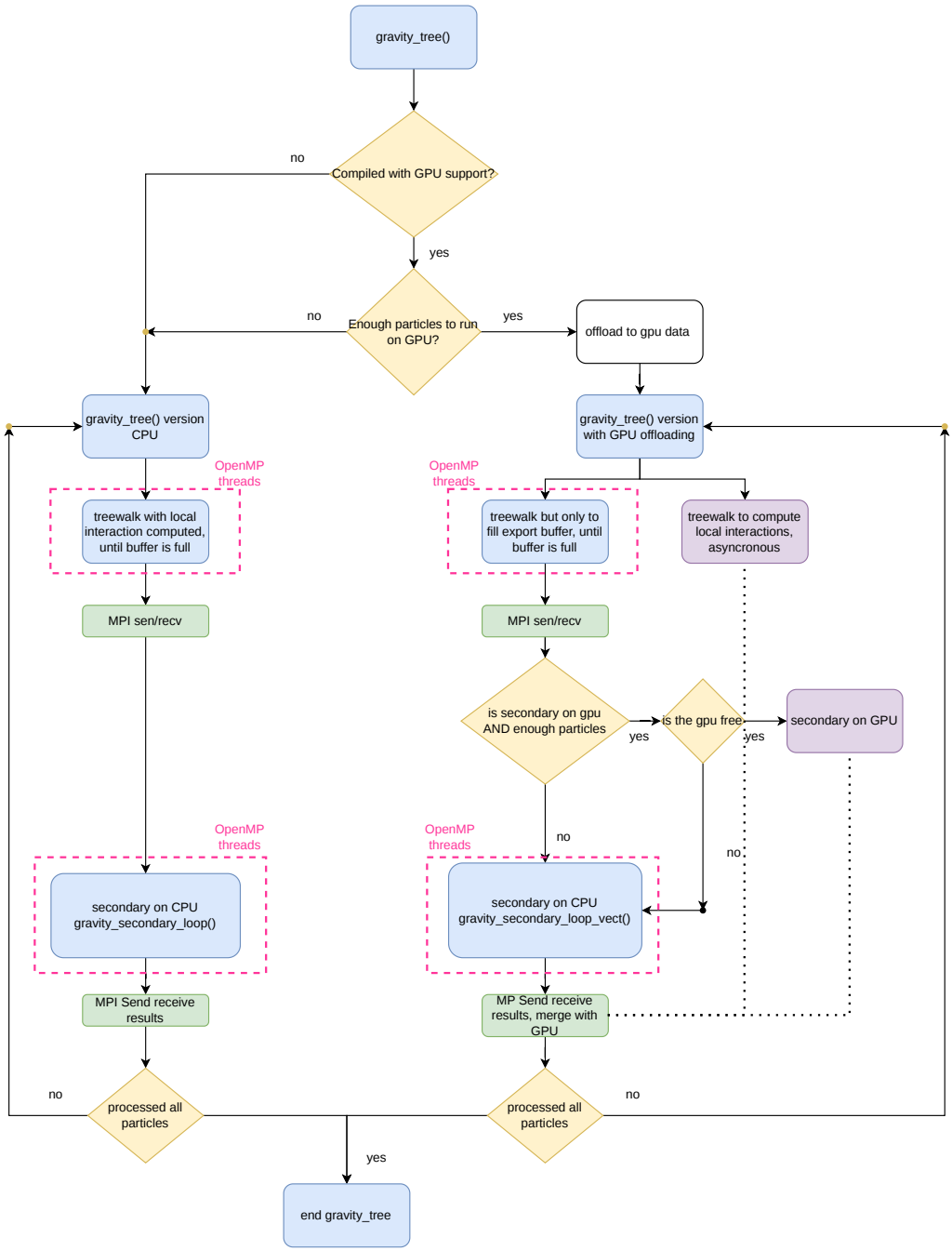


Figure 4.5. Flow chart for gravity_tree

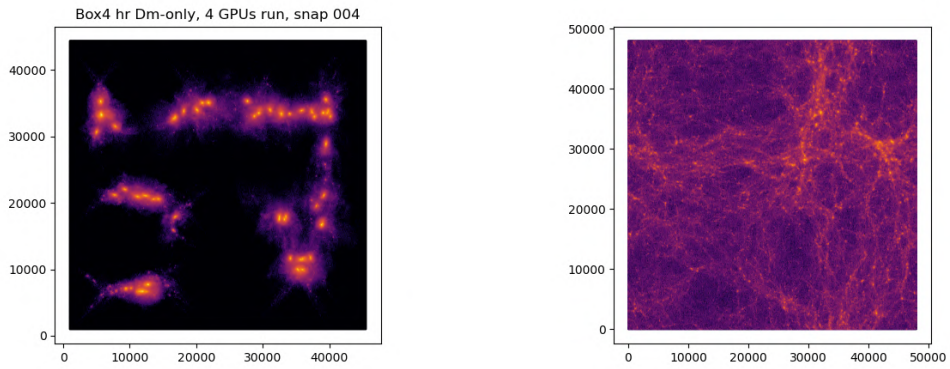


Figure 4.6. Effect of an MPI communication failure in a dark-matter-only simulation with 4 GPUs (4 MPI ranks) using the tree algorithm. Right: normal operation; left: snapshot illustrating the bug, where each rank only sees its own particles. This also visualises the domain decomposition across ranks.

Secondary Loop. After the MPI exchange, the secondary loop processes particles imported from other MPI ranks. On the CPU, this phase can be executed either through the vectorised OpenMP implementation (`gravity_secondary_loop_vect()`) or the standard OpenMP-parallelised version (`gravity_secondary_loop()`), depending on compilation flags.

In the GPU execution path, the code evaluates whether the secondary loop should also be offloaded. If sufficient particles are present on the rank and the GPU is available, the secondary loop is executed on the device; otherwise, the computation falls back to the OpenMP CPU implementation. This dynamic decision ensures efficient hardware utilisation, prevents GPU usage in cases where the overhead is dominant and avoids idle accelerator time.

Results from remote domains are communicated via MPI and merged with locally computed contributions on each rank. In the GPU case, this may involve combining device-computed results with data received from other ranks. The primary and secondary phases iterate until all particles assigned to the rank have been processed.

An illustrative example of an MPI communication issue is shown in Fig. 4.6. The figure presents two snapshots of dark-matter-only simulations run on 4 GPUs (4 MPI ranks) using the tree algorithm exclusively.

The right panel shows the simulation at a given redshift under normal operation, while the left panel illustrates the effect of a bug in MPI communication. In this case, contributions to the gravitational potential from the secondary loop are not properly computed, and each rank only has knowledge of its own particles. Consequently,

structures appear in isolation rather than as part of the full system. This figure also provides intuition about the domain decomposition across different MPI ranks.

4.5 Time Integration and Hierarchical Time-stepping

For time integration, Gadget employs the Leapfrog scheme in kick–drift–kick (KDK) form to achieve second-order accuracy. The method is symplectic and therefore maintains good long-term energy conservation despite requiring only a single force computation per timestep. Springel (2005) describes the use of hierarchical time steps in Gadget2. While OpenGadget3 builds on the unreleased Gadget3, some implementation details differ. In this section, we analyse the source code to show how hierarchical time steps are realised in OpenGadget3 and what additional modifications have been introduced to improve GPU performance.

Hierarchical Timesteps

To efficiently evolve systems with a large dynamical range while also reducing computational cost, Gadget uses hierarchical (or adaptive) time-steps. Particles are assigned to discrete time-bins whose time-steps are powers-of-two subdivisions of a global base time-step. As a result, not all particles are updated simultaneously. At each synchronisation point, only particles belonging to active time bins are advanced (drifted) and kicked. This is an implementation detail in which OpenGadget3 differs from Gadget2, where all particles were drifted at every time-step.

This adaptive strategy significantly improves computational efficiency, since now the code scales with the number of *active* particles at a specific time-bin. This introduces an important subtlety: at any given time, particles in different time bins may reside at different integration times. Consequently, before gravitational forces are evaluated, it must be ensured that all interacting particles are consistent. To maintain this consistency, particle drifting occurs in two distinct contexts:

- **Drifting at synchronisation points**
  - Executed at each synchronisation point determined by the hierarchical timestep scheme.
  - Advances *active particles* to the new global time, i.e., particles for which i is active in the current timestep are drifted.
  - Ensures that particles receiving kicks are temporally up-to-date.

- **Drift of encountered particles during the tree walk:**
  - Performed during force computation.
  - Ensures that neighbour particles participating in interactions are consistent with the current simulation time.
  - A particle is drifted "just-in-time" before its contribution to the force is evaluated.
  - On GPU, if acceleration is active, all particles are drifted at the beginning of the time-step to avoid costly conditional updates during the tree walk.

This inevitably leads to small differences in the particle trajectories during the evolution of the simulation. This is the compromise to optimise performance on GPU architectures, where the many cores favour using SIMD parallelisation strategies over "just in time" updates during the tree walk. This approach avoids repeated branching during the tree walk.

4.6 Full Scaling of OpenGadget3

The parallelisation design of OpenGadget3, combining distributed-memory parallelism (MPI across ranks), shared-memory parallelism (OpenMP within each rank), and optional accelerator offloading (OpenACC) is what allows OpenGadget3 to reach full scaling. The execution path is selected dynamically based on compilation options and local workload size, ensuring both scalability and efficient hardware utilisation across heterogeneous architectures. In this section we briefly discuss how the parallelisation design we just discussed impacts runtime and influences setup choices.

Figure 4.7 shows a performance test for the hybrid OMP/MPI parallelisation on CPU. The total number of computational resources is kept constant while varying the number of OpenMP threads per MPI process. In the resulting plot, the x-axis represents the number of OpenMP threads used within each MPI process, while the y-axis shows the relative performance change expressed as gain or loss with respect to a chosen baseline configuration. By presenting the results in terms of loss/gain, the plot highlights how shifting the balance between MPI processes and OpenMP threads in the hybrid parallelisation affects the overall performance. We can see that for the gravity_tree reducing the number of MPI excessively ranks hinders performance.

Figure 4.8 shows the strong scaling for OpenGadget3 with a DM only box. In a strong scaling test, the total workload remains constant while the number of processors is increased. The ideal scaling behaviour is described by Amdahl's Law, according

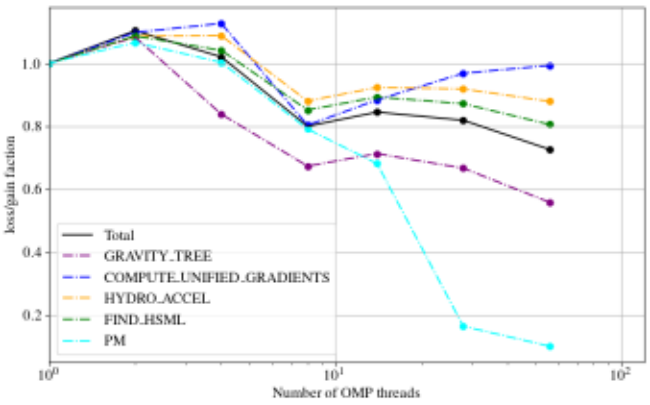


Figure 4.7. OpenMP performance of OpenGadget3 (on SuperMUC-NG), switching between MPI and OpenMP threads, provided by OpenGadget3 developers

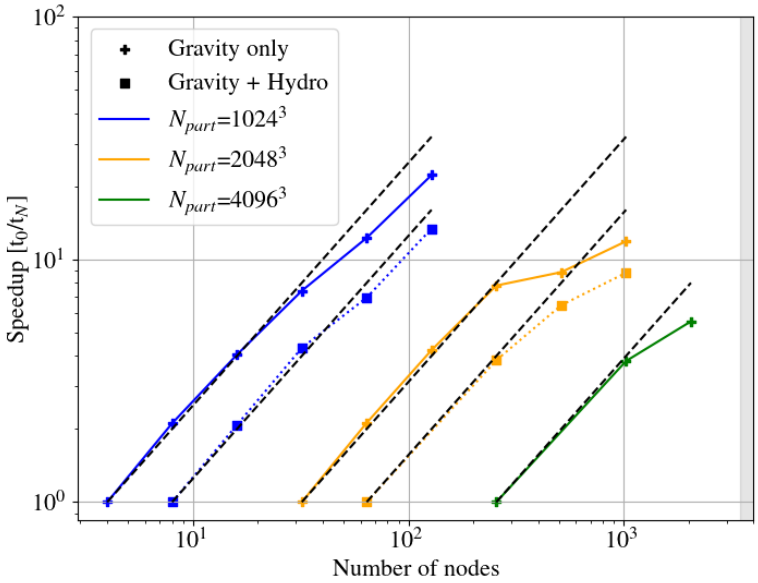


Figure 4.8. Strong scaling on Leonardo Booster using OpenGadget3 (from 06/2023) with a dm only box, showing the ability to use up to 1024 nodes with 4x A100 GPUs for each node, provided by OpenGadget3 developers

to which the theoretical speed-up ($speedup = \frac{runtime(N)}{runtime(baseline)}$) achievable when only a fraction of the program can be parallelised can be written as:

$$S(N) = \frac{1}{(1-p) + \frac{p}{N}}.$$

Where (p) denotes the fraction of the code that can be parallelised and (N) the number of processing units. This formulation shows that as the number of processors increases, the serial fraction (1 - p) becomes the limiting factor for performance improvements. Consequently, deviations from the ideal scaling curve in the strong scaling plot indicate the presence of additional overheads such as communication costs, synchronisation, or load imbalance.

4.7 Halo Finder and Output of the Simulations

As illustrated in this chapter, under standard conditions, the simulation ends when the final time is reached. In this section, we briefly describe the output generated by a dark-matter-only simulation.

The simulation produces snapshots at user-specified redshifts throughout the run. Each snapshot consists of the particle data at the corresponding simulation time, allowing analysis of the evolution of the system.

Gadget simulations can handle up to six different particle types. Gas particles (type=0), dark matter particles (type=1), star particles (type=4), black hole particles (type=5) and type=2/3 particles, which are either used as boundary particles (in so called zoom simulations) or are used as additional collision-less particles to describe initial disk stars (type=2) and bulge stars (type=3) in idealised galaxy simulations. In this work all simulations were DM-only, and they mostly used type-1 particles. For the applications of this thesis, the particle information provided in the snapshots that are relevant to us is: position, mass and particleID. The particle ID is conserved through the simulation. More details on the exact setup of the simulations used are available in Chapter 5.

Aside from information about single particles in the simulations, the code also outputs information about *groups*, namely haloes and subhaloes. The identification of haloes, virialised particle groups, is done employing a friend-of-friends (FOF) algorithm. According to this algorithm, particles with a separation lower than a *linking length* "b" are placed in the same group. With this approach, the formed particle groups correspond to regions enclosed by isodensity contours with a threshold value of about $\frac{r}{b^3}$. The choice of linking lengths is made so that the detected groups have the over-density characteristic

of virialised objects predicted by the spherical collapse model (Sec. 1.5.1). This method has proven to be effective in reproducing the predictions of the Press-Schechter (Press & Schechter, 1974) theory. For the identification of substructures within the main haloes the Subfind algorithm (Springel et al., 2001; Dolag et al., 2009) is employed. Subfind is generally run *on-the-fly*, meaning that is run together with the main simulation.

This algorithm is able to identify substructure, which in this context is defined as locally over-dense, self-bound particle groups within a larger parent group (the one identified with FoF). Subfind starts by computing a local estimate of the density at the positions of all particles in the input group. This is done in the way it is usually done for Smoothed-Particle-Hydrodynamics (SPH)

The locally over-dense regions are considered to be candidates for substructures. More specifically, these regions are defined as the ones enclosed by an isodensity contour that traverses a saddle point. From a practical perspective this is done using a global density threshold that is progressively lowered to reveal locally over-dense regions, which then form the set of substructure candidates. The outer 'edge' of the substructure candidate is determined by a density contour that passes through a saddle point of the density field. These saddle points represent where the substructure candidate joins onto the background structure.

In a final step, all substructure candidates are subjected to a gravitational unbinding procedure where only the self-bound part is retained as member of the substructure. To be registered as a subhalo this substructure must have a number of bound particles above a minimum detection threshold.

The output of subfind is provided using once again different types: haloes (type=0), subhaloes (type=1) and member particles (type=2). The data structure is shown in Fig. 4.9, which illustrates how the information is stored.

Since the analysis of the simulation output was carried out in Python, snapshot and group files were read using g3read (Ragagnin (n.d.)), a set of Python tools to read Gadget files.

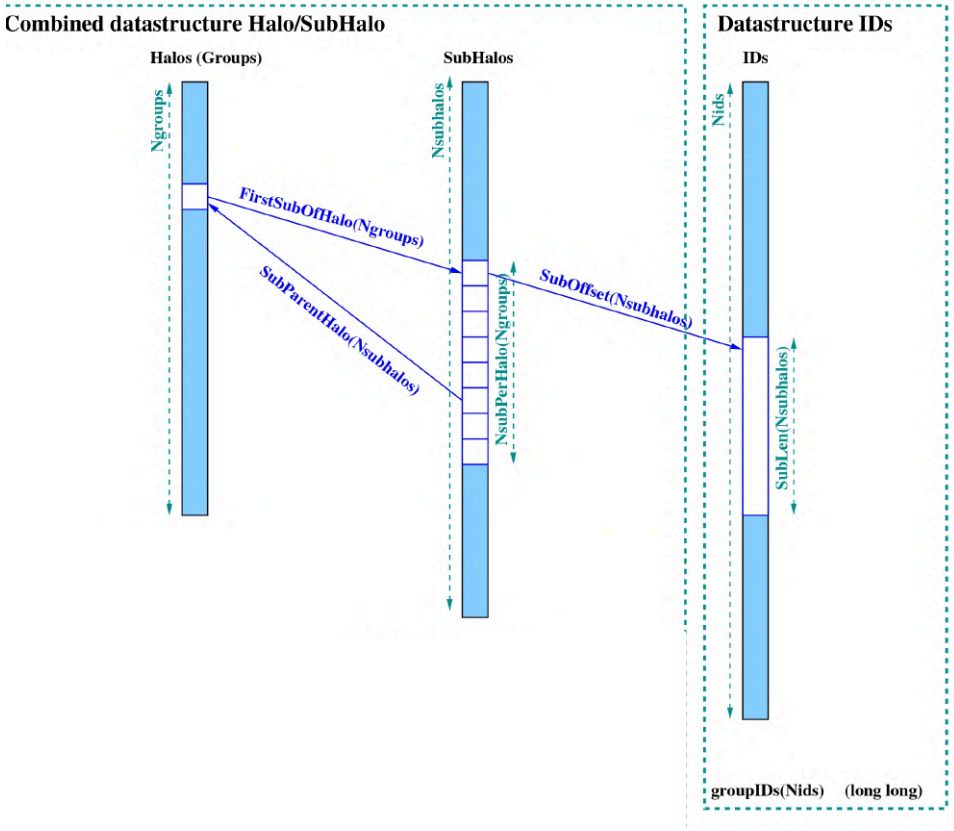


Figure 4.9. Data layout for the output of subfind: Haloes, and then subhaloes are ordered for decreasing mass, finally, particle IDs belonging to subhaloes.

Chapter 5

The Gravity tree on GPU

Stage IV cosmological surveys, such as Euclid, aim to measure the large-scale structure of the Universe with unprecedented statistical precision, requiring for their cosmological exploitation theoretical predictions of the matter power spectrum at the percent-level accuracy across a wide range of scales (Adamek et al., 2025).

Enabling GPU offloading in this context contributes to the speedup needed to run larger simulations at higher resolution; this efficiency must be accompanied by maintaining high levels of reliability and precision. It is difficult to study the convergence of cosmological simulations to this precision, especially in the nonlinear regimes where analytical models break down, and only high-resolution N-body simulations can provide reliable results.

This chapter discusses the validation tests we performed on the GPU offloading of OpenGadget3 (Ragagnin et al., 2020), discussed in Chap. 4. These tests probe different physical scales and statistical properties of the simulation outputs, including (i) comparison of the *matter power spectrum*, which characterises the statistical distribution of density fluctuations across spatial scales; (ii) analysis of the *halo mass function*, assessing the impact on the abundance of collapsed structures; and (iii) examination of the *radial density profiles of virialised haloes*, probing the internal structure of gravitationally bound systems. In each case, the results were compared with a reference simulation to assess convergence. In these tests, our goal is to vary only the gravity solver implementation, focusing on the CPU/GPU switch, keeping all other simulation settings identical. Simulations are therefore run with the same initial conditions, configuration files, and parameters across all fields, except those controlling the gravity solver. We acknowledge that other factors, such as box size, particle mass resolution, time-stepping, and gravitational softening, can also affect the convergence of the simulation. Previous studies have quantified these effects (Schneider et al., 2016; Sembolini et al., 2016; Power et al., 2003; Hernández-Aguayo et al., 2023), and we assess their impact on our analysis throughout this chapter.

The structure of this chapter is as follows: we first describe the simulation setup, then we proceed to the validation strategy, in which methods and results are presented together to facilitate comparison.

5.1 Simulation setup

We performed our simulations using the OpenGadget3 code. To perform our tests, we ran simulations of cosmological boxes with the same initial conditions adopted for the Magneticum simulations' Box4-hr (Dolag et al., 2025).

Table 5.1. Setup of the cosmological box

Box size	n Particles	Mass Resolution (type 0 → 1)	Mass Resolution (type 1 → 2)
48 [Mpc/h]	$2 \cdot 216^3$	1.4e8 [$M_\odot/h$]	6.9e6 [$M_\odot/h$]

The cosmological box has a size of 48 Mpc/h, with $2 * 216^3$ tracer particles, the setup of the cosmological box is summarised in 5.1. We ran DM-only simulations, starting from Initial Conditions (ICs) designed for hydrodynamical simulations, which included both type 0 and type 1 particles. The simulations were run with the simulation flag PATCH_IO, which shifts the particle types by 1. Type 0 particles from the Initial conditions (gas particles) will be treated like type 1 (dark matter particles), and type 1 like type 2. These type-2 particles, typically used as boundary particles in zoom-in simulations, behave like dark matter in cosmological boxes. Because our ICs have two particle types, Table 5.1 shows two different mass resolutions. This was taken into account during the analysis, especially when interpreting the output from Subfind for radial profiles.

Table 5.2 summarises the simulations used in this study, and we refer to it throughout this chapter when discussing the validation tests. In all tests, we compare simulations in which the gravity tree is executed with GPU support against two identical tree-only runs on CPU, which serve as a baseline reference.

Figure 5.1 shows the evolution of the density field for the reference tree-only run throughout the simulation, from redshift z=10 to z=0. The maps were generated by reading the particle positions and constructing two-dimensional hexagonal histograms using Python. Each hexagonal bin is colour-coded according to the local particle density, providing a visual representation of the growth of large-scale structure over time.

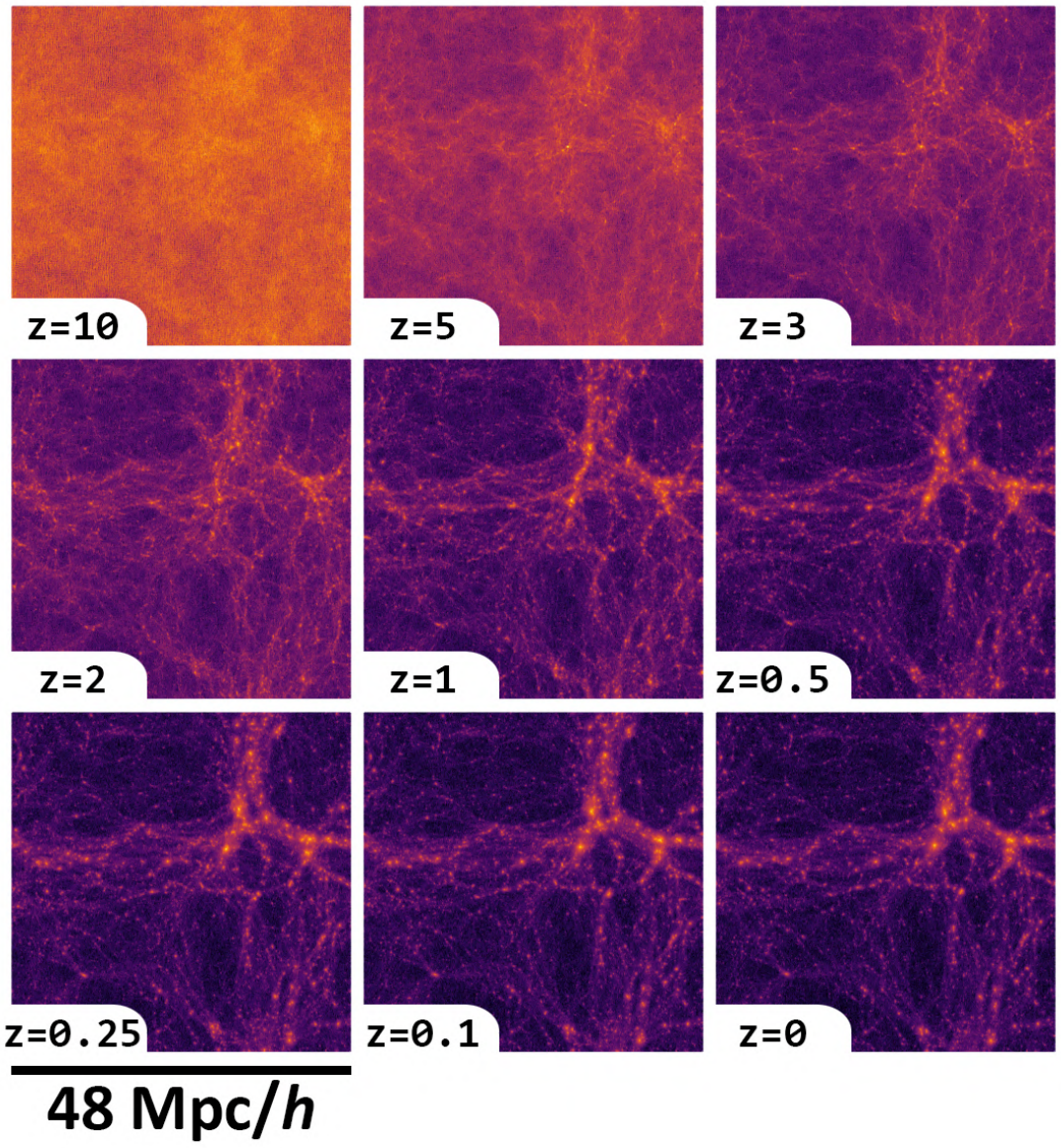


Figure 5.1. Density maps obtained from nine snapshots of the simulation, illustrating the evolution of the particle distribution from redshift z=10 to z=0. The maps are constructed from particle positions using a two-dimensional hexagonal histogram.

Table 5.2. Summary of the simulation runs used in this Chapter.

Run	Gravity Solver Setup	Purpose
TREE REF	Tree (CPU)	Reference run using the tree solver on CPU.
TREE	Tree (CPU)	Baseline run used for comparison with GPU implementations.
TREEPM	TreePM (CPU), 384^3 PM cells	Baseline run using the TreePM gravity solver.
TREE 1GPU	Tree (GPU), 1 MPI rank, 1 GPU	Validation of the GPU tree solver implementation.
TREE 2GPU	Tree (GPU), 2 MPI ranks, 2 GPUs	Test of MPI communication in multi-GPU runs.

5.2 The Matter Power Spectrum

The non-linear matter power spectrum is a key outcome of structure formation and allows us to test the convergence of numerical simulations.

The main goal of this section is to analyse the accuracy of the auto power spectrum $P(k)$. We measured $P(k)$ on a 128^3 grid using TSC interpolation; this grid size was chosen to minimise noise.

Figure 5.2 shows the resulting power spectra at redshift 0 for the different runs listed in Table 5.2. At small wavenumbers (large scales), the measurement is limited by the finite size of the simulation box; the fundamental mode can be computed, following Hernández-Aguayo et al. (2023), as:

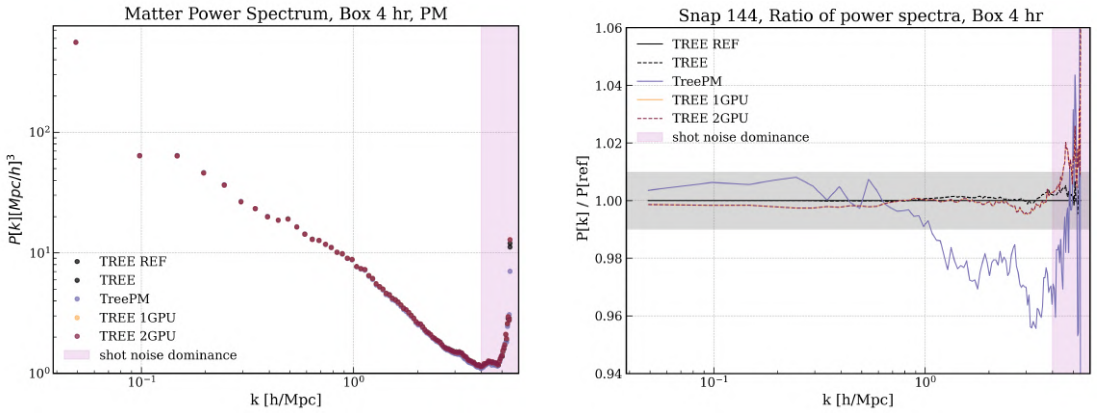
$$k_{box} = \frac{2\pi}{L} \tag{5.1}$$

where L is the size of the box, in our case $L = 48h^{-1}Mpc$. On the right side of the plot, the highlighted area indicates a range of values where the power spectrum is dominated by the Poissonian shot noise contribution:

$$P_{shot} = \frac{L^3}{N_p} \tag{5.2}$$

where N_p is the total number of particles.

Looking at the power spectrum in Fig. 5.2a we cannot see any visible differences, since we are looking at percent-level differences we repropose the power spectrum comparison in Fig 5.2b, which shows the power spectrum, normalised to the reference



(a) **(b)**

Figure 5.2. Comparison of the dark matter power spectrum computed on CPU and GPU at redshift $z = 0$. The left panel shows the matter power spectrum, the right the ratio of power spectra to the reference run

.

tree-only run, as $\frac{P(k)}{P_{ref}(k)}$. The shaded grey horizontal region represents the 1% difference region with respect to the reference run. For the simulations using the tree, on CPU and GPU, we observe differences well within the 1% region, until the power spectrum becomes non-linear. We find that CPU and GPU runs are in perfect agreement on all scales down to the point where the power spectrum is dominated by shot noise. We compared two tree-only runs to maximise the potential errors that could arise during GPU porting. As illustrated in Fig. 4.6 from the previous chapter, running the tree-only version (since it is the only part of the algorithm that was ported) provides a clear way to validate the implementation. In that figure, an evident bug is visible, demonstrating how errors in the tree algorithm can be exposed through this testing approach.

We have two baseline runs: one tree-only run identical to the reference, and one run using the standard TreePM setup. Typically, simulations of cosmological boxes employ the TreePM method as the standard configuration because it improves computational efficiency over the tree. Even in the original paper presenting the code (Springel, 2005), the PM grid is presented as an optional extension to the tree solver.

Figure 5.2 reveals significant differences between the power spectra obtained from the Tree and TreePM runs. Given that the underlying physical setup of the simulations is identical, these differences indicate that a numerical aspect of the solver affected convergence. This motivated a more systematic investigation, described in detail in Chap. 6.

5.3 Large Scale Structure: Halo Mass function

Dark matter haloes are the building blocks of cosmic structures; the study of their abundance and clustering is important for modern cosmological analysis. The abundance of dark matter haloes can be quantified by the halo mass function (Sect. 1.5.1), which gives the number of dark matter haloes as a function of halo mass, cosmic time, and cosmological parameters, in different mass intervals.

To compute the halo mass function (HMF), we used the halo masses provided by SUBFIND. The masses reported correspond to M_{200}, defined as the mass enclosed within a radius R_{200} where the mean density is 200 times the critical density of the Universe. The halo centres adopted for the spherical overdensity calculation in SUBFIND are defined as the positions of the particles with the minimum gravitational potential of the bound component of each friends-of-friends (FOF) group (Springel et al., 2001).

The quantity M_{200} is commonly used as a proxy for the virial mass of dark matter haloes. This is justified by the spherical collapse model (Sect. 1.5.1), according to which virialised structures are expected to reach an overdensity of order ~ 200 relative to the critical density.

We computed the HMF for each simulation realisation by binning haloes into equally spaced logarithmic mass intervals:

$$\Delta \log_{10} M_i = \log_{10}(M_{i+1}) - \log_{10}(M_i), \tag{5.3}$$

and counting the number of objects in each bin:

$$n(M_i) = \frac{N_i}{V \, \Delta \log_{10} M_i}, \tag{5.4}$$

where N_i be the number of haloes in bin i and V the simulation volume. Statistical uncertainties arise primarily from the discrete number of haloes (Poisson noise):

$$\sigma_i = \frac{\sqrt{N_i}}{V \, \Delta \log_{10} M_i}. \tag{5.5}$$

It should be noted that cosmic variance contributes to the uncertainty in the contributions from massive haloes.

In Figure 5.3 we show the HMF for the Tree reference run at redshift $z = 0$ and $z = 2.7$. For the data points at $z = 0$, the error bars are shown, representing the Poisson uncertainty arising from the finite number of haloes in each mass bin. The size of these error bars is small for low-mass haloes, according to Eq. 5.5. This illustrates

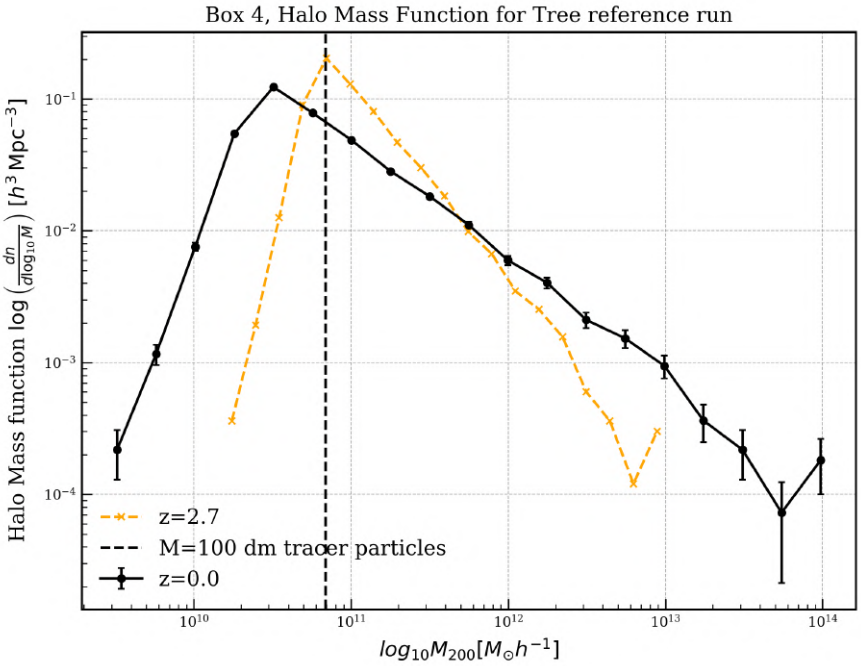


Figure 5.3. The halo mass function (HMF) for reference tree-only run on CPU at redshift $z = 0$, in black, and $z = 2.7$ in orange.

the statistical reliability of the HMF in different mass ranges: low-mass bins are well sampled, while the abundance of massive haloes is more uncertain due to their low counts.

The vertical dashed line in Fig. 5.3 marks the mass corresponding to one hundred dark matter particles, which we adopt as a conservative lower limit for reliably resolved haloes at the given mass resolution. Another indicator of the reliability of the HMF is the cutoff observed at the low-mass end. This cutoff arises from the limited number of particles in low-mass haloes, which prevents them from being fully resolved. The position of this cutoff depends on simulation resolution and gravitational softening; in a higher-resolution run, the cutoff would move to lower masses.

At redshift $z = 2.7$, a higher abundance of low-mass haloes is observed, consistent with the hierarchical nature of structure formation in Cold Dark Matter models. Since massive haloes are intrinsically rarer at higher redshifts, a larger simulation volume would be required to reliably sample them. The figure thus captures both the evolution of halo abundance with redshift and the limitations imposed by finite simulation resolution and volume.

Figure 5.4 displays the relative differences of the resulting halo mass function between

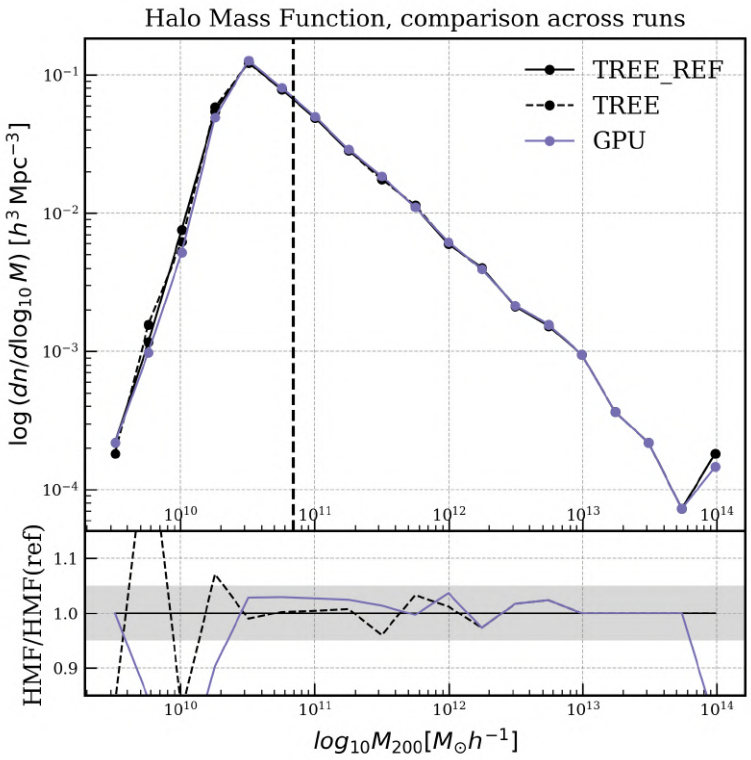


Figure 5.4. The halo mass function (HMF) at redshift $z = 0$ for tree runs of a Box4-hr Cosmological Box. In black, the CPU runs, in purple the GPU run. Haloes are binned in mass intervals equally spaced in base 10 logarithmic space.

the tree-only simulations with and without GPU. The observed differences, shown in the bottom panel as the ratio $\frac{HMF}{HMF_{ref}}$, are within 5% over scales correctly resolved by the simulation. Overall, the differences are consistent with values reported in the literature (e.g. Hernández-Aguayo et al. (2023)).

5.4 Radial Profiles of Virialised Structures

To compare the inner structure of virialised haloes, we analysed their radial profiles. Given the resolution of the simulation, this analysis was restricted to the three most massive haloes, which have masses of the order of $10^{14} M_{\odot}/h$.

Based on the experience gained during preliminary tests, we found that an accurate determination of the halo centre is crucial, particularly for haloes that deviate from spherical symmetry. To ensure a robust and consistent centring procedure, we adopted the shrinking sphere method described in Power et al. (2003). This method employs an iterative algorithm in which the centre of mass of particles contained within a sphere is computed recursively until convergence is reached. At each iteration, the centre of the sphere is reset to the most recently computed barycentre, and the radius of the sphere is reduced by 2.5 per cent. This procedure effectively converges towards the halo's densest region, minimising biases due to substructure or asymmetries.

Radial profiles were constructed by dividing the halo into concentric spherical shells, each containing the same number of particles (in our case, 10^3). This was achieved by sorting the particles according to their distance from the halo centre and slicing the resulting array accordingly. The minimum and maximum radii of each slice were then used to define the inner and outer boundaries of the corresponding shell, allowing the shell volume to be computed consistently.

As illustrated in Fig. 5.5, the resulting radial profiles show good agreement across the different haloes. Moreover, the internal substructure of the haloes is preserved, as confirmed by visual inspection.

Summary of the results

In this chapter, we tested the GPU offloading of the gravity tree (Barnes & Hut, 1986; Springel, 2005; Ragagnin et al., 2020) to verify its consistency with the CPU implementation. Validation was performed primarily using the dark matter power spectrum, along with checks of the halo mass function and radial density profiles across different simulation setups.

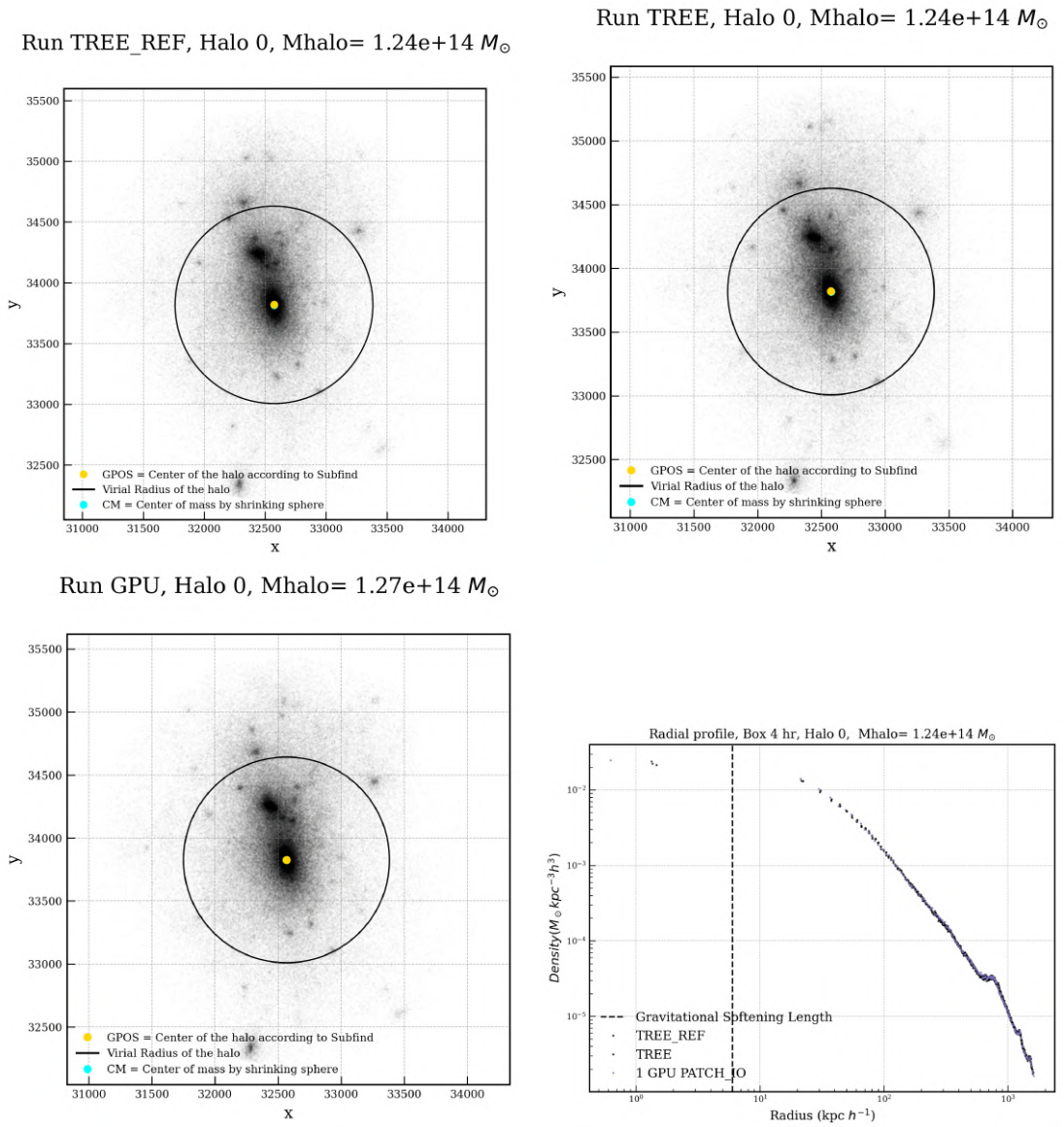


Figure 5.5. The first three panels represent the visualisation of the selected halo on the x-y plane for three different runs, with its virial radius (as per SUBFIND) and centre. In yellow, the center identified by subfind. The last panel shows the radial profiles. The vertical dashed line represents the scale of the gravitational softening length ($\epsilon = 6$).

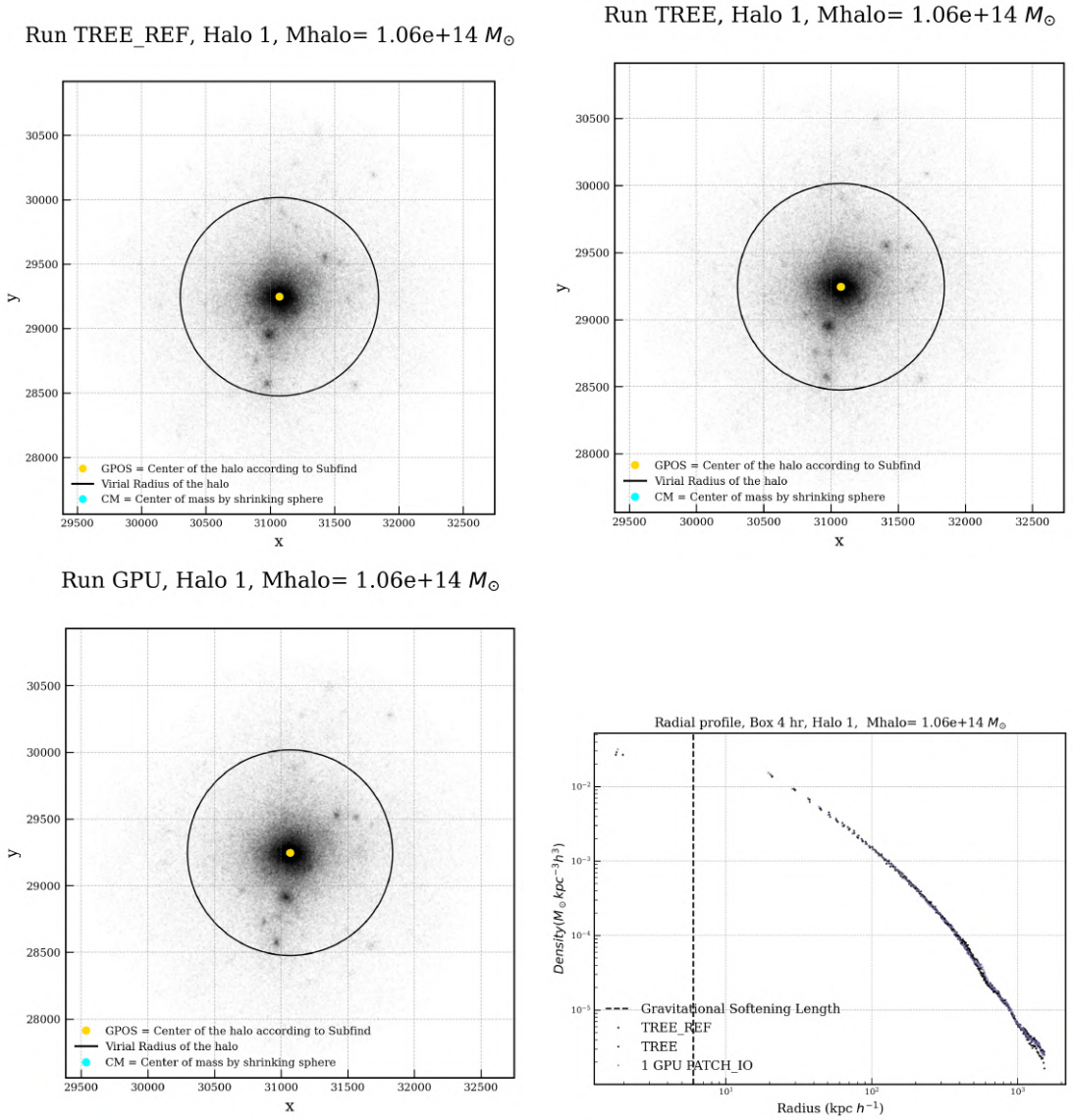


Figure 5.5. Continued.

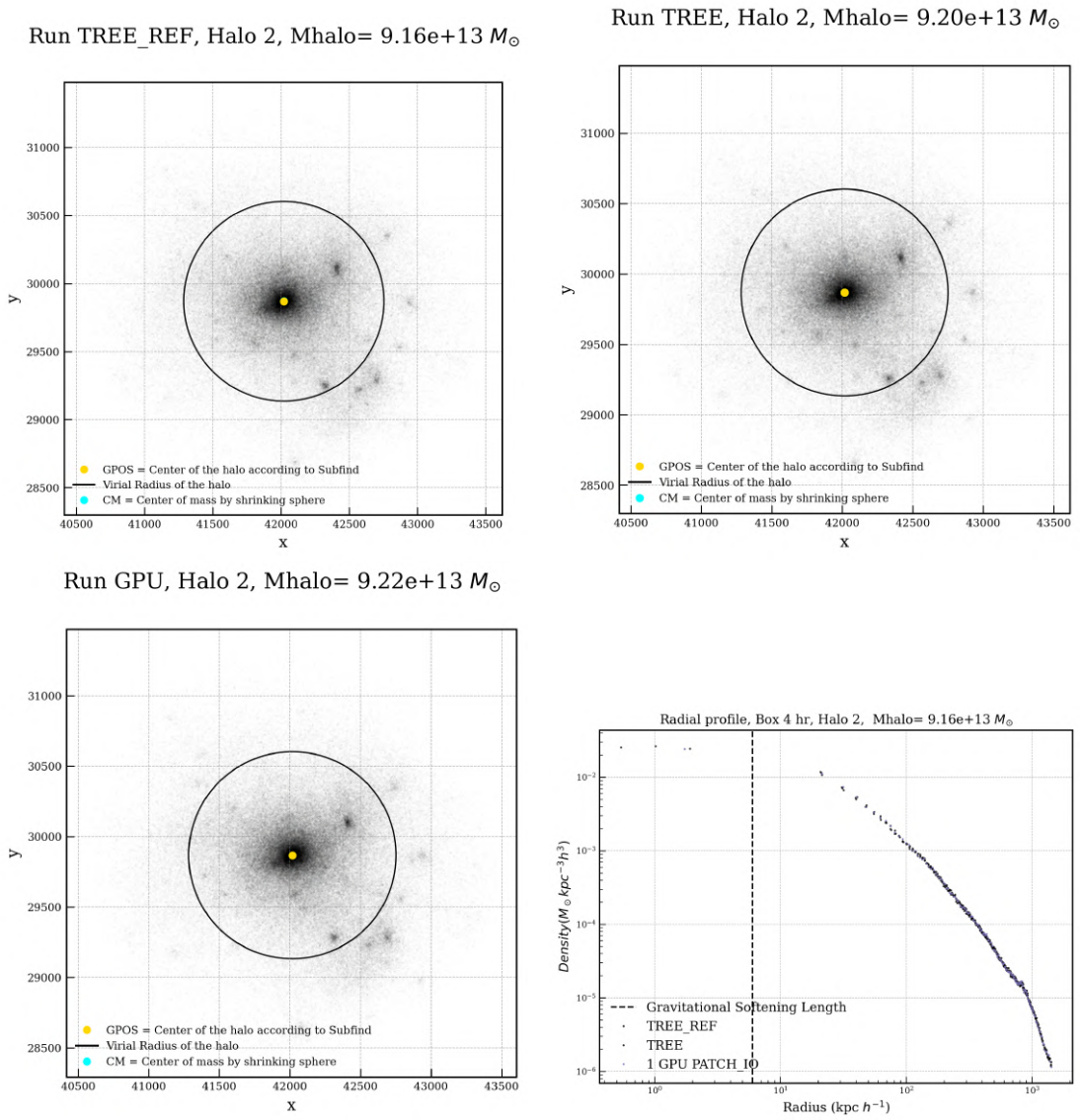


Figure 5.5. Continued.

We found that the power-spectrum differences remain below 0.05% across all well-sampled scales in our 48 Mpc/h box. The halo mass function agrees within 5% for haloes above the resolution threshold of $10^{11} M_{\odot}/h$, and the radial profiles exhibit consistent substructure upon visual inspection. These results confirm that the GPU offloading is reliable and reproduces the CPU tree solver with the desired accuracy.

Our analysis of the power spectrum revealed significant differences between the power spectra obtained from the Tree and TreePM runs on CPU. Given that the underlying physical setup of the simulations is identical, these differences indicate that a numerical aspect of the solver affected convergence. This motivated a more systematic investigation, described in detail in Chap. 6.

Chapter 6

The role of the PM grid in CPU-only runs

Building on the validation results of the GPU porting presented in Chap. 5, this chapter explores how variations in the treePM gravity solver setup influence the same observables. Motivated by the results of the power spectrum analysis presented in Section 5.2, we decided to investigate the differences observed in the dark matter power spectrum between simulations using the Tree and TreePM algorithms on CPU-only runs. In particular, we found that these differences extend beyond the 1% relative deviation region, indicating a potential sensitivity to the numerical scheme. Stepping away from the GPU implementation, we focus on the numerical parameters that determine the setup of the hybrid method treePM (Sect. 2.3) on the CPU.

Such insights not only provide guidance on the ideal setup for running the current version of the solver but also inform a possible roadmap for a comprehensive redesign. Through this chapter, the tests are run exclusively on CPU, without GPU offloading.

First, we step back from statistical analyses to focus on the underlying matter distribution (Sect. 6.1), examining halo positions and particle trajectories throughout cosmic history. This approach offers a complementary perspective on the lack of power in the non-linear regime observed in the previous chapter. In Section 6.2, we then return to the analysis of the auto power spectrum, running four sets of simulations designed to assess the impact of the solver's numerical configuration.

6.1 Comparing Tree and TreePM simulations in real space

In this section, we examine a behaviour observed in the simulations that provides complementary information to the matter power spectrum and helps clarify the role of the setup. Rather than relying solely on the statistical summary provided by the power spectrum, we analyse the particle distribution directly to build intuition for the observed behaviour.

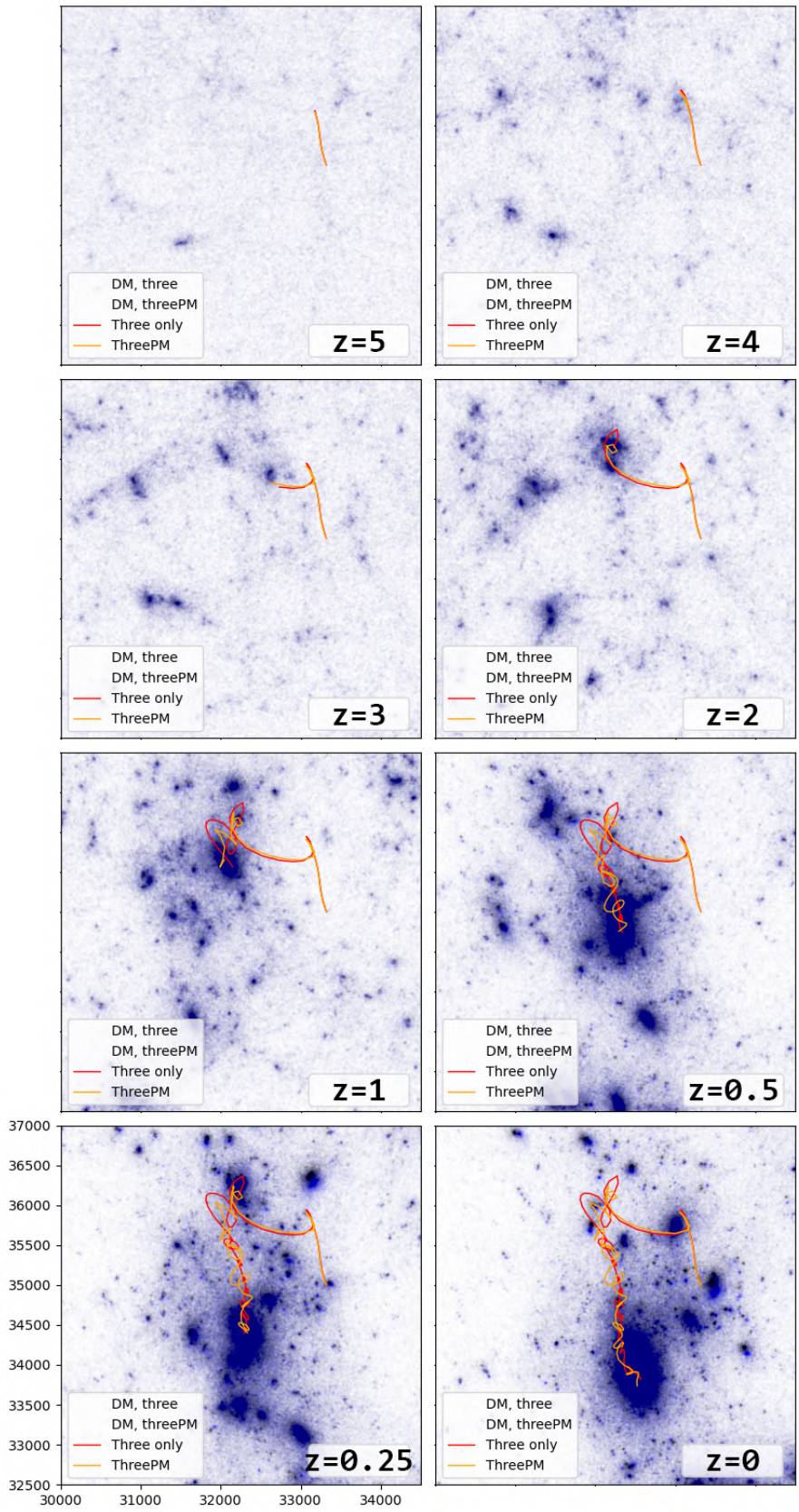


Figure 6.1. Trajectory of a particle across cosmic time, from two simulations. In red, the trajectory for the tree-only reference run, in yellow from a treePM run.

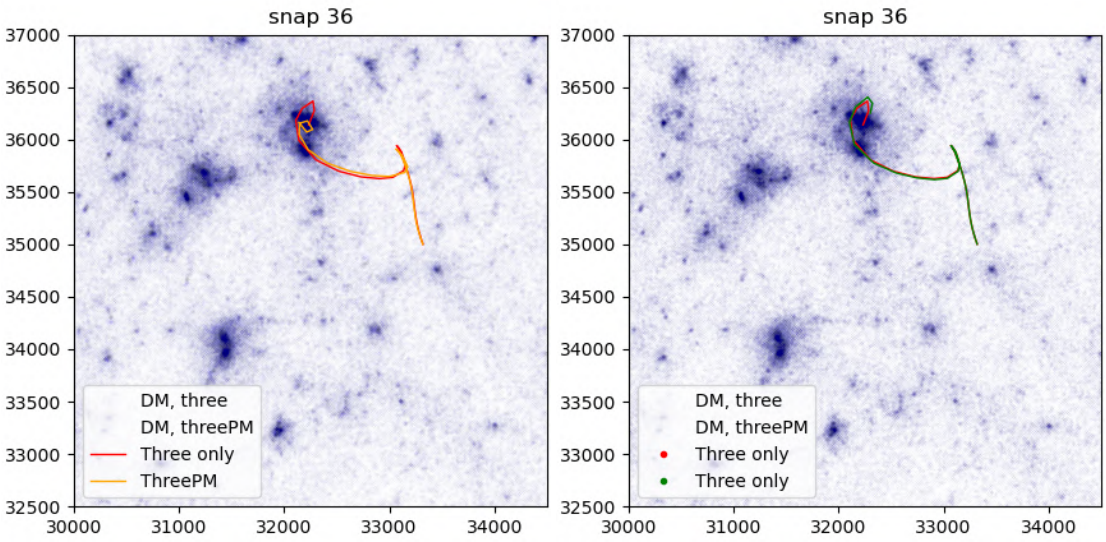


Figure 6.2. Particle trajectory comparison across simulations. The left panel shows deviations between the tree-only reference and the TreePM run, while the right panel compares the tree-only reference to an independent tree-only simulation.

6.1.1 Particle trajectories across cosmic time

This detailed investigation was motivated by early attempts to trace particle trajectories within virialised haloes. These observations suggested that the underlying matter distribution may be shifted between a tree-only and a treePM run. Figure 6.1 provides a concrete example: a single particle, identified in the reference run as belonging to a virialised halo at the end of the simulation, is traced backwards through time in both the tree and the treePM runs. The fourth panel (redshift $z = 2$) is particularly notable, as it illustrates the point at which the two trajectories begin to diverge, coinciding with the emergence of structure in the underlying matter distribution.

Figure 6.2 shows the comparison of the trajectories in Fig. 6.1 at redshift $z = 2$ compared with the trajectories for the same particle, at the same redshift, in two identical tree-only runs. This serves as a reference for understanding the behaviour seen for the tree/treePM comparison: unlike the divergence observed there, the trajectories in the identical tree runs remain closely aligned, illustrating the baseline level of variation when no differences exist between realisations.

We highlight that, for the correct interpretation of these results, one should not expect, even for identical runs, a perfect match in particle trajectories. Even when starting from identical initial conditions, the paths of individual particles rapidly diverge due to the chaotic nature of the N-body problem. Hemsendorf & Merritt (2002) provides

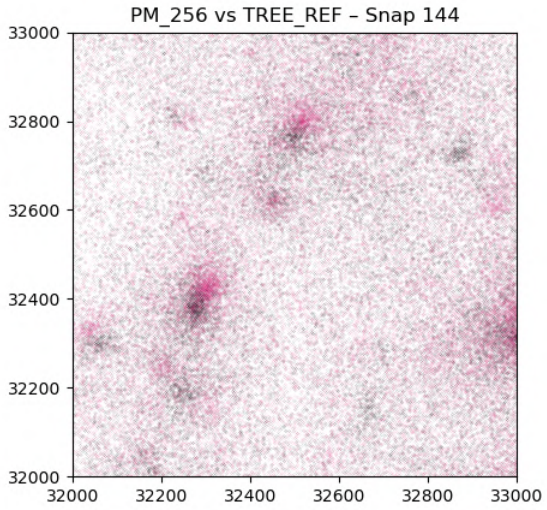


Figure 6.3. Cutout of a Box-4 DM only simulation, demonstrating the shift in formed structures across two different runs at redshift $z = 0$. In black, the particle distribution of a run using the tree algorithm, in pink, a run using treePM with 256 grid cells.

a detailed quantification of this phenomenon, using the characteristic e-folding time of the trajectories. Small numerical differences, such as round-off errors inherent to floating-point calculations, are exponentially amplified over time. This chaotic behaviour can be formally quantified using Lyapunov exponents, which measure the rate at which initially nearby trajectories separate. The associated e-folding time provides a characteristic timescale for this divergence, and according to this study, is typically much shorter than the system's crossing time. They conclude that although individual trajectories are highly sensitive, the system's macroscopic evolution can remain statistically predictable: the initial exponential growth of errors eventually saturates and stabilises at an amplitude that decreases as the particle number N increases. As a result, the macroscopic behaviour of the system resembles that of the smooth potential limit predicted by the collisionless Boltzmann equation (CBE), even though single-particle trajectories remain chaotic.

6.1.2 The offset in the structures' positions

These arguments about the chaotic nature of N-body systems and their implications for macroscopic dynamics apply to the analysis of cutouts from snapshots of our simulations. Figure 6.3 illustrates the shift in the formed structures in tree (black) and treePM (pink) simulations, on the order of tenths of kpc/h. This difference is more pronounced for small structures, as the offset between the structure's position is comparable to its virial

radius. No shift in the underlying matter distribution is observed when comparing the two identical tree-only runs. This tells us that, even for identical simulations, single trajectories can diverge, yet the overall density distribution is reproduced, whereas for the tree/treePM comparison, even the macroscopic behaviour is different. This motivates a closer examination of the numerical and physical factors responsible for the observed offsets.

Upon visual examination, we concluded there is no intrinsic shift in the grids; in other words, the shift in the structures is uneven within a single snapshot. This is also in agreement with the differences highlighted in the matter power spectrum; if the shift had been uniform, it would have had no impact on the power spectrum. We then analysed the shift of structures over time. By examining the evolution of this offset over time, we sought to determine whether the shift occurred before the N-body simulation even started, at a very early time in the simulation, or after mergers. This can help answer the question: could this offset be linked to errors in the forces in highly non-linear interactions?

This comparison proved to be nontrivial, as we had to cross-match haloes across runs and also back in time. During the bottom-up formation of structures, smaller haloes in the early universe merge to form bigger ones. The FoF algorithm used to detect haloes, when run on the fly (together with the simulation, providing output for every snapshot), allows access to haloes in order of decreasing mass. After a merger event, and in general during the evolution of the simulation, this ordering will not be maintained.

In the analysis pipeline, the cross-matching was implemented according to the following logic: both when matching across different runs and back in time, we look for haloes indexed close to the current halo, since the indexing is based on mass (see Fig. 4.9 for a better understanding of the indexing). Among the haloes close in mass range, the one closest (at minimum distance) to the current halo is selected. The match is accepted only if the distance is smaller than the virial radius. This simplified method, when compared, for example, with using merger trees, holds only because we compare offsets of massive structures in the simulation, and, most importantly, because we do not require high accuracy. The details of the algorithm are described in the pseudocode (1).

In the appendix B we shows the results (Fig. B.1) from this cross-matching procedure for the 10 most massive haloes (Fig. 5.4) in the simulation.

Using the distances of the 10 most massive haloes of the simulation with respect to their counterpart in the tree-only reference run, we then investigated the general

Algorithm 1: Backward halo matching between Tree and TreePM simulations

Input: OpenGadget3 Snapshots from simulations you want to compare, at different times

Output: Matched halo positions (centre)

foreach *of the N most massive haloes at* $z = 0$ **do**
- Initialise reference halo index i_{ref};
- Initialise reference halo position $\mathbf{x}_{\text{ref}}$;
- **for** *snapshot* $s = s_{\text{last}}$ *down to* s_{first} **do**
  - `// -- Reference run: track the halo backwards --`
  - Select candidate haloes in index range $[max(i_{\text{ref}} - \Delta i, 0),\ i_{\text{ref}} + \Delta i]$;
  - Compute distances between candidates and $\mathbf{x}_{\text{ref}}$;
  - Select the closest halo;
  - Update i_{ref} and $\mathbf{x}_{\text{ref}}$;
  - `// -- Comparison runs: match to reference halo --`
  - **foreach** *comparison run* **do**
    - Select candidate haloes around i_{ref};
    - Compute distances to $\mathbf{x}_{\text{ref}}$;
    - Select closest halo;
    - **if** *distance* $< R_{\text{vir}}^{\text{ref}}$ **then**
      - Accept halo as a valid match;
    - **else**
      - Reject halo match;

behaviour of halo offsets. To this end, we plotted the median offset across the 10 haloes in Figure 6.4. To reduce the influence of extreme outliers, the data were pre-filtered using a distance threshold of 120 kpc. Despite residual noise, likely due to minor mismatches in halo tracking, small-number statistics, and merger events, the plot shows that the offsets are not due to a small initial separation at very early times but rather arise during the evolution of the haloes. This is particularly evident when looking at the offset between redshifts $z = 4$ and $z = 0$. For the two identical tree-only runs, the offset oscillates around $10kpc/h$, this is a shift of the same order of magnitude as the gravitational softening used in the simulations ($\epsilon = 6kpc/h$). The tree-treePM offsets keep increasing within the specified redshift range, reaching values of $60 - 80kpc/h$, one order of magnitude larger than the gravitational softening.

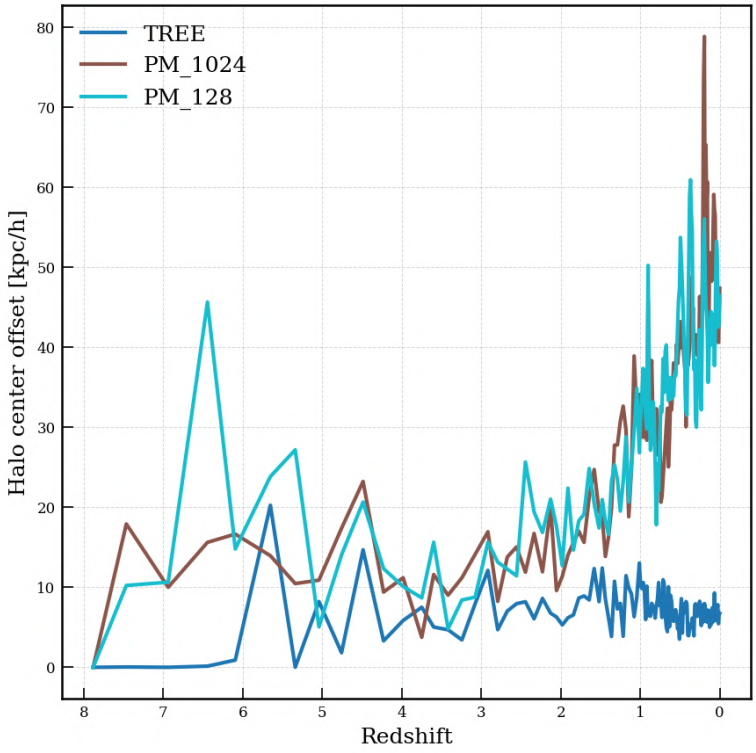


Figure 6.4. Median offset across the 10 tracked haloes as a function of cosmic time. The dark blue line shows the baseline median from the tree-only reference vs. tree-only comparison.

6.2 Numerical Parameters

Having understood that tracking these offsets would not yield a definitive answer, we decided to use the DM power spectrum itself to investigate the impact of several numerical parameters on the results. Throughout this section, we describe the effect that changing the parameters in Tab. 6.1 has on the Matter Power Spectrum. These include the size of the PM grid, which determines the resolution of the long-range force; the force-split scale between short- and long-range potentials, which sets the boundary between the tree and PM contributions; and the frequency at which the long-range potential is recomputed, which can influence the accuracy of time integration.

We always use two reference trees-only runs as a baseline, while the specific set of simulations for each test will be specified on a case-by-case basis.

At the end of the chapter we also explore how changing the opening criterion affects the DM power spectrum, using once again the parameters summarised in Tab. 6.1.

Table 6.1. Gravity Solver setup parameters

Parameter	Description
Tree / Particle-Mesh (PM) parameters	
`MaxRMSDisplacementFac`	Frequency over which the PM contribution to the force is recalculated.
`PMGRID`	Number of grid cells used to compute the density field for the PM method.
`RCUT`	Physical scale of the force split, expressed in units of PM grid cells.
TreePM opening criterion parameters	
Opening Criterion	By default set based on force accuracy; can be switched to a classical geometrical criterion.
`ErrTolTheta`	Parameter controlling the opening criterion when using a geometrical (Barnes–Hut–like) criterion.
`ErrTolForceAcc`	Parameter used to set the opening criterion based on force accuracy and to specify the target accuracy.

6.2.1 Size of the PM grid

The first test we ran was using different PM grid sizes, summarised in Tab. 6.2. The results are illustrated in Fig. 6.5. We tested five different PM grid sizes: the recommended 384 according to the literature for the box size, and 4 additional sizes, all

Table 6.2. Summary of PM grid size tests

Name of the run	Gravity Solver Setup (PMGRID)
TREE REF 1024	tree only
TREE	tree only
treePM 128	TreePM, PMGRID = 128
treePM 256	TreePM, PMGRID = 256
treePM 384	TreePM, PMGRID = 384 (recommended)
treePM 512	TreePM, PMGRID = 512
treePM 1024	TreePM, PMGRID = 1024

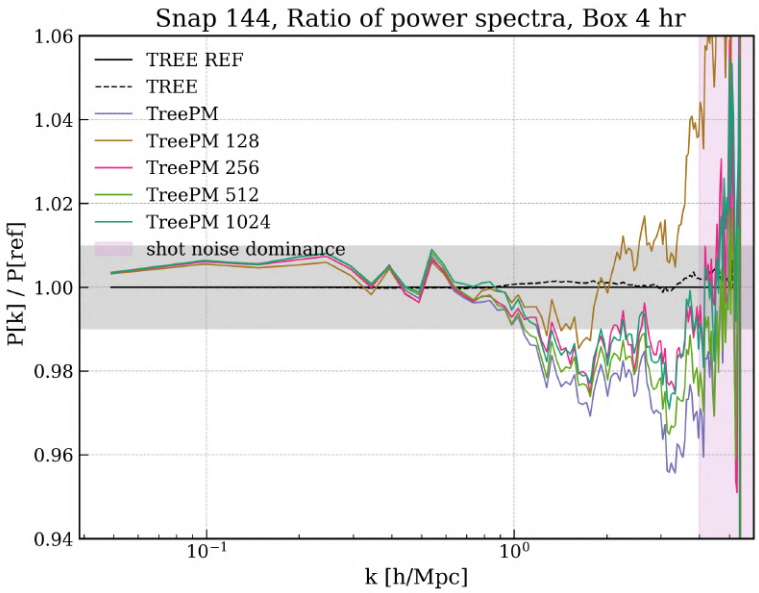


Figure 6.5. Ratios of the dark matter power spectrum to the tree-only reference run, for simulations using five different PM grid sizes

separated by a factor of 2. Using a Coarser PM grid should enhance the scale at which the tree is more relevant to the force computations. We observed no clear trend, but the power spectrum for the treePM 128 run, which behaves differently from the others in the non-linear regime.

6.2.2 Scale of the force split

Closely related to the size of the PM grid is the parameter R_{cut} (Sect. 2.3), which determines the scale of the force split. This scale is specified in the configuration file in units of the PM grid, so different PM grids correspond to different physical force split scales. The runs discussed here are summarised in Tab. 6.3, and Figure 6.6 shows the

Table 6.3. PM grid and R_{cut} settings for selected runs. R_{cut} is given in units of the PM grid.

Name of the run	PMGRID	R_{cut} [PM grid units]
TreePM 128	128	6
TreePM 256	256	6
TreePM 256	256	12

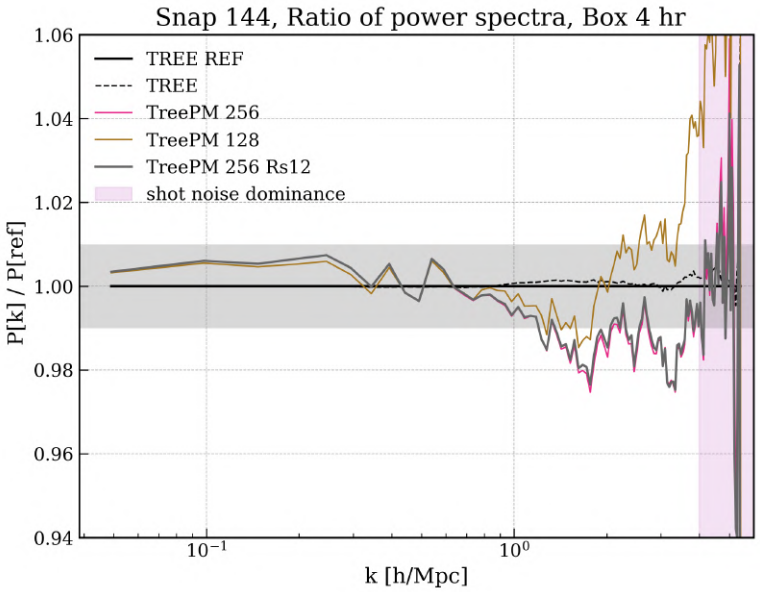


Figure 6.6. Ratios of the dark matter power spectrum to the tree-only reference run, for simulations using different force split scales R_s.

results of this test. The default value for R_{cut} is 6. The TreePM 128 and TreePM 256 runs with $R_{\text{cut}} = 12$, therefore have the same physical scale for the force split. We again observe no significant changes in the dark matter power spectrum, indicating that the standard choice $R_{\text{cut}} = 6$ provides reliable convergence.

However, one conclusion from this test is that the unusual behaviour of TreePM 128, which shows a deviation in the non-linear regime of the power spectrum compared to other PM grid settings, is due to the grid cells being too large. This results in a loss of resolution at small scales when combining the tree and PM forces. This effect does not propagate to the TreePM 256 run with $R_{\text{cut}} = 12$, because even though the physical force split scale is the same, the PM grid is computed at higher resolution.

Table 6.4. Summary of runs testing the frequency of PM grid contribution is computed (`MaxRMSDisplacementFac`).

Name of the run	`PMGRID`	`MaxRMSDisplacementFac`)
TREEPM	384	0.25 (default)
TREEPM	384	0.1
Run 3	384	0.05

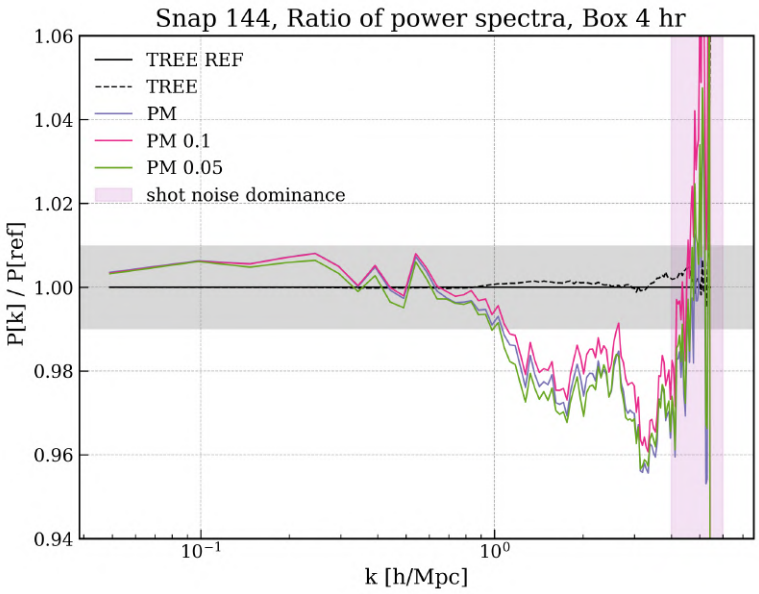


Figure 6.7. Ratios of the dark matter power spectrum relative to the tree-only reference run, shown for simulations with progressively higher update frequencies of the long-range contribution of the PM grid.

6.2.3 Timestep for long range contribution

Another hypothesis we tested was that the PM grid contribution to the force was not computed sufficiently often, which could introduce systematic differences in the forces. The PM grid contribution (the long range potential described in Sec. 2.3) is not recomputed at every time step for efficiency reasons. The frequency is determined by the parameter `MaxRMSDisplacementFac` in the Config file. The results for these runs are displayed in figure 6.7, and we can see that there is no relevant change in the power spectrum when choosing to update the long-range contribution more often. We therefore conclude that the standard frequency is adequate, and it cannot justify the observed differences between tree and treePM runs.

6.2.4 The opening criterion

Table 6.5. Summary of runs testing the Opening Criterion for the tree.

Name of the run	Opening Criterion	value
TREE	`ErrTolForceAcc`	0.005(default)
TREE 0.45	`ErrTolTheta`	0.45 (literature value)
TREE 0.2	`ErrTolTheta`	0.2

Finally, to understand how to bridge the gap in the power spectra and to prioritise the setup to use in future production runs, we sought to identify what led us closer to "correct" results. This is not easy, as the correct solution for the matter power spectrum is not known. To assess correctness, we would, in principle, have wanted to resort to direct summation. As this is extremely computationally expensive, we started playing with the aperture parameter (Sect. 2.1). Table 6.5 summarises the run used for this test, compared to a treePM 256 run. As already outlined, the default criterion to open tree nodes in OpenGadget3 is based on force accuracy. To run the numerical experiments, we found it more intuitive to use the geometrical (Barnes–Hut–like) opening criterion using the paramter `ErrorTolTheta`, set to 0.45 as prescribed in Springel (2005). The resulting power spectrum is displayed in Figure 6.8 in blue. Reducing the value by half affects the volume by a factor of 8, bringing many more particles into the threshold for direct summation, or making smaller clusters of particles approximated by the multiple expansion around their centre of mass. This reflects in the power spectrum. As seen in Figure 6.8, the dashed green line(corresponding to tree-only with geometrical opening criterion = 0.2) aligns more closely with the profiles obtained with the treePM algorithm.

We conclude that the differences in the power spectrum across all scales are attributable to the tree algorithm not treating particles symmetrically when computing their contributions to the potential.

Summary of the results

In this chapter, we investigated the setup of the hybrid TreePM gravity solver on CPU, motivated by the lack of power of up to 4% on scales larger than $k = 1$ h/Mpc and disagreements of about 1% in the mildly non-linear regime observed between Tree and TreePM runs. We studied the impact of the numerical setup first in the real space, looking at particle trajectories in the context of chaotic systems and at positions of haloes across different runs. We then connected the observed offset in the formed

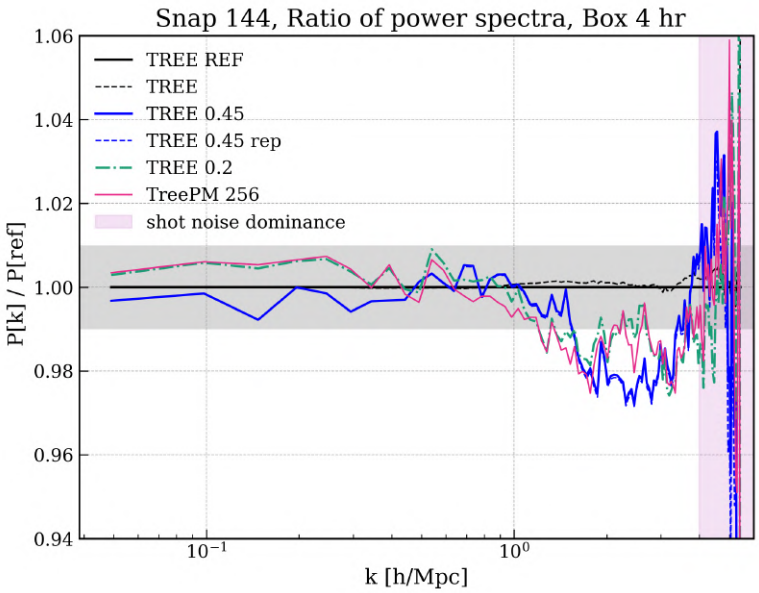


Figure 6.8. Ratios of the dark matter power spectrum relative to the tree-only reference run, shown for TreePM 256 and tree-only simulations with two different geometrical opening criteria. The TreePM 256 results are shown in purple, and the tree-only run with opening angle 0.2 in green, illustrating negligible differences in the power spectrum.

structures between tree and treePM runs to the observed lack of power in the DM power spectrum. Through the matter power spectrum, we studied the effects on convergence of: (i) the size of the PM grid, (ii) the scale for the force split between short- and long-range potential, and (iii) the frequency of recomputation of the long-range potential. We found that results remained stable, and concluded that the canonical choice of these parameters is justified. Therefore, the observed differences in the power spectrum cannot be attributed to inaccuracies in the PM grid setup.

We identified the differences observed in the power spectra as arising from the accuracy of the tree. By testing the effect of reducing the opening angle in tree-only simulations, we see the differences between tree and treePM decrease to well below 1%. We therefore conclude that the previously observed discrepancies can be attributed to the approximate nature of the tree method. In tree algorithms, the gravitational force on each particle is computed independently by walking the hierarchical tree structure. As a consequence, the interaction between two particles is not necessarily evaluated in a symmetric way. For example, when computing the force on particle A, the cell containing particle B may be approximated by a multipole expansion, whereas during the tree walk for particle B, the interaction with particle A may be computed directly. Because the two forces are evaluated using different approximations, the numerical

forces are not guaranteed to satisfy exact action–reaction symmetry. This asymmetry can lead to small deviations in momentum conservation. Using a stricter opening criterion reduces this effect by bringing more particles into the direct summation regime, increasing accuracy at the cost of performance.

Conclusions and Outlook

Stage IV cosmological surveys such as Euclid (Mellier et al., 2025) and DESI (Aghamousa et al., 2016) aim to measure the large-scale structure of the Universe with unprecedented statistical precision, requiring for their full cosmological exploitation theoretical predictions of the matter power spectrum and other cosmological observables at the percent-level accuracy across a wide range of scales (Adamek et al., 2025). Using GPU-enabled cosmological codes contributes to the speedup needed to run larger simulations at higher resolution; this efficiency must be accompanied by reliability and precision. Validating gravity solvers to such precision is a challenge, since in the nonlinear regime analytical models break down, and only high-resolution N-body simulations can provide reliable results.

In this thesis, we have studied the numerical implementation of the Gravity Solver in OpenGadget3 (Ragagnin et al., 2020) and tested its convergence through validation tests that probe scales relevant to cosmology. In particular, we considered (i) the *matter power spectrum*, which characterises the statistical distribution of density fluctuations across spatial scales; (ii) the *halo mass function*, assessing the impact on the abundance of collapsed structures; and (iii) the *radial density profiles of virialised haloes*, probing the internal structure of gravitationally bound systems.

Results

In this work, we carried out two main investigations. First (Chap. 5), we tested whether the GPU offloading of the gravity tree (Barnes & Hut, 1986; Springel, 2005) is consistent with the CPU implementation, validating it using mainly the dark matter power spectrum and verifying that the halo mass function and radial profiles are reproduced across runs with different setup. We observed: (i) differences in power spectrum of the order of 0.05% across all scales properly sampled in our 48 Mpc/h box; (ii) differences in the halo mass function within 1% for haloes with masses above the resolution threshold of $10^{11} M_{\odot}/h$, and (iii) consistent radial profiles for haloes

with reproduced substructure upon visual inspection, therefore we concluded that the offloading is consistent and provides the desired accuracy against the reference tree-only run on CPU.

Second, as we observed a lack of power of up to 4% on scales larger than $k = 1$ h/Mpc, together with disagreements of the order of 1% in the mildly non-linear regime between Tree and TreePM runs, we carried out (Chap. 6) extensive testing of the setup of the hybrid gravity solver on CPU.

We hypothesised that the PM grid setup was introducing some errors, therefore we studied the effects on convergence of: (i) the size of the PM grid, (ii) the scale for the force split between short- and long-range potential and (iii) the frequency over which the long-range potential is recomputed. We observed no clear trend in the power spectra from these tests, that remained overall unchanged.

We then tested the effect of reducing the opening angle in the tree-only simulations and compared this new power spectrum for tree-only with stricter opening criterion to the treePM simulations and observe the differences in the power spectra almost disappear. Minimal differences (well within 1%) were observed when comparing our more accurate tree-only runs to the canonical treePM setup.

We therefore conclude that the previously observed discrepancies can be attributed to the approximate nature of the tree method, and not to issues with the PM grid. Because tree walks approximate distant particle groups differently for each particle, the computed forces are not strictly symmetric, leading to small deviations from exact momentum conservation. Imposing a stricter opening criterion for the tree, when simulations are run tree-only, reduces this effect by bringing more particles into the direct summation regime, thereby increasing accuracy but degrading performance.

From these validations tests we conclude that: (i) the canonical choice for the parameters linking tree and PM is justified and (ii) the differences observed in the power spectra we observed in the tests were attributed to a lack of accuracy in tree-only runs.

The results presented in this work provide a basis for informed decisions on solver design in the era of GPU-based high-performance computing, where many particles can be handled in the direct-summation regime. In this context, our findings confirm that including the PM grid is essential for achieving accuracy and ensuring convergence without excessively increasing the number of direct computations.

Future prospects

Building on the results of this thesis, several interesting extensions of this analysis are possible. Our simulations were performed using initial conditions from the Magneticum

Box4-hr dark-matter-only runs. This box has a size of 48 Mpc/h, which limits the range of large scales that can be probed. While this box size provided a practical compromise for performing multiple numerical experiments given the available computational resources, running simulations as large as Magneticum Box2b ($640 Mpc/h$) would allow us to probe smaller wavenumbers and better resolve the BAO scale, which is an important probe for cosmology. Larger boxes are also motivated by previous studies emphasising the impact of box size and cosmic variance on simulation results (Hernández-Aguayo et al., 2023).

A valuable use of computational resources for running a large box would be to participate in the code comparison project presented in Schneider et al. (2016). For this project, initial conditions were made publicly available to encourage systematic comparisons between different codes, and the box size of 500 Mpc/h allows the study to be extended to significantly larger scales. Since there is no unique "correct solution" for the matter power spectrum, the scientific community must converge towards a generally accepted reference. Participating in a code comparison project is an effective way to contribute to this effort and to place one's work in a broader perspective.

Since our simulations were dark-matter-only, we cannot assess the impact of hydrodynamics. A natural next step would therefore be to extend this analysis to hydrodynamical non-radiative simulations, as hydrodynamics affects small-scale structures and leads to an increase of power at small scales. Because most observational probes of the Universe rely on baryonic luminous matter, and the dark matter distribution is often reconstructed from luminosity–mass relations, full-physics simulations aim to reproduce observables by incorporating galaxy evolution models that capture processes below the resolution scale. Carrying out convergence studies in this context would therefore be extremely valuable. However, connecting physical insight to numerical convergence in full-physics simulations remains challenging, as these simulations rely on subgrid models to capture processes such as AGN feedback and star formation. These models contain several free parameters and are highly sensitive to their calibration. As a result, careful studies will be required to disentangle numerical effects from modelling choices when extending analyses such as the one presented here to full-physics simulations.

III

Appendix

Chapter A

Monte Carlo-like simulations on GPU

This appendix presents a benchmark illustrating the performance of GPU architectures for embarrassingly parallel workloads. As discussed in Chapter 3, GPUs are particularly well suited to problems in which a large number of independent operations can be executed simultaneously following the SIMD/SIMT paradigm.

Monte Carlo methods are a canonical example of such workloads, since each sample can be computed independently of the others, with occasional need for synchronisation/reductions during iterative methods. To illustrate this we present CAM(L)AMBERT – Computationally Accelerated Module for Lambert, a novel GPU accelerated Lambert solver developed as a technology demonstrator to explore the advantages of modern parallel implementations for astrodynamics applications during an internship carried out at the European Space Agency.

The Lambert solver is a foundational tool in astrodynamics, enabling the determination of an orbital trajectory that connects two position vectors in a specified time of flight, in Keplerian dynamics. Since the earliest developments of space mission design, numerous analytical and numerical techniques have been formulated to solve Lambert's orbital boundary value problem (Lancaster et al., 1969; Gooding, 1990; Battin, 1987). The increasing demand for large-scale trajectory optimisation and rapid mission design studies would benefit from a high-performance, massively parallel implementation of the Lambert solver.

The solver is implemented in C++/CUDA and leverages CUDAjectory' header-only libraries (Masat et al., 2021; Inno et al., 2022), providing types for efficient vector algebra in GPU-accelerated code, to enable efficient, intuitive and memory safe computations on modern accelerators, along with simplified transition from directive-based parallelisation on CPU. The tool is based on the algorithm introduced in Izzo (2014), selected for its robustness, accuracy, and its widespread adoption and recognition within the field.

Particular focus was placed on identifying and minimising data dependencies within the algorithm to enable fine-grained parallelisation across very large input datasets

characteristic of Monte Carlo analyses and mission design studies. The main technical steps involved in ensuring correctness and performance were:

- identification of data dependencies in the algorithm and Python implementation
- C++ implementation with object-oriented programming and unit tests
- CPU fine-grained parallelisation using OMP
- GPU parallelisation with CUDA kernels

The tool was evaluated against Godot (GODOT Team, ESA Flight Dynamics, 2025), the standard tool used in ESA for these studies, both for correctness and computational performance. We summarise the performance gain achieved on CPU with strong scaling tests and on GPU in Fig. A.1. CAM(L)AMBERT consistently matches the numerical accuracy of GODOT. Performance tests show a 20x speedup for 28 OpenMP threads. Even with non-parallelised executions, CAM(L)AMBERT retains an almost 2x speed advantage over GODOT, thanks to the intrinsic memory management and the automatic internal vectorisation. Performance benchmarking on one million samples demonstrates a three-order-of-magnitude (1000x) speedup using an NVIDIA V100 GPU compared to GodotPy with 8 OpenMP threads (the most favourable setup for GodotPy in terms of runtime). Even on an older, lower-end workstation GPU (Nvidia QUADRO P2000), the tool delivers a two-order-of-magnitude (100x) acceleration over GodotPy.

Overall, CAM(L)AMBERT demonstrates the substantial benefits of GPU acceleration for classical astrodynamics algorithms. Beyond serving as a technology demonstrator for the use of GPUs, it provides a scalable foundation for integration into next-generation trajectory design tools and mission analysis frameworks.

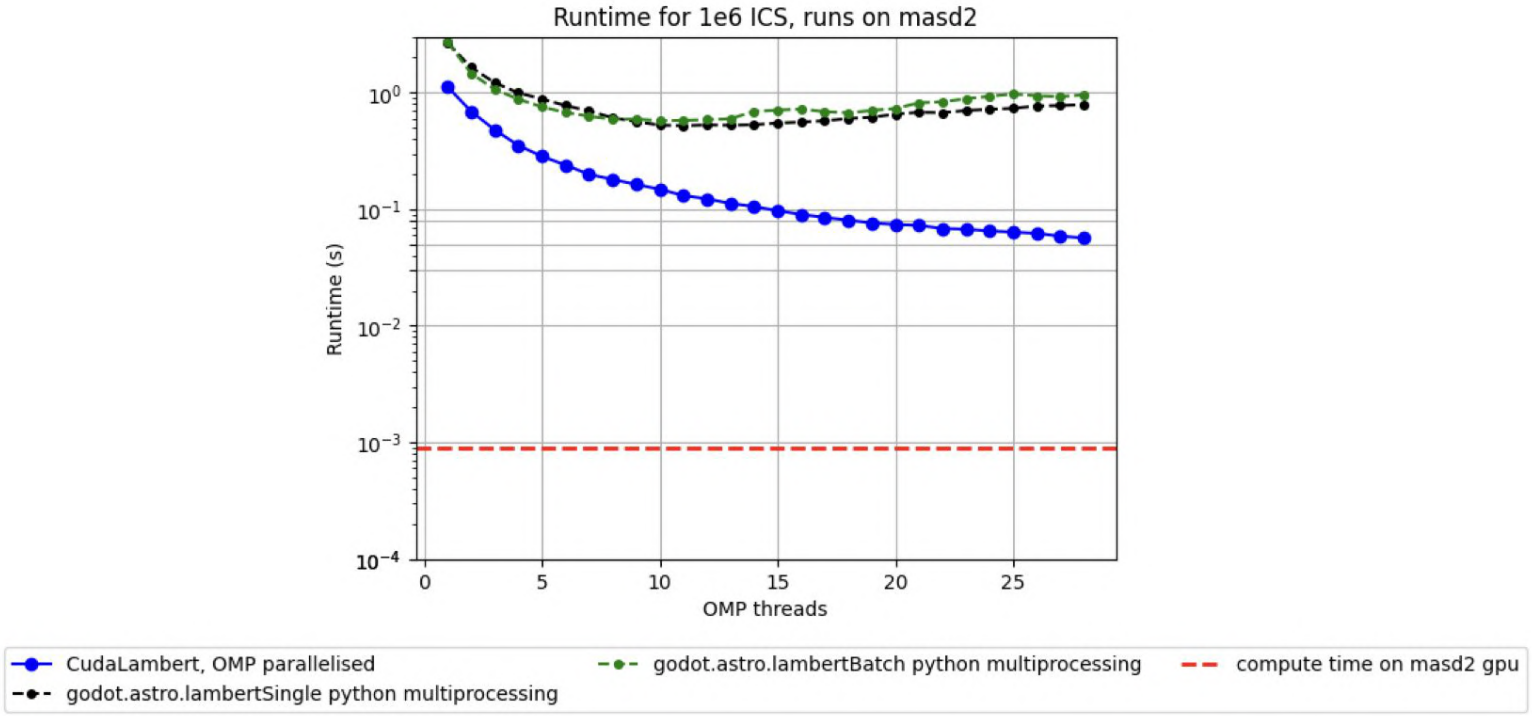


Figure A.1. CAM(L)AMBERT strong scaling test for 10^6 initial conditions on: CPU (in blue) and GPU computing time on NVIDIA V100 (red), against runtime for the standard algorithm currently in use (in black and green).

Chapter B

Additional Figures

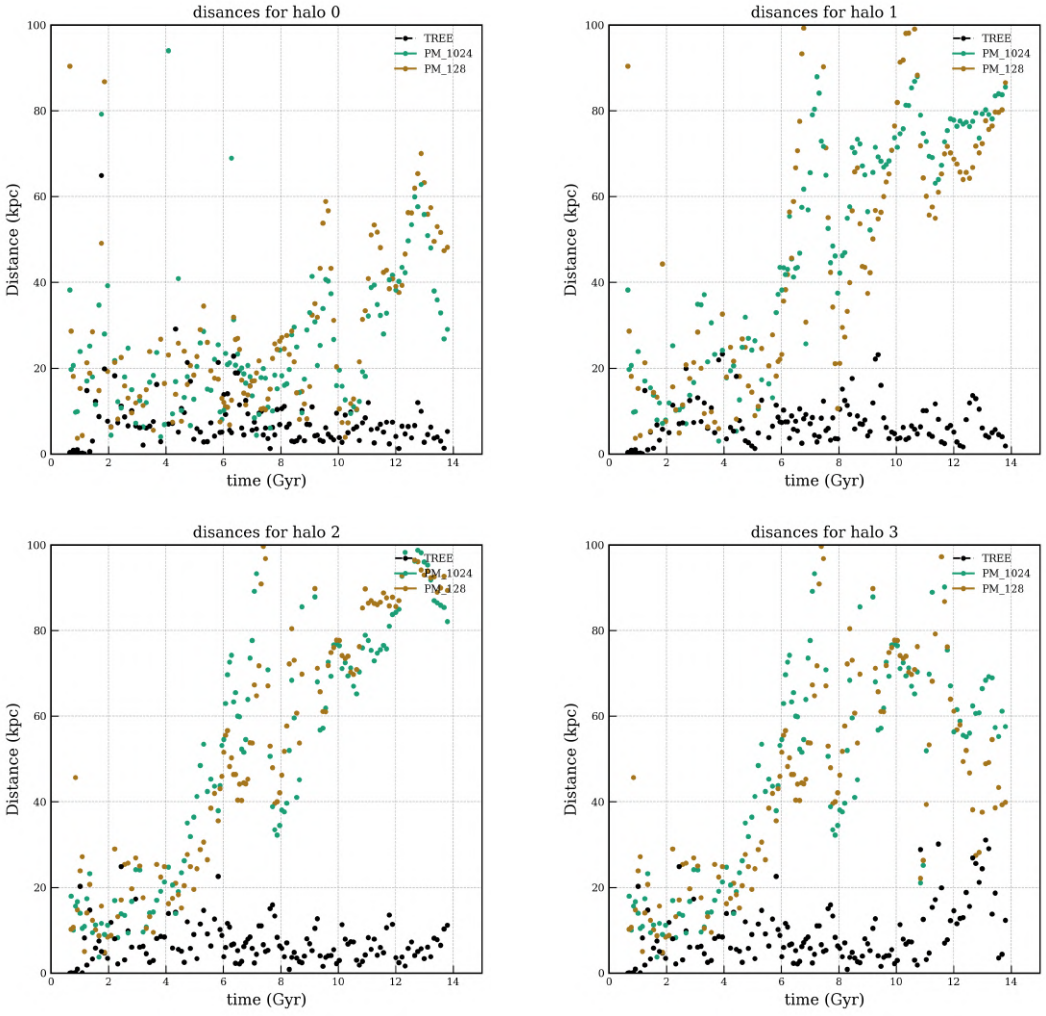


Figure B.1. Distances of structures over time in the simulation. Each panel displays the distances for one of the ten most massive haloes in the simulation. Distances are taken with respect to the tree reference run.

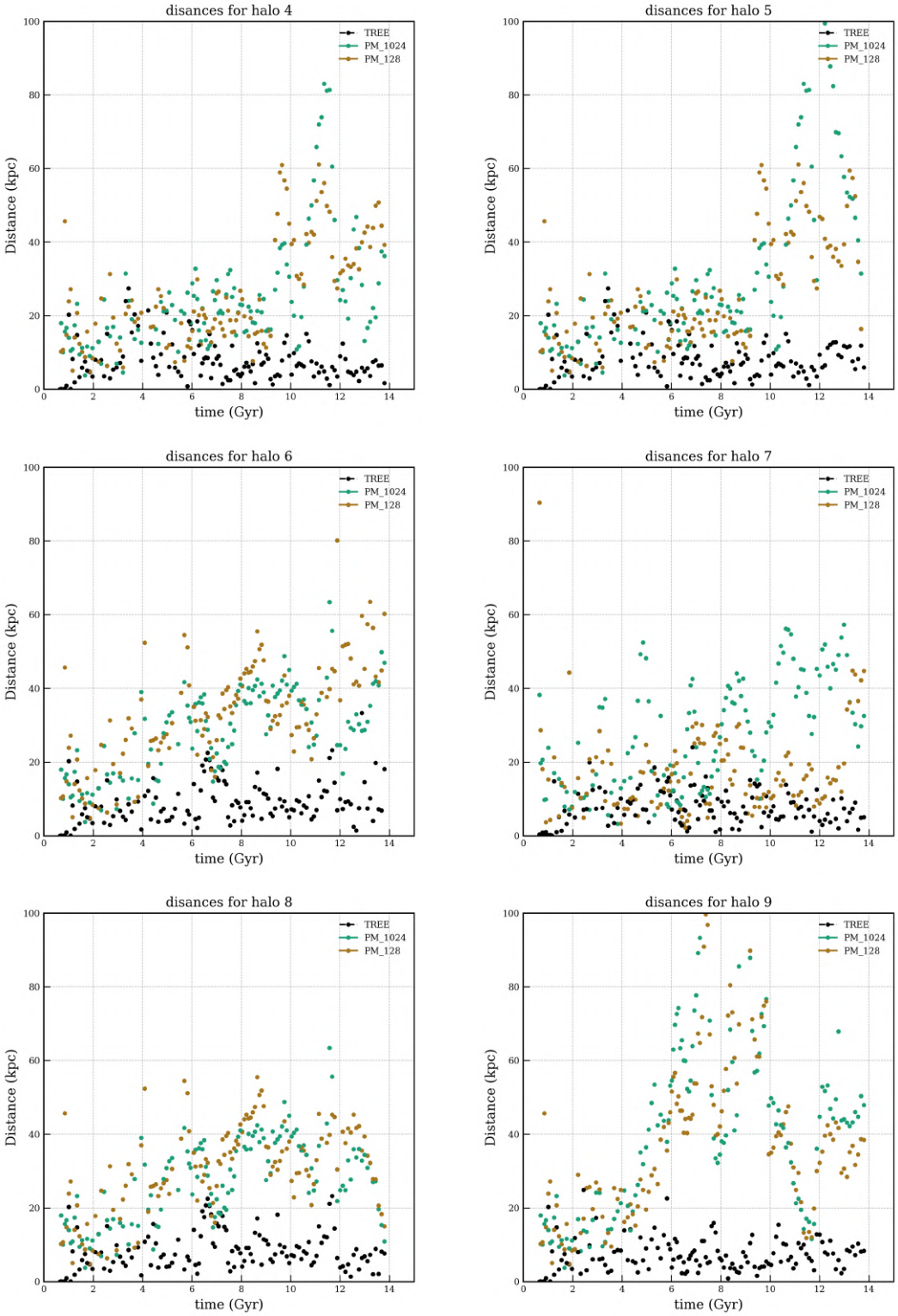


Figure B.1. Continued.

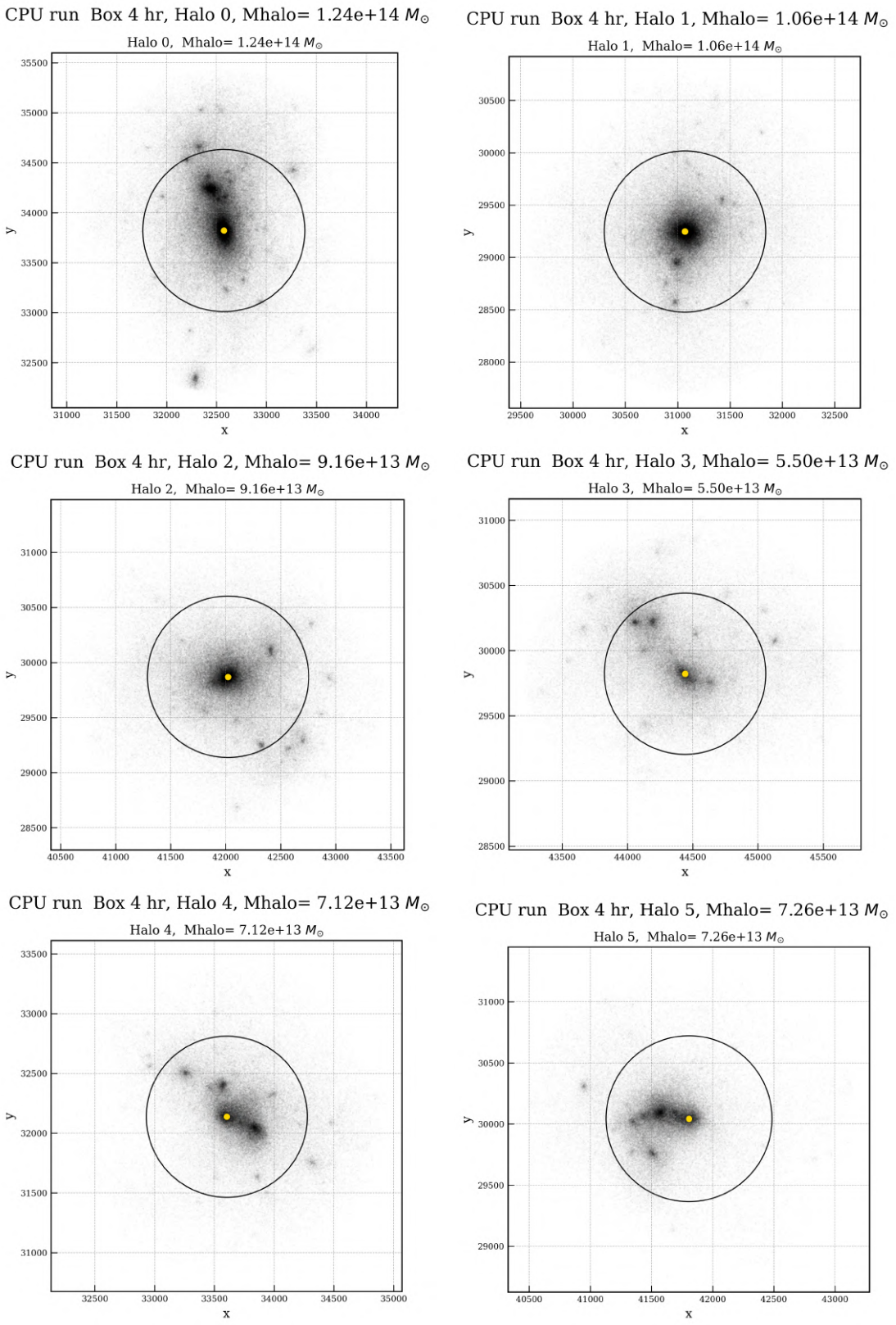


Figure B.2. The ten most massive haloes in the simulation at redshift zero from the reference tree only simulation, used for the cross-matching procedure described in Sect. B.1.

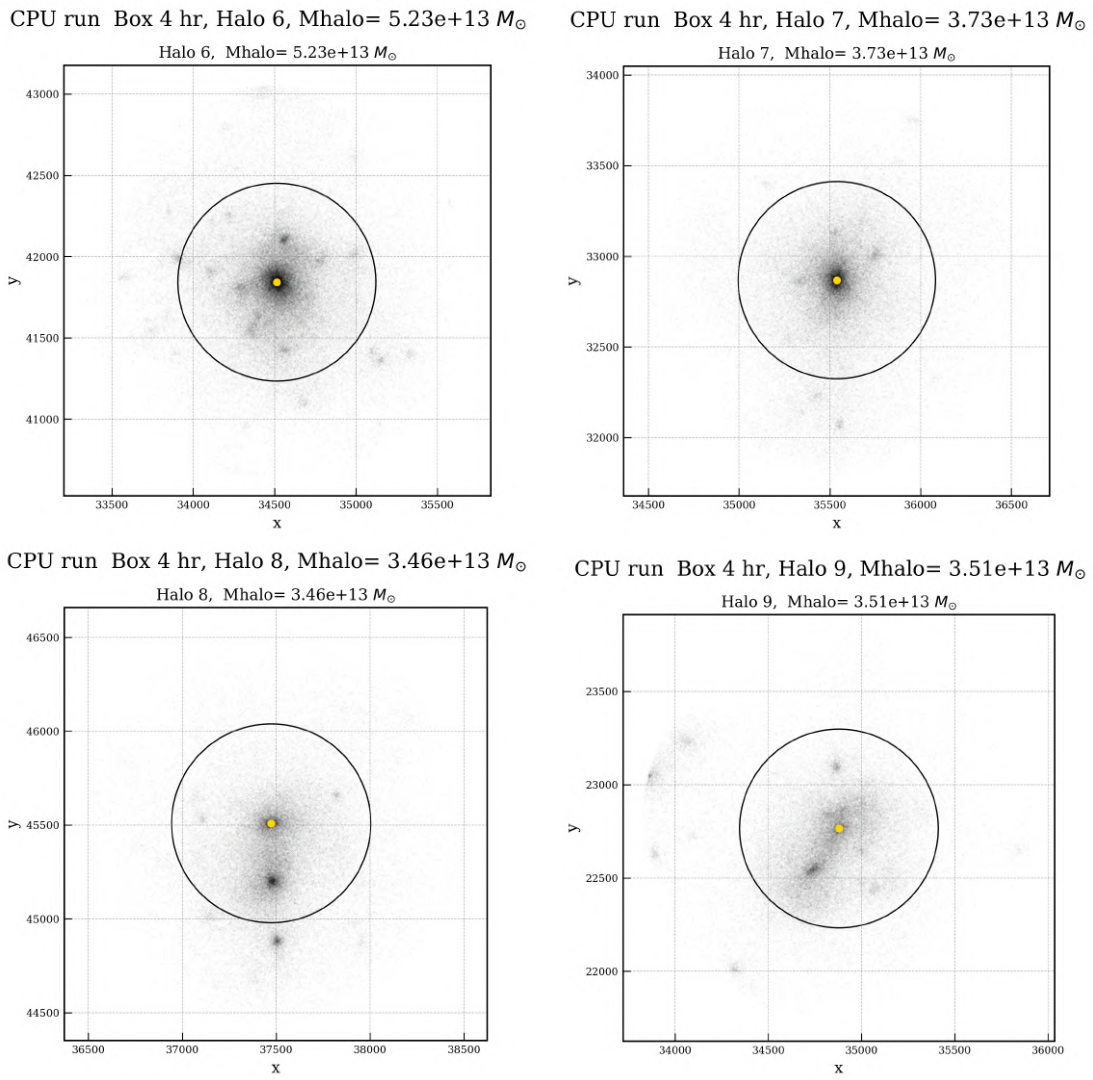


Figure B.2. Continued.

Bibliography

Aarseth, S. J. 1963, , 126, 223, doi: 10.1093/mnras/126.3.223

Adamek, J., Fiorini, B., Baldi, M., et al. 2025, Astronomy and Astrophysics, 695, doi: 10.1051/0004-6361/202452180

Ade, P. A. R., Aghanim, N., Alves, M. I. R., et al. 2014, Astronomy amp; Astrophysics, 571, A1, doi: 10.1051/0004-6361/201321529

Aghamousa, D. A., et al. 2016, The DESI Experiment Part I: Science,Targeting, and Survey Design. https://arxiv.org/abs/1611.00036

AMD. 2026a, What is HiP?, https://rocm.docs.amd.com/projects/HIP/en/latest/what_is_hip.html

—. 2026b, Porting NVIDIA CUDA code to HIP, https://rocm.docs.amd.com/projects/HIP/en/latest/how-to/hip_porting_guide.html

—. n.d., API syntax, https://rocm.docs.amd.com/projects/HIP/en/latest/reference/api_syntax.html

Angulo, R. E., & Hahn, O. 2022, Living Reviews in Computational Astrophysics, 8, doi: 10.1007/s41115-021-00013-z

Bagla, J. S., & Padmanabhan, T. 1997, Pramana, 49, 161–192, doi: 10.1007/bf02845853

Barnes, J., & Hut, P. 1986, , 324, 446, doi: 10.1038/324446a0

Battin, R. 1987, An Introduction to the Mathematics and Methods of Astrodynamics, AIAA Textbook Series (American Institute of Aeronautics and Astronautics). https://books.google.de/books?id=Y2ujAAAAIAAJ

Bennett, C. L., Larson, D., Weiland, J. L., et al. 2013, , 208, 20, doi: 10.1088/0067-0049/208/2/20

Bryan, G. L., Norman, M. L., O'Shea, B. W., et al. 2014, Astrophysical Journal, Supplement Series, 211, doi: 10.1088/0067-0049/211/2/19

Centrella, J., & Melott, A. 1983, Nature, 305, 196, doi: 10.1038/305196a0

Coles, P., & Lucchin, F. 1995, Cosmology. The origin and evolution of cosmic structure

Contributors to the Kokkos project. 2026, kokkos: the programming model, https://www.khronos.org/sycl/

Di Valentino, E., Mena, O., Pan, S., et al. 2021, Classical and Quantum Gravity, 38, 153001, doi: 10.1088/1361-6382/ac086d

Dicke, R. H., Peebles, P. J. E., Roll, P. G., & Wilkinson, D. T. 1965, , 142, 414, doi: 10.1086/148306

Dolag, K., Borgani, S., Murante, G., & Springel, V. 2009, Monthly Notices of the Royal Astronomical Society, 399, 497, doi: 10.1111/j.1365-2966.2009.15034.x

Dolag, K., Borgani, S., Schindler, S., Diaferio, A., & Bykov, A. M. 2008, doi: 10.1007/s11214-008-9316-5

Dolag, K., Remus, R.-S., Valenzuela, L. M., et al. 2025. http://arxiv.org/abs/2504.01061

Dolag et al. n.d., OpenGadget3

Doroshkevich, A. G., Kotok, E. V., Poliudov, A. N., et al. 1980, , 192, 321, doi: 10.1093/mnras/192.2.321

Fischer, M. S., Wiertel, M., Arido, C., et al. 2026, The Cosmological Simulation Code OpenGadget3 – Implementation of Self-Interacting Dark Matter. https://arxiv.org/abs/2603.10107

Fukushige, T., Ito, T., Makino, J., et al. 1991, Publications of the Astronomical Society of Japan, 43, 841, doi: 10.1093/pasj/43.6.841

GODOT Team, ESA Flight Dynamics. 2025, GODOT Software Specification Document, Software Specification Document ESA-DOPS-FD-SP-0008, European Space Agency (ESA), ESOC, Darmstadt, Germany. https://godot.io.esa.int/ssd/godot-ssd.pdf

Gooding, R. H. 1990, Celestial Mechanics and Dynamical Astronomy, 48, 145, doi: 10.1007/BF00049511

Gunn, J. E., & Gott, III, J. R. 1972, , 176, 1, doi: 10.1086/151605

Harrison, E. R. 1970, Phys. Rev. D, 1, 2726, doi: 10.1103/PhysRevD.1.2726

Hemsendorf, M., & Merritt, D. 2002, The Astrophysical Journal, 580, 606–609, doi: 10.1086/343027

Henon, M. 1973, , 24, 229

Hernández-Aguayo, C., Springel, V., Pakmor, R., et al. 2023, Monthly Notices of the Royal Astronomical Society, 524, 2556, doi: 10.1093/mnras/stad1657

Hockney, R. W., & Eastwood, J. W. 1981, Computer Simulation Using Particles (McGraw-Hill). https://ui.adsabs.harvard.edu/abs/1981csup.book.....H

IBM. 2025, What is High Performance Computing?, https://www.ibm.com/think/topics/hpc

Inno, A. F., Colombo, C., Masat, A., Bucci, L., & Renk, F. 2022, in Proceedings of the 5th International Workshop on Key Topics in Orbit Propagation Applied to Space Situational Awareness (KePASSA-22) (Logroño, Spain: Universidad de La Rioja), 150–152. https://dialnet.unirioja.es/servlet/articulo?codigo=10044104

Intel. 2021, OpenMP accelerator offload, https://www.intel.com/content/www/us/en/developer/articles/technical/openmp-accelerator-offload.html

Izzo, D. 2014, Celestial Mechanics and Dynamical Astronomy, 121, 1–15, doi: 10.1007/s10569-014-9587-y

Klypin, A. A., & Shandarin, S. F. 1983, , 204, 891, doi: 10.1093/mnras/204.3.891

Kravtsov, A. V., Klypin, A. A., & Khokhlov, A. M. 1997, ADAPTIVE REFINEMENT TREE-A NEW HIGH-RESOLUTION N-BODY CODE FOR COSMOLOGICAL SIMULATIONS, Tech. rep.

Lacey, C., & Cole, S. 1993, , 262, 627, doi: 10.1093/mnras/262.3.627

Lancaster, E., Blanchard, R., Aeronautics, U. S. N., Administration, S., & Center, G. S. F. 1969, A Unified Form of Lambert's Theorem, NASA technical note (National Aeronautics and Space Administration). https://books.google.de/books?id=lTj2Wfu84MsC

Leclercq, F. 2020, Evolution of cosmological simulations over the last 50 years, http://florent-leclercq.eu/blog.php?page=2,

Makino, J., Taiji, M., Ebisuzaki, T., & Sugimoto, D. 1990, Computer Physics Communications, 60, 187, doi: 10.1016/0010-4655(90)90003-J

Masat, A., Colombo, C., & Boutonnet, A. 2021, Surfing Chaotic Perturbations in Interplanetary Multi-Flyby Trajectories: Augmented Picard-Chebyshev Integration for Parallel and GPU Computing Architectures, doi: 10.2514/6.2022-1275

Mather, J. C., Cheng, E. S., Shafer, R. A., et al. 1990, Astrophysical Journal; (USA), 354, doi: 10.1086/185717

Mellier, Y., Abdurro'uf, Acevedo Barroso, J. A., et al. 2025, Astronomy amp; Astrophysics, 697, A1, doi: 10.1051/0004-6361/202450810

Meszaros, P. 1974, , 37, 225

Miller, R. H. 1983, , 270, 390, doi: 10.1086/161133

Miyoshi, K., & Kihara, T. 1975, , 27, 333, doi: 10.1093/pasj/27.2.333

Moore, G. 2006, Solid-State Circuits Newsletter, IEEE, 11, 33 , doi: 10.1109/N-SSC.2006.4785860

Moscardini, L., & Dolag, K. 2011, Cosmology with Numerical Simulations, ed. S. Matarrese, M. Colpi, V. Gorini, & U. Moschella (Dordrecht: Springer Netherlands), 217–237, doi: 10.1007/978-90-481-8685-3_4

Navarro, J. F., Frenk, C. S., & White, S. D. M. 1997, , 490, 493, doi: 10.1086/304888

Norman, M. L. 2004, The Impact of AMR in Numerical Astrophysics and Cosmology, Tech. rep.

Nvidia. 2025, CUDA Programming Guide, https://docs.nvidia.com/cuda/archive/13.1.0/cuda-programming-guide/01-introduction/introduction.html

—. 2026, OpenACC getting started guide, https://docs.nvidia.com/hpc-sdk/compilers/openacc-gs/#supporting-documentation-and-examples

OpenMP. 2026, OpenMP Specifications, https://www.openmp.org/specifications/

Peebles, P. J. E. 1971, Physical cosmology. https://ui.adsabs.harvard.edu/abs/1971phco.book.....P

Penzias, A. A., & Wilson, R. W. 1965, , 142, 419, doi: 10.1086/148307

Perivolaropoulos, L., & Skara, F. 2022, , 95, 101659, doi: 10.1016/j.newar.2022.101659

Perlmutter, S., Aldering, G., Goldhaber, G., et al. 1999, The Astrophysical Journal, 517, 565–586, doi: 10.1086/307221

Planck Collaboration, Aghanim, N., Akrami, Y., et al. 2020, , 641, A6, doi: 10.1051/0004-6361/201833910

Power, C., Navarro, J. F., Jenkins, A., et al. 2003, The inner structure of CDM haloes-I. A numerical convergence study, Tech. rep. https://academic.oup.com/mnras/article/338/1/14/1094443

Press, W. H., & Schechter, P. 1974, , 187, 425, doi: 10.1086/152650

Ragagnin. n.d., g3read, https://github.com/aragagnin/g3read

Ragagnin, A., Dolag, K., Wagner, M., et al. 2020, doi: 10.3233/APC200043

Schneider, A., Teyssier, R., Potter, D., et al. 2016, Journal of Cosmology and Astroparticle Physics, 2016, 047–047, doi: 10.1088/1475-7516/2016/04/047

Sembolini, F., Yepes, G., Pearce, F. R., et al. 2016, Monthly Notices of the Royal Astronomical Society, 457, 4063, doi: 10.1093/mnras/stw250

Springel, V. 2005, The cosmological simulation code GADGET-2, doi: 10.1111/j.1365-2966.2005.09655.x

Springel, V., Yoshida, N., & White, S. D. M. 2001, GADGET: A code for collisionless and gasdynamical cosmological simulations, Tech. rep. http://www.mpa-garching.mpg.de/gadget

Teyssier, R. 2002, Astronomy and Astrophysics, 385, 337, doi: 10.1051/0004-6361:20011817

The Khronos Group. 2026, sycl: C++ Programming for Heterogeneous Parallel Computing, https://www.khronos.org/sycl/

The Open MPI Community. 2026, MPI Documentation, https://docs.open-mpi.org/en/v5.0.x/

Top500. 2025a, Highlights - November 2025, https://top500.org/lists/top500/2025/11/highs/

—. 2025b, List Statistics for Accelerators, https://top500.org/statistics/list/

Vogelsberger, M., Marinacci, F., Torrey, P., & Puchwein, E. 2020, Nature Reviews Physics, 2, 42, doi: 10.1038/s42254-019-0127-2

von Hoerner, S. 1960, , 50, 184

Zeldovich, Y. B. 1972, Monthly Notices of the Royal Astronomical Society, 160, 1P, doi: 10.1093/mnras/160.1.1P

Zwart, S. F. P., Belleman, R. G., & Geldof, P. M. 2007, New Astronomy, 12, 641, doi: 10.1016/j.newast.2007.05.004