UNIVERSITÀ DEGLI STUDI DI TRIESTE

UNIVERSITÀ DEGLI STUDI DI TRIESTE

XXXVIII CICLO DEL DOTTORATO DI RICERCA IN

FISICA

Finanziato dall'Unione europea - NextGenerationEU
Funded by the European Union - NextGenerationEU

Fast cosmological simulations with GPU-enabled PINOCCHIO: from dark matter halos to cosmic voids

Settore scientifico-disciplinare: ASTRONOMIA E ASTROFISICA

DOTTORANDO
Marius Daniel Lepinzan

COORDINATORE
PROF. Angelo Bassi

SUPERVISORE DI TESI
PROF. Pierluigi Monaco

CO-SUPERVISORE DI TESI
DOTT. Tiago Castro

CO-SUPERVISORE DI TESI
DOTT. Luca Tornatore

ANNO ACCADEMICO 2024/2025

Finanziato dall'Unione europea NextGenerationEU | Ministero dell'Università e della Ricerca | Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA | UNIVERSITÀ DEGLI STUDI DI TRIESTE

To Mario Nonino, who left us far too soon, yet more than anyone taught me how to look at the sky with the right eyes

Acknowledgements

I am deeply grateful to my parents, who have supported me since my very first steps in the academic world, not only financially but, above all, emotionally. They have always stood by my side, in both joyful and difficult moments, and have never failed to offer a shoulder to lean on when needed.

I would like to thank my supervisors, Pierluigi Monaco, Tiago Castro, and Luca Tornatore, who guided me throughout the development of this project and, at the same time, taught me the value of patience and independence.

I am also grateful to my friends, first and foremost, and collaborators at the Trieste Astronomical Observatory: Erik, Daniele, Thomas G., Thomas V., Federico, Roberta, Samuele, and Tiago for the fruitful discussions and, above all, for the laughter and the happy moments we shared that made this journey lighter and more enjoyable.

I also thank all the PhD students and postdoctoral researchers with whom I have had the pleasure of interacting, for making these years both enriching and stimulating, with special thanks to my office mates, Alice, Luca (chef), Roberto, Luca, and Lapo.

Last but not least, I would like to thank my partner and companion in life, Alessandra, who has been by my side from the very beginning, for her constant support, her great patience during stressful times, and for the scientific insight that helped refine this work.

Abstract

Large-scale Stage IV galaxy surveys, including Euclid, aim to map the distribution of matter across unprecedented cosmic volumes, enabling high-precision tests of the standard ΛCDM cosmological model and its possible extensions. Meeting the scientific goals of these surveys, which are limited by systematic effects rather than statistical ones, requires large ensembles of accurate mock catalogs, thereby posing significant computational challenges. Full N-body simulations, while highly accurate, become prohibitively expensive when thousands of realizations are needed, motivating the increased interest in fast approximate methods that are both computationally efficient and scientifically reliable.

This thesis focuses on the PINOCCHIO *code, a Lagrangian Perturbation Theory-based approach for efficiently generating dark matter halo catalogs from an initial density field, and addresses two complementary research directions. First, we modernize and optimize key components of* PINOCCHIO *for execution on heterogeneous high-performance computing platforms. In particular, we port two of the main computational modules of the code to Graphics Processing Units using primarily OpenMP target directives, achieving portable performance across NVIDIA and AMD architectures. Detailed performance and roofline analyses demonstrate significant speed-ups relative to CPU-only implementations, achieving up to a $\times 8$ speed up depending on the system. Energy-to-solution measurements, performed using a newly developed parallel version of the Power Measurement Toolkit, show reductions of up to a factor of $\times 8$, relative to CPU-only executions, corresponding to an overall efficiency improvement of up to $\times 64$. These results establish GPU offloading as an effective strategy for improving both performance and sustainability of large-scale cosmological simulation campaigns in the exascale era.*

Second, we present the first comprehensive validation of PINOCCHIO *in underdense environments by assessing its ability to reproduce cosmic void statistics. Using matched initial conditions and number-density–matched halo catalogs, we compare* PINOCCHIO *with the full N-body code* OpenGADGET3 *across multiple redshifts and resolutions. We analyze several void summary statistics, including the void size function, void ellipticity function, core density function, and radial density profiles. Across all metrics,* PINOCCHIO

shows good agreement with the N-body results, with differences typically remaining within the $\pm 10\%$ *level and below the* 2σ *significance threshold, and with no evidence of significant systematic biases. The level of agreement is comparable to that previously achieved for halo statistics in overdense regions, confirming the robustness of* PINOCCHIO *in the quasi-linear regime.*

Together, these results show that PINOCCHIO *provides a computationally efficient and accurate framework for generating large ensembles of cosmological simulations. The code has been used to produce more than 4500 realizations designed to reproduce the Euclid spectroscopic galaxy sample, by populating dark matter halos with galaxies through a halo occupation distribution model calibrated on the Euclid Flagship simulation, supporting the galaxy clustering analyses of the Euclid Data Release 1. In addition, the validation of* PINOCCHIO *in underdense environments demonstrates its suitability for void-based analyses, extending its applicability beyond traditional halo statistics.*

Contents

Introduction

Modern cosmology has entered an era of precision measurements driven by large-scale galaxy surveys such as *Euclid* (Mellier et al. 2024), the Dark Energy Survey (DES, Dark Energy Survey Collaboration et al. 2016), the Dark Energy Spectroscopic Instrument (DESI, Levi et al. 2013), the Large Synoptic Survey Telescope (LSST, LSST Science Collaboration et al. 2009), the Nancy Grace Roman Space Telescope (Spergel et al. 2015), the Spectro-Photometer for the History of the universe, Epoch of Reionization, and Ices Explorer (SPHEREx, Bock et al. 2025), and the SKA Observatory (SKAO, Bacon et al. 2020). These missions aim to map the distribution of matter across an unprecedented volume of the Universe, enabling high-precision tests of the ΛCDM model, dark energy (DE), and theories of gravity (Laureijs et al. 2011; Weinberg et al. 2013; Amendola et al. 2018). Meeting these scientific objectives requires theoretical predictions of comparable precision, and, crucially, an increasingly rigorous control of systematic effects. As upcoming surveys will no longer be limited by statistical uncertainties but by systematics, understanding and mitigating these effects becomes essential. This necessity places strong demands on large ensembles of cosmological simulations, which are used to quantify and assess the impact of these effects on the extracted cosmological information.

Running these simulations at the volume and resolution demanded by upcoming surveys poses two major challenges. First, full N-body simulations, while highly accurate, remain computationally expensive and often require millions of CPU-hours for a single high resolution realization (see for instance Castander et al. 2025). Producing the large number of mock catalogs needed for survey analyses becomes impractical under these constraints. Second, the transition toward pre-exascale and exascale computing platforms introduces new architectural and sustainability challenges. Efficient use of heterogeneous systems, including Graphic processing unit (GPUs), is now essential not only for performance but also for long-term energy sustainability (see for instance Lacopo et al. 2025b; Shukla et al. 2025).

Approximate methods such as `PINOCCHIO` (Monaco et al. 2002a; Monaco 2016) offer an attractive solution to the growing computational demands. By exploiting Lagrangian

perturbation theory, PINOCCHIO generates accurate dark matter halo catalogs at a fraction of the cost of full N-body simulations. This makes it particularly suitable for producing the large suites of mocks required by current and future surveys.

However, meeting these demands requires not only scientific accuracy but also the ability to exploit modern High Performance Computing (HPC) platforms efficiently. This includes accelerating the most computationally intensive components through GPU offloading and assessing both performance and energy efficiency: two essential ingredients for sustainable large-volume simulation campaigns in the exascale era. At the same time, broadening its scientific validation remains crucial. While PINOCCHIO has been extensively validated for halo statistics, such as the halo mass function, halo bias, and two-point statistics, its performance in reproducing other components of the cosmic web remains largely unexplored.

Among these components, cosmic voids are emerging as powerful and complementary cosmological probes (Hamaus et al. 2016; Pisani et al. 2019). As the most underdense structures in the Universe, voids are sensitive to the expansion history, gravitational dynamics, and the influence of massive neutrinos and modified gravity. Their structure and evolution is largely governed by linear and quasi-linear dynamics, suggesting that approximate methods like PINOCCHIO may be particularly effective in tracing them. Yet, a systematic assessment of the ability of PINOCCHIO in reproducing void statistics has been missing until now.

This thesis addresses these topics through two complementary lines of research:

- **Modernization, GPU porting, and energy-efficiency assessment of** PINOCCHIO: We offload to GPUs two of the three main computational modules of PINOCCHIO, namely, the collapse times and the FFT-based calculations. We perform detailed performance and roofline analyses across multiple GPU architectures, and we quantify the impact of GPU offloading on both runtime and energy consumption using a new parallel implementation of the Power Measurement Toolkit (PMT). Our goal is to enhance the performance, portability, and long-term sustainability of PINOCCHIO for future large-volume simulation campaigns.

- **Cosmological validation of** PINOCCHIO **through void statistics:** Using matched initial conditions the full N-body code OpenGADGET3 as reference, we evaluate how well PINOCCHIO reproduces key void summary statistics, including the void size function, void ellipticity function, core density function, and radial density profiles. This provides the first comprehensive assessment of the accuracy of PINOCCHIO in underdense regions, where Lagrangian dynamics are expected to perform best.

In Chapter 1, we introduce the numerical tools used to model the formation and evolution of cosmic structure. We review the role of cosmological N-body simulations in connecting theoretical predictions with observations, discuss the requirements of upcoming surveys such as *Euclid*, and motivate the need for fast and energy efficient approximate methods, setting the stage for the use of `PINOCCHIO`.

In Chapter 2, we present the theoretical and algorithmic foundations of the `PINOCCHIO` code. We summarize the physical motivation, the Lagrangian perturbation theory (LPT) framework on which it is built, the ellipsoidal collapse model, and the main components of the code pipeline, concluding with an overview of its cosmological applications and validation against N-body.

In Chapter 3, we describe the modernization and GPU acceleration of key `PINOCCHIO` modules, focusing on the collapse time calculation and the development of custom GPU-native interpolation routines. We outline the rationale behind the porting strategy and the use of OpenMP for offloading, and we present the performance results obtained on both NVIDIA- and AMD-based platforms.

In Chapter 4, we introduce the PMT library and its new parallel implementation for MPI-based applications, and we report the energy-efficiency analysis of the GPU-accelerated collapse time module on heterogeneous HPC systems.

In Chapter 5, we present ongoing technical developments, including preliminary results from the heFFTe-based FFT offloading and exploratory tests of a new `PINOCCHIO` halo finding strategy based on a 3D grid clustering algorithm.

In Chapter 6, we explore a novel scientific application of `PINOCCHIO` by assessing its accuracy in predicting cosmic void statistics relative to a full N-body simulation. After introducing the void finder and the void summary statistics considered, we analyze number density-matched halo catalogs and compare void properties, across different simulation resolutions and redshifts, assessing the robustness of `PINOCCHIO` in underdense environments.

Finally, in Chapter 7 we summarize the conclusions of both research directions and outline future work for performance optimization and scientific applications.

In Appendix A, we present the 2D grid–based version of the clustering algorithm used in the halo finding study, including tests on both simulated and real *Euclid* images for deblending applications.

The technical terminology and hardware-specific concepts used throughout this work are defined in Appendix B.

Together, these studies demonstrate both the computational readiness of `PINOCCHIO` for the exascale era and its scientific reliability for void-based cosmology, strengthening

its role as a fast, accurate, and sustainable tool for next-generation large-scale structure surveys.

Chapter 1

Numerical tools for precision cosmology

This Chapter provides an overview of the numerical tools used to model the formation and evolution of cosmic structure. We begin by introducing cosmological N-body simulations and their central role in connecting theoretical predictions with observations. We then focus on the requirements of the *Euclid* mission, presenting the *Euclid* Flagship simulation as a representative example of state-of-the-art large-scale computations. Because upcoming surveys demand large ensembles of high-quality mock catalogs, we discuss the computational cost of full N-body simulations and the resulting need for fast approximate methods. Finally, we examine the computational challenges that motivate the development of fast and energy-efficient simulation suited for modern High Performance Computing (HPC) platforms, providing the context for the optimizations and applications of `PINOCCHIO` code discussed later in this thesis.

1.1 Numerical simulation

Understanding the formation and evolution of cosmic structures is a central goal of modern cosmology. The Large-scale structure (LSS) arise from the gravitational collapse and hierarchical merging of small primordial perturbations (Peebles 1980; Coles & Lucchin 1995; Mo et al. 2010) imprinted in the very early Universe and observed today as temperature anisotropies in the cosmic microwave background (CMB, Ade et al. 2016). While linear perturbation theory provides accurate predictions for the evolution of these perturbations at early times and on large scales (Bernardeau et al. 2002), the dynamics become rapidly non-linear as the perturbations grow and collapse. In this non-linear regime, analytical approaches break down due to the intrinsically non-local and non-linear nature of gravitational interactions. To overcome this limitation, the evolution of structure must be followed numerically. The most direct approach is to discretize the matter distribution in a

cosmological volume into a large number of fluid elements (N) and compute their mutual gravitational interactions.

Numerical N-body and hydrodynamical simulations offer a controlled environment in which theoretical models can be tested and compared with observations (Navarro et al. 1997; Jenkins et al. 1998; de Theije et al. 1998). N-body simulations follow the gravitational evolution of matter, dominated by the dark matter (DM) component, predicting the distribution of DM halos, their abundances, clustering, and the statistical properties of large-scale structure across cosmic time. To model additional physical processes, such as gas cooling, star formation, chemical enrichment, and stellar and AGN feedback, hydrodynamical simulations add a treatment of baryonic physics on top of the gravitational dynamics (Springel et al. 2005; Borgani & Kravtsov 2011; Vogelsberger et al. 2014). These simulations aim to reproduce detailed galaxy and interstellar medium properties, although at a substantially higher computational cost. As a result, N-body and hydrodynamical simulations together provide a comprehensive theoretical framework linking the growth of cosmic structure to the observable galaxy distribution.

The state of the art in cosmological N-body simulations continues to advance as computational resources and algorithms improve. One of the early milestones in this direction was the Millennium simulation (Springel 2005), which followed 10^{10} dark matter particles in a $500\,h^{-1}\mathrm{Mpc}^3$ volume and provided a foundation for semi-analytic galaxy formation models and large-scale structure studies for more than a decade. Recent efforts have achieved unprecedented volume, mass resolution, and physical realism. For example, the AbacusSummit suite (Maksimova et al. 2021), consists of more than 150 simulations of $2\,h^{-1}\mathrm{Gpc}^3$ volumes, each evolved with over 10^{12} particles. Other major large-scale simulation programs include Quijote (Villaescusa-Navarro et al. 2020), a suite of more than 44.000 N-body simulations covering over 7.000 cosmological models for emulator and machine learning training; OuterRim (Heitmann et al. 2019), which evolves $> 10^{12}$ particles in boxes of side length 4.2 Gpc; and Uchuu (Ishiyama et al. 2021), whose largest volume contains over 10^{12} particles in a $2\,h^{-1}\mathrm{Gpc}^3$ box. Hydrodynamical simulation campaigns such as IllustrisTNG (Nelson et al. 2019) complement these by modelling the formation of galaxies and the thermodynamics of baryons within volumes of 50-300 Mpc with up to 10^{10} resolution elements.

However, accuracy alone is not sufficient. Modern galaxy surveys such as *Euclid*, DES, DESI, LSST, Nancy Grace Roman Space Telescope, SPHEREx, and SKAO probe extremely large cosmological volumes and measure clustering statistics with small statistical uncertainties. To correctly model and interpret these observations, it is necessary to generate not only a single simulated Universe, but large ensembles of independent realizations. These are essential for: (i) estimating covariance matrices for robust uncertainty quan-

tification; (ii) validating end-to-end analysis pipelines; (iii) assessing systematic effects and model assumptions; and (iv) exploring cosmological parameter spaces, including extensions beyond ΛCDM.

Among these forthcoming surveys, *Euclid* stands out due to the unprecedented combination of survey volume, depth, and redshift coverage it will achieve. The scientific return of the mission therefore relies critically on the availability of accurate and statistically representative mock catalogs, motivating the development and use of large-scale cosmological simulations. In the following Section, we provide an overview of the *Euclid* mission and the *Euclid* Flagship simulation, which constitutes the main reference dataset for its galaxy clustering analyses.

1.2 The *Euclid* mission

Euclid (Mellier et al. 2024) is a medium-class mission in the Cosmic Vision 2015–2025 programme of the European Space Agency (ESA), launched in 2023 and designed to investigate the origin of cosmic acceleration and the nature of dark energy (DE), DM, and gravity on cosmological scales. The mission will map the three-dimensional large-scale distribution of galaxies and weak gravitational lensing (WL) distortions over approximately 14 000 deg^2 of the extragalactic sky out to redshift $z \sim 2$ through its Wide Survey, complemented by a Deep Survey covering ~ 50 deg^2 to greater imaging and spectroscopic depth.

The spacecraft hosts two scientific instruments. The Visible Camera (VIS, Cropper et al. 2025) provides high-resolution optical imaging for weak-lensing shape measurements of galaxies in the redshift range $0.1 \lesssim z \lesssim 1.5$. VIS delivers the largest space-based imaging survey to date, achieving a depth of $m_{\rm AB} \geq 24.5$ at signal-to-noise ratio $\mathrm{S/N} \geq 10$ in a single broad optical band over the Wide Survey (EWS), and reaching $m_{\rm AB} \geq 26.4$ in the Deep Survey (EDS).

The Near-Infrared Spectrometer and Photometer (NISP, Jahnke et al. 2025) provides near-infrared photometry and slitless spectroscopy in the wavelength range 0.95–2.02 μm. In photometric mode, NISP obtains imaging in the Y, J, and H bands, reaching a limiting depth of $m_{\rm AB} \simeq 24.5$, at $\mathrm{S/N} = 5$, for point sources with individual exposure times of approximately 100 s. In spectroscopic mode, NISP performs slitless grism spectroscopy with resolving power $R \gtrsim 450$, achieving a point-source sensitivity corresponding to a $\mathrm{S/N} \approx 3.5$ detection of an emission line with flux $\sim 2 \times 10^{-16}$ erg s^{-1} cm^{-2}.

The combination of galaxy shape measurements from VIS and spectroscopic redshifts from NISP enables precise constraints from both WL and galaxy clustering (GC), the two

main cosmological probes of the *Euclid* mission.

Among these, GC in particular encodes a rich set of observables that probe both the expansion history of the Universe and the growth of cosmic structures. *Euclid* will measure the galaxy power spectrum and the two-point correlation function (2PCF), exploiting baryon acoustic oscillations (BAO), redshift-space distortions (RSD), and full-shape clustering analyses to constrain the nature of DE, test deviations from the ΛCDM model, and investigate possible time evolution of the DE equation of state (Mellier et al. 2024). In this context, *Euclid* will operate in synergy and competition with ground-based spectroscopic surveys such as DESI, which target complementary redshift ranges and galaxy populations. Recent DESI results have further highlighted the importance of precise galaxy clustering measurements by providing increasingly tight constraints on DE models and motivating renewed interest in possible departures from a cosmological constant, including scenarios with a time-evolving DE equation of state (Adame et al. 2025).

To interpret the survey data and extract cosmological information, a large ensemble of realistic mock catalogs is required. These must reproduce the statistical properties of the galaxy distribution, observational systematics, and the survey selection function. For this purpose, the *Euclid* Collaboration has developed the *Euclid* Flagship simulation, which serves as the reference framework for pipeline validation and performance assessment.

1.2.1 The *Euclid* Flagship simulation

The *Euclid* Flagship 1 (FS1) simulation is based on the `PKDGRAV3` (Potter et al. 2017) code, a highly efficient GPU-accelerated N-body solver specifically designed for large cosmological simulations. `PKDGRAV3` combines the fast multipole method (FMM, Greengard & Rokhlin 1987) with individual, adaptive particle time stepping.

Thanks to its low memory footprint and optimized GPU implementation, `PKDGRAV3` has been used to evolve cosmological volumes with up to several trillion (10^{12}) particles on supercomputers such as Piz Daint (CSCS)[1] and Titan (ORNL)[2], achieving runs of two trillion particles down to redshift $z = 0$ within roughly 350000 node hours ($\sim$ 80 hours of wall time) and demonstrating near-ideal scaling up to 4000 of GPU-accelerated nodes. This computational capability forms the foundation for the *Euclid* FS1, enabling the generation of the large high-resolution dataset required for the mission's cosmological analyses.

To reproduce the volume probed by the *Euclid* Wide Survey, the simulation constructs a full-sky past light cone during the run by periodically replicating the simulation box whenever needed to fill the observational footprint. The halo catalogue is then generated

[1] https://www.cscs.ch/computers/piz-daint
[2] https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/

from the particle distribution using the ROCKSTAR halo finder (Behroozi et al. 2013), and is publicly available through the CosmoHub[3] data platform.

A second-generation run, known as Flagship 2 (FS2), was completed in 2020 to extend the capabilities of the original simulation (Castander et al. 2025). FS2 increased the mass resolution by a factor of two, reaching a particle mass of $m_{\rm p} = 10^9\,h^{-1}M_\odot$, enabling the modelling of galaxies approximately one magnitude fainter at all redshifts. The light-cone volume was also extended from $z = 2.3$ in FS1 to $z = 3$, improving completeness for the galaxy populations expected to be observed by *Euclid*. In addition, the procedure for assigning spectral energy distributions was revised to better match observed photometric properties.

Because it is computationally infeasible to simulate such an enormous cosmological volume (box size of 3780 h^{-1} Mpc) with hydrodynamical methods, the creation of the Flagship mock catalog, as shown in Castander et al. (2025), adopt a hybrid strategy. A large DM-only N-body run is first performed, after which halos are identified using the Rockstar halo finder. Galaxies are then introduced by populating these haloes with an empirical halo occupation distribution (HOD, Cooray & Sheth 2002) and abundance-matching (AM, Tasitsiomi et al. 2004) scheme calibrated for *Euclid*. This approach assigns a wide range of galaxy properties, including multi-band luminosities and fluxes, three-dimensional positions and velocities, redshifts, spectral energy distributions, galaxy shapes and sizes, stellar masses, star formation rates, metallicities, emission-line fluxes, and lensing-related quantities, enabling the construction of mock galaxy catalogs suitable for the mission's GC and WL analyses.

1.2.2 Approximate methods for large mock catalogs

Modern large-scale structure analyses require not only a single high-fidelity simulation but large ensembles of independent mock catalogs, primarily for estimating covariance matrices (Hartlap et al. 2007; Taylor et al. 2013; Dodelson & Schneider 2013), validating analysis pipelines (Manera et al. 2012; Krause et al. 2017; Abbott et al. 2018), and assessing systematic effects (Norberg et al. 2009; Payerne et al. 2023; Fumagalli et al. 2024, 2025b). Numerical covariance estimates become reliable only when a large number of independent realizations, typically thousands, are available since the statistical noise affecting the covariance matrix decreases as the number of simulations increases. This requirement becomes even more stringent when the covariance depends on cosmological parameters. Producing such large samples with full N-body simulations like the *Euclid* Flagship is therefore computationally impossible, as each realization demands millions of core hours.

[3]https://cosmohub.pic.es/login

These considerations make fast approximate methods, based on perturbative or semi-analytical approaches, indispensable for generating the large mock suites needed for precision cosmology (Sahni & Coles 1995; Monaco 2016; Lippich et al. 2018; Colavincenzo et al. 2019). A wide variety of such methods exist, ranging from perturbation-theory-based solvers such as `COLA` (Tassev et al. 2013) and `FastPM` (Feng et al. 2016), to stochastic and lognormal field generators like `FLASK` (Xavier et al. 2016), and hybrid bias-based approaches such as `PATCHY` (Kitaura et al. 2014). These frameworks trade exact small-scale accuracy for computational efficiency, enabling the production of thousands of mock realizations tailored to the needs of modern surveys.

In this context, the `PINOCCHIO` code (Monaco et al. 2002a, 2013; Munari et al. 2017) based on Lagrangian Perturbation Theory (LPT) provides an attractive compromise, offering orders-of-magnitude speed-ups over full N-body simulations while retaining sufficient physical fidelity to model halo formation, merger histories, and large-scale clustering. A detailed description of the code is provided in Chapter 2.

1.3 Efficiency requirements for Next-Generation cosmological simulations

The growing scale of simulations in computational cosmology (Section 1) demands increasingly efficient execution to keep pace with scientific and observational progress. Modern HPC platforms are rapidly evolving toward exascale architectures, where effective use of heterogeneous nodes, particularly those equipped with Graphics Processing Units (GPUs), has become essential (Taffoni et al. 2024; Shukla et al. 2025).

Leveraging these capabilities is now imperative for large-scale cosmological simulations, many of which must run on the supercomputers currently ranked in the Top500, HPCG500, and Green500 lists[4]. The Top500 list ranks the HPC systems according to their rank in the High Performance Linpack (HPL), a benchmark which solves linear systems of equations with an exceptional level of optimizations. However, the HPL is poorly representative for actual scientific simulations, as the impact of memory transfer is largely ignored. To address this limitation, a new benchmark named High Performance Conjugate Gradients (HPCG) has been introduced to test these systems under more realistic conditions, better reflecting the behavior of real scientific applications. When running under full workload, these HPC systems show a power absorption on the order of tens of MW (megawatts).

[4] https://top500.org/

Large cosmological simulations require hundreds of thousands core/node hours, forcing HPC facilities to operate at full capacity for extended periods. Sustaining such workloads for such amount of time leads to unsustainable energy costs, especially in an age of environmental awareness (Taffoni et al. 2024; Shukla et al. 2025). It is easy to understand that in the next future the number of HPC facilities will increase and their power demands will rise accordingly, potentially becoming unsustainable. For this reason in the last decades "green awareness" has become increasingly important, and nowadays pure performance is not enough to establish the efficiency of a scientific code. In particular, hardware and software development must advance at the same pace, and co-design is required to guarantee that scientific codes runs efficiently on modern HPC facilities (Goz et al. 2019; Taffoni et al. 2020a; Goz et al. 2020; Lacopo et al. 2025b).

The technical part of this thesis focuses on optimizing the fast cosmological simulation code `PINOCCHIO`. By relying on perturbative dynamics rather than computing the full gravitational interaction between particles, `PINOCCHIO` offers orders-of-magnitude faster performance than traditional N-body simulations while retaining sufficient physical fidelity to model halo formation and merger histories. Thanks to its hybrid MPI+OpenMP CPU parallelization, it plays a central role in producing mock catalogues for surveys such as *Euclid* (Mellier et al. 2024; Monaco et al. 2025), where thousands of realizations of the Universe are required for covariance estimation, pipeline validation, and systematic-error assessment (Fumagalli et al. 2021, 2024; Salvalaggio et al. 2024; Fumagalli et al. 2025b). In this context, improving both the computational performance and the energy efficiency of the code is of great importance.

To meet these demands, part of the `PINOCCHIO` workflow has been ported to modern GPU-accelerated architectures. In Lepinzan et al. (2025), an embarrassingly parallel module of the code, responsible for the computation of collapse times for each particle involved in the simulation (Section 2), was offloaded to GPUs using OpenMP target directives (Martineau et al. 2016), thereby exploiting GPU parallelism while maintaining portability across NVIDIA and AMD platforms. The porting strategy and performance results are discussed in Section 3.

Building on this effort, a companion work Lacopo et al. (2025.) evaluates the energy impact of GPU offloading by comparing CPU and GPU power consumption across different hardware architectures, using the Power Measurement Toolkit (PMT, Corda et al. 2022) to monitor power usage on heterogeneous nodes. The measurement methodology and main results are presented in Section 3.

Together, these two works enable a quantitative assessment of both the speed-up and the energy efficiency of the optimized `PINOCCHIO` implementation, providing insight into its performance on current pre-exascale systems and guiding future optimizations for

exascale-era simulations.

1.4 Cosmic voids in the large-scale structure

Cosmic voids, vast underdense regions in the large-scale structure of the Universe, occupy the majority of its volume. They are promising cosmological probes that offer complementary and improved constraints on cosmological models (Pisani et al. 2019; Davies et al. 2021; Moresco et al. 2022), and their capacity to inform cosmological constraints will continue to grow in this era of large-scale surveys (Hamaus et al. 2022; Contarini et al. 2023; Radinovic et al. 2023). Their underdense nature not only makes them less affected by non-linear gravitational dynamics (Hamaus et al. 2014; Schuster et al. 2023), providing a unique window into the growth of structures and the properties of the Universe's expansion (Hamaus et al. 2016; Schuster et al. 2025), but also provides unique sensitivity to the underlying cosmological model.

From a theoretical perspective, the evolution of cosmic voids can be effectively described using the excursion-set formalism, which provides an analytical framework for modeling their formation and statistical properties. Unlike overdense structures, voids expand quasi-linearly, making them particularly well-suited for this approach. Their evolution follows two key processes: the void-in-void mechanism, where smaller voids merge into larger ones, and the void-in-cloud process, where voids embedded in initially overdense regions are compressed and eventually erased (Sheth & van de Weygaert 2004; Jennings et al. 2013; Schuster et al. 2025).

In addition to providing insights into structure formation, voids serve as powerful probes for testing fundamental physics. Their shapes evolve under the combined influence of tidal distortions and gravitational dilution, making them powerful probes for constraining the dark energy equation of state (Lee & Park 2009; Bos et al. 2012). Furthermore, the size abundance of voids also depends on the dark energy equation of state: as matter flows out of voids, they expand while surrounding structures collapse, making their size and evolution inherently sensitive to the effects of cosmic acceleration (Pisani et al. 2015).

Voids offer an ideal setting for testing modified gravity theories (Li 2011; Davies et al. 2019). In these models, deviations from general relativity are suppressed by screening mechanisms which are generally present in overdense environments, but these same screening mechanisms becomes ineffective in underdense enviroments. This makes voids particularly effective in detecting deviations from General Relativity (GR, Perico et al. 2019). Additionally, void statistics can help disentangle the degeneracy between modified gravity and the neutrino mass sum, probing their combined effects on cosmic structure

formation (Contarini et al. 2021).

Using precise and computationally efficient tools is crucial for studying the large-scale structure of the universe. Fast semi-analytic approaches, such as PINOCCHIO leverage LPT to rapidly generate halo distributions without the computational overhead of full N-body simulations. Previous studies have demonstrated the accuracy of PINOCCHIO in reproducing halo statistics, such as the halo mass function (HMF), halo bias, and 2-point statistics (see, for instance, Monaco et al. 2013; Munari et al. 2017; Fumagalli et al. 2025a; Monaco et al. 2025, and Section 2.5 for an overview). However, its effectiveness in modeling other components of the cosmic web remains an open question.

This motivates the second part of this thesis, where the suitability of PINOCCHIO for tracing cosmic voids is investigated in detail as part of the work presented in Lepinzan et al. (2025). The measurement methodology and main results are presented in Section 6.

Chapter 2

PINOCCHIO

This Chapter introduces the theoretical and algorithmic foundations of the PINOCCHIO code. We begin with an overview of the code, followed by the theoretical framework on which it is built. We first describe Lagrangian Perturbation Theory (LPT) and its role in modelling the growth of cosmic structures, and then introduce the ellipsoidal collapse framework that underpins the prediction of halo formation in PINOCCHIO. A detailed presentation of the code pipeline follows, outlining its main computational steps and algorithmic components. Finally, we review the key cosmological applications of PINOCCHIO and its validation against full N-body simulations.

2.1 Overview of the PINOCCHIO code

PINOCCHIO (PINpointing Orbit Crossing-Collapsed HIerarchical Objects; Monaco et al. 2002a, 2013; Munari et al. 2017; Monaco et al. 2025) is a LPT code designed to rapidly simulate the formation and evolution of dark matter halos, starting from an initial density contrast field δ. The code bridges two key paradigms in structure formation theory: the Extended Press-Schechter (EPS) approach (Press & Schechter 1974; Bond et al. 1991), and its generalization to a more realistic ellipsoidal collapse model (Monaco 1997). These models describe the hierarchical buildup of dark matter halos as overdensities of the primordial density field evolving and collapsing under gravity.

By leveraging this approximate yet accurate formulation, together with a highly parallelized structure for most of its components, PINOCCHIO provides a computationally efficient alternative to full N-body simulations. It achieves good accuracy in reproducing standard halo statistics (e.g. HMF, halo bias, and 2-point statistics), while maintaining a computational cost more than three orders of magnitude lower than N-body calculations (Munari et al. 2017). This balance of accuracy and efficiency makes PINOCCHIO well

suited for applications that require large numbers of realizations.

The code begins by generating a linear density contrast field as a realization of a Gaussian random field, which is then smoothed over a hierarchy of decreasing radii R. The collapse time for each grid point is computed using ellipsoidal evolution (Monaco 1997), approximated through third-order LPT (3LPT). Since LPT accurately describes the dynamics up to orbit crossing events, it provides a reliable estimate for the moment when an ellipsoid collapses along its shortest axis (Monaco 1995). At this stage, the particle (or equivalently the grid point), is considered to have entered a multi-stream regime, becoming part of a dark matter halo or the filamentary network connecting halos (Shandarin & Zeldovich 1989).

Collapsed particles are grouped into halos and filaments using an algorithm named FRAGMENTATION (Monaco et al. 2013), which mimics the hierarchical processes of accretion and merging to construct halo merger trees. The accretion of particles onto a halo, as well as the merging of two halos, is determined by comparing their predicted Eulerian positions, computed via LPT displacements, and applying a distance criterion: accretion or merging occurs when their separation d falls below a threshold $d_{\rm thr}$, which depends on the Lagrangian radius of the larger object. LPT displacements are applied in two key steps: first, during halo construction, and second, when determining their final Eulerian positions for the output catalog [1].

2.2 Lagrangian perturbation theory

The theoretical foundation of PINOCCHIO lies in Lagrangian Perturbation Theory, a perturbative framework for describing the evolution of a self-gravitating fluid in an expanding universe. Unlike the Eulerian formulation, where the observer is fixed in the comoving space and follows the evolution of the density and velocity fields at given positions, the Lagrangian approach tracks individual mass elements as they move with the flow of matter (Shandarin & Zeldovich 1989; Buchert 1992; Buchert & Ehlers 1993).

2.2.1 Eulerian versus Lagrangian descriptions

In the Eulerian formulation of cosmological perturbation theory, the basic variables are the density contrast field $\delta(\mathbf{x}, t)$ and the peculiar velocity field $\mathbf{v}(\mathbf{x}, t)$, both defined at fixed spatial comoving coordinates. Their evolution is governed by the continuity, Euler, and

[1] The displacement of a halo is computed as the average displacement of all particles that belong to it.

Poisson equations in comoving coordinates:

$$\frac{\partial \delta}{\partial t} + \frac{1}{a}\nabla \cdot [(1+\delta)\,\mathbf{v}] = 0, \tag{2.1}$$

$$\frac{\partial \mathbf{v}}{\partial t} + \frac{\dot{a}}{a}\,\mathbf{v} + \frac{1}{a}(\mathbf{v}\cdot\nabla)\mathbf{v} = -\frac{1}{a}\nabla\phi, \tag{2.2}$$

$$\nabla^2\phi = 4\pi G a^2 \bar{\rho}\,\delta, \tag{2.3}$$

where ϕ is the gravitational potential and $\bar{\rho}$ is the mean background density. While these equations fully describe the growth of density perturbations, they become nonlinear and difficult to solve once fluctuations grow large.

The Lagrangian approach reformulates the same dynamics by following the trajectories of individual fluid elements rather than the evolution of fields at fixed comoving positions. In this formalism, each infinitesimal mass element is identified by its initial Lagrangian coordinate $\mathbf{q}$, which serves as a unique label for that element. The position of the same element at cosmic time t (or equivalently, at scale factor a) is given by its Eulerian coordinate $\mathbf{x}(\mathbf{q}, t)$, related to the initial coordinate by a displacement field:

$$\mathbf{x}(\mathbf{q}, t) = \mathbf{q} + \boldsymbol{\Psi}(\mathbf{q}, t), \tag{2.4}$$

where $\boldsymbol{\Psi}(\mathbf{q}, t)$ is the displacement field. Mass conservation requires that:

$$(1+\delta)\,d^3x = d^3q, \tag{2.5}$$

which implies that:

$$1 + \delta(\mathbf{x}, t) = \frac{1}{J(\mathbf{q}, t)}, \qquad J(\mathbf{q}, t) = \det\left(\frac{\partial x_i}{\partial q_j}\right). \tag{2.6}$$

Equation (2.6) shows that the Lagrangian and Eulerian pictures are equivalent as long as the mapping between $\mathbf{q}$ and $\mathbf{x}$ remains one-to-one; when this condition is violated, orbit crossing occurs, marking the transition from a single-stream to a multi-stream regime and causing the two descriptions to cease being equivalent.

2.2.2 Linear growing mode and perturbative expansion

At the linear level, the density contrast δ evolves according to the following second-order differential equation:

$$\frac{\partial^2\delta}{\partial t^2} + 2\frac{\dot{a}}{a}\frac{\partial\delta}{\partial t} = 4\pi G\bar{\rho}\,\delta. \tag{2.7}$$

This equation admits two independent solutions: a decaying mode and a growing mode. The growing mode, denoted by $D(t)$ (or equivalently as a function of the scale factor a), represents the linear growth factor that governs the amplification of density perturbations due to gravitational instability. It therefore provides a natural time-dependent variable for describing structure formation in the linear regime.

In the Lagrangian formalism, the problem of the evolution of a self-gravitating, pressureless fluid is reformulated in terms of the dynamics of the displacement field $\mathbf{\Psi}(\mathbf{q}, t)$. The goal of LPT is to determine $\mathbf{\Psi}$ from the equations of motion in an expanding background. A complete derivation of these equations can be found in Catelan (1995) or Buchert (1995). The main idea is that the Euler-Poisson system of equations can be recast into an equivalent set of equations for $\mathbf{\Psi}(\mathbf{q}, t)$. For the purposes of the following discussion, it is sufficient to recall that for small displacements, the system can be solved perturbatively by expanding the displacement field in powers of the linear growth factor $D(t)$. Owing to the separability of the solutions in time and space, as shown in Ehlers & Buchert (1997), the displacement field can be written as:

$$\mathbf{\Psi} = D(t)_1\,\mathbf{\Psi}^{(1)} + D_2(t)\,\mathbf{\Psi}^{(2)} + D_{31}(t)\,\mathbf{\Psi}^{(31)} + D_{32}(t)\,\mathbf{\Psi}^{(32)} + \cdots, \tag{2.8}$$

where each term represents a contribution of increasing perturbative order, characterized by a time-dependent growth factor $D(t)_i$, which encodes the dependence on the background cosmology, and a purely spatial displacement field Ψ^n, obtained from Poisson-like equations. At third order, only the two independent scalar contributions (Ψ^{31} and Ψ^{32}) are retained in the PINOCCHIO implementation.

The first-order term of Eq. (2.8) corresponds to the Zel'dovich approximation (ZA) (Zel'dovich 1970), which provides an intuitive and accurate description of structure formation up to the rising of nonlinearity. In this approximation, particles move along straight trajectories determined by the gradient of the initial gravitational potential, while their velocity remains constant when expressed in terms of the growth factor $D(t)$:

$$\mathbf{x}(\mathbf{q}, t) = \mathbf{q} - D(t)\,\nabla_q\phi(\mathbf{q}), \tag{2.9}$$

where $\phi(\mathbf{q})$ is the rescaled peculiar potential. The corresponding velocity is:

$$\mathbf{u} = \frac{d\mathbf{x}}{dD} = -\nabla_q \phi(\mathbf{q}), \tag{2.10}$$

which remains constant with respect to $D(t)$.

The density field follows from mass conservation, as in Eq. (2.5) and Eq. (2.6), and can be written as:

$$1 + \delta(\mathbf{q}, t) = \frac{1}{\det\left(\frac{\partial x_i}{\partial q_j}\right)} = \frac{1}{|[1 - D(t)\lambda_1][1 - D(t)\lambda_2][1 - D(t)\lambda_3]|}, \tag{2.11}$$

where λ_i are the eigenvalues of the deformation tensor $\partial_i \partial_j \phi(\mathbf{q})$. This expression shows that the density formally diverges when $D(t) = 1/\lambda_i$, corresponding to the intersection of particle trajectories and the formation of caustics. The first collapse occurs along the largest eigenvalue λ_1 direction.

Despite its simplicity, the ZA reproduces the correct large-scale behavior of structure formation and provides the foundation for higher-order LPT, as well as for the ellipsoidal collapse model implemented in PINOCCHIO, where 2LPT and 3LPT terms are used to include tidal effects, improving accuracy in the mildly nonlinear regime.

2.2.3 Orbit crossing and the limits of LPT

As noticed in the ZA, a natural limitation of the Lagrangian formulation arises when different mass elements reach the same Eulerian position during gravitational collapse. This event is known as orbit crossing (OC) or shell crossing. Mathematically, OC occurs when the Jacobian determinant of the mapping in Eq. (2.4) vanishes:

$$J(\mathbf{q}, t) = \det\left(\frac{\partial x_i}{\partial q_j}\right) = 0. \tag{2.12}$$

At this point, the mapping $\mathbf{q} \to \mathbf{x}$ is no longer one-to-one, as multiple Lagrangian points collapse to the same Eulerian location. The fluid element density:

$$1 + \delta(\mathbf{x}, t) = \frac{1}{J(\mathbf{q}, t)}, \qquad J(\mathbf{q}, t) = \det\left(\frac{\partial x_i}{\partial q_j}\right), \tag{2.13}$$

formally diverges at $J = 0$, producing singularities known as caustics. The regions where OC has occurred are referred to as multi-stream regions, characterized by overlapping particle trajectories and a breakdown of the single-stream, collisionless-fluid approximation.

Physically, the occurrence of OC marks the onset of strong nonlinearity in the DM field.

While the Lagrangian formulation provides an accurate analytical description up to the first OC event, it cannot by itself describe the fully nonlinear evolution beyond this point.

Despite this limitation, LPT remains a powerful approximation when applied to smoothed versions of the initial density field. By filtering out small-scale fluctuations that would otherwise lead to early OC, it is plausible to assume that large-scale modes evolve quasi-linearly and are only weakly affected by small-scale multi-streaming. This scale-separation hypothesis underlies most dynamical approximations based on the Lagrangian formulation, including PINOCCHIO, which exploits LPT predictions up to the point of collapse.

2.3 Ellipsoidal collapse

The ZA describes the evolution of a fluid element as an initially homogeneous region that deforms and collapses independently along the three axes determined by the eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$ of the deformation tensor. Collapse along each axis occurs when $D(t) = 1/\lambda_i$, and the first axis to collapse is that associated with the largest eigenvalue λ_1. However, this treatment assumes that the three directions evolve independently and neglects both the nonlinear coupling between them and the influence of the surrounding tidal field. In other words, the ZA provides an exact solution up to OC only in the case of planar symmetry.

While the ZA provides a simple and elegant description of anisotropic collapse, it ultimately fails to reproduce the correct abundance of collapsed structures when used to predict the halo mass function (HMF) as shown in Monaco (1995). The HMF is a key statistical measure in cosmology that describes the number density of DM halos as a function of their mass and redshift (Press & Schechter 1974; Sheth & Tormen 1999; Castro et al. 2023), and is determined by the fraction of mass elements that have collapsed by a given time.

2.3.1 Lagrangian deformation and variable change

As already pointed out for the ZA in Section 2.2, in the Lagrangian framework, the collapse condition depends on the local deformation of the gravitational potential, described by the three eigenvalues of its Hessian. These eigenvalues, following Monaco (1995), can be reparameterized in terms of more convenient variables that separate the effects of density and shear:

$$\delta = -\lambda_1 - \lambda_2 - \lambda_3, \qquad x = \lambda_1 - \lambda_2, \qquad y = \lambda_2 - \lambda_3, \tag{2.14}$$

where δ represents the local overdensity, and (x, y) quantify the anisotropy of the perturbation: when $x = 0$ and $y = 0$, the shear vanishes and the perturbation is perfectly spherical.

For a Gaussian random field, the joint probability distribution of the eigenvalues of the deformation tensor is given by the Doroshkevich formula (Doroshkevich 1970). In the variables (δ, x, y), this defines a joint probability density $P(\delta, x, y)$ that describes the likelihood of finding a region with a given overdensity and tidal configuration. This distribution provides the statistical foundation for connecting local collapse dynamics to the HMF: once a collapse threshold $\delta_c(x, y)$ is defined, the mass fraction in halos follows from integrating $P(\delta, x, y)$ over all configurations that satisfy the collapse condition.

2.3.2 Collapse threshold in the ZA and link to HMF

Within the parametrization defined in Eq. (2.14), the ZA provides a simple analytical expression for the collapse time. For a perturbation characterized by (δ, x, y), the collapse condition in ZA corresponds to the vanishing of one of the principal axes, leading to a collapse factor proportional to $3/(\delta + x + 2y)$. If $\delta_c(x, y)$ is defined as the linear overdensity required for a perturbation to collapse at $a_c = 1$, this condition can be written as:

$$\delta_c(x, y) = 3 - (x + 2y). \tag{2.15}$$

This relation shows that the collapse threshold depends explicitly on the local tidal shear through (x, y): for $x = y = 0$, corresponding to a spherical configuration with vanishing shear, the ZA prediction for $\delta_c = 3$. This value is significantly higher than the linearly extrapolated spherical-collapse threshold, $\delta_c \simeq 1.686$, associated with perturbations whose nonlinear evolution formally diverges at collapse, indicating that the ZA underestimates the actual amount of collapsed matter, especially for nearly spherical perturbations.

The cumulative fraction of matter contained in collapsed structures with mass greater than M can be obtained by integrating the joint probability distribution $P(\delta, x, y)$ over all initial conditions that have collapsed by the present time, i.e. those for which $\delta > \delta_c(x, y)$:

$$F(> M) = \int_0^\infty dx \int_0^\infty dy \int_{\delta_c(x,y)}^\infty P(\delta, x, y)\, d\delta. \tag{2.16}$$

In the case in which the probability distribution $P(\delta, x, y)$ is assumed to be Gaussian in the linearly extrapolated density field δ, with variance $\sigma^2(M)$, on mass scale M, Eq. (2.16) reduces to the well-known Press–Schechter (PS) formulation (Press & Schechter 1974). In

this limit, the fraction of mass collapsed into objects with mass larger than M is given by:

$$F_{\rm PS}(>M) = 2\int_{\delta_c}^{\infty} P(\delta; M)\, d\delta = \mathrm{erfc}\left(\frac{\delta_c}{\sqrt{2}\,\sigma(M)}\right), \tag{2.17}$$

where δ_c is the linearly extrapolated collapse threshold, $\sigma(M)$ denotes the standard deviation of the smoothed density field, and the factor 2 accounts for the cloud-in-cloud problem inherent to the PS formalism.

The HMF then follows from differentiating the collapsed fraction with respect to mass:

$$N(M)\, dM = \bar{\rho}\,\frac{\partial F(>M)}{\partial M}\,\frac{dM}{M} = \bar{\rho}\,\frac{\partial F}{\partial \sigma}\,\frac{d\sigma}{dM}\,\frac{dM}{M}, \tag{2.18}$$

where $\bar{\rho}$ is the mean matter density and $\sigma(M)$ indicates the standard deviation of the linear density field smoothed on scale M. Because the ZA overestimates the collapse threshold δ_c, it predicts a too small collapsed fraction $F(>M)$ and thus an underabundance of halos, particularly in the high-mass tail of the HMF.

2.3.3 Ellipsoidal collapse model

This shortcoming motivates the adoption of a more accurate dynamical model, namely, the ellipsoidal collapse model,in which the three main axes of a homogeneous ellipsoid evolve self-consistently under both their mutual gravitational coupling and the influence of the external tidal field.

The Lagrangian perturbative expansion, presented in Eq. (2.8), can be truncated in such a way that the dynamics of a mass element formally resemble those of a homogeneous ellipsoid evolving under the influence of an external tidal shear field (Monaco 1995; Bond & Myers 1996; Monaco 1997).

This approximation naturally emerges when the peculiar gravitational potential is expanded as a Taylor series around the lagrangian position $\mathbf{q}$, retaining terms up to second order:

$$\phi(\mathbf{q}) \simeq \frac{1}{2}\,\phi_{,ab}\, q_a q_b, \tag{2.19}$$

where $\phi_{,ab}$ is the deformation tensor and repeated indices are implicitly summed over. In its principal-axis frame, the potential can be written as:

$$\phi(\mathbf{q}) = \frac{1}{2}\left(\lambda_1 q_1^2 + \lambda_2 q_2^2 + \lambda_3 q_3^2\right), \tag{2.20}$$

with eigenvalues λ_i ordered as $\lambda_1 \geq \lambda_2 \geq \lambda_3$. The trace of this tensor defines the linear

density contrast:

$$\delta_l = \lambda_1 + \lambda_2 + \lambda_3, \tag{2.21}$$

which remains constant in time at linear order.

At an initial epoch t_i, the semi-axes a_i of the ellipsoid can be written as:

$$a_i = a(t_i)\,[1 - D(t_i)\,\lambda_i], \tag{2.22}$$

where $a(t)$ is the cosmological scale factor and $D(t)$ is the linear growth factor. This corresponds to an initially spherical region slightly perturbed by the local tidal field. The subsequent evolution of the three axes can be obtained either by numerically integrating the full ellipsoidal-collapse equations (Monaco 1995; Bond & Myers 1996), or by solving the third-order Lagrangian equations specialized for the ellipsoidal case, in which only first and second derivatives of the potential are retained. By doing this, the collapse factor D_c, defined as the value of the linear growth factor $D(t)$ at which one of the ellipsoidal axes collapses, can be computed perturbatively at different LPT orders (Monaco 1997).

At first order (ZA), the collapse of a general ellipsoid occurs when:

$$D_c^{(1)} = \frac{1}{\lambda_1}, \tag{2.23}$$

where λ_1 is the largest eigenvalue of the deformation tensor, corresponding to the axis that collapses first.

At second order, the solution for initially overdense ellipsoids becomes:

$$D_c^{(2)} = \frac{7\lambda_1 - \sqrt{7\lambda_1(\lambda_1 + 6\delta_l)}}{3\lambda_1(\lambda_1 - \delta_l)}, \tag{2.24}$$

where $\delta_l = \lambda_1 + \lambda_2 + \lambda_3$ is the linear overdensity. This expression, however, gives unphysical results for large underdensities ($\delta_l < 0$), as even voids are spuriously predicted to collapse, illustrating the breakdown of 2LPT in such regimes.

At third order, the collapse factor $D_c^{(3)}$ is obtained as the smallest non-negative root of the cubic equation:

$$1 - \lambda_1 D_c^{(3)} - \frac{3}{14}\lambda_1(\delta_l - \lambda_1)(D_c^{(3)})^2 - \left[\frac{\mu_3}{126} + \frac{5}{84}\lambda_1\delta_l(\delta_l - \lambda_1)\right](D_c^{(3)})^3 = 0, \tag{2.25}$$

where $\mu_3 = \lambda_1\lambda_2\lambda_3$. The smallest positive root of this equation represents the physical collapse time.

The third-order LPT solution provides an excellent approximation to the exact ellip-

soidal dynamics across a wide range of configurations, differing appreciably only in the perfectly spherical limit. A small empirical correction to the collapse time, introduced in Monaco (1997), restores agreement with the exact spherical-collapse threshold. Beyond its accuracy, this formalism offers a general analytic approximation for the evolution of mass elements under gravity, forming the dynamical basis for semi-analytic models such as PINOCCHIO.

The same physical considerations motivating the ellipsoidal collapse model also form the basis of the Sheth–Tormen (ST) model for the HMF (Sheth et al. 2001; Sheth & Tormen 2002). This model provides a statistical implementation of the ellipsoidal collapse picture within the excursion-set formulation of the EPS theory. In this framework, the dependence of the collapse threshold on tidal shear and triaxiality is incorporated phenomenologically by replacing the constant spherical-collapse barrier with an effective barrier that depends on the variance of the smoothed density field. This moving barrier encodes the delayed collapse of low-mass perturbations and the enhanced collapse efficiency of rare, high-mass peaks, thereby modifying the first-crossing statistics of density trajectories and yielding halo abundances in significantly improved agreement with N-body simulations.

A closely related prescription is implemented explicitly in semi-analytic algorithms such as PINOCCHIO, where the collapse time of each mass element is computed as a function of the smoothing scale (Section 2.4), and hence of the variance, using ellipsoidal collapse dynamics. In this way, PINOCCHIO provides a dynamical realization of the moving-barrier concept underlying the ST approach.

2.4 Legacy implementation of PINOCCHIO

The theoretical formalism described in the previous Sections is implemented in PINOCCHIO through a fully Lagrangian approach: each grid element evolves as a homogeneous ellipsoid whose collapse time is estimated from the local tidal deformation field. From an algorithmic point of view, the code operates in three main stages: `Initialization`, `Fmax` and `Fragmentation`.

2.4.1 Initialization

The first stage of PINOCCHIO generates a Gaussian random realization of the linear density contrast field $\delta(\mathbf{x})$, consistent with the input power spectrum $P(k)$. This is performed directly in Fourier space, where to each mode $\tilde{\delta}(\mathbf{k})$ is assigned a random phase and an amplitude drawn from a Gaussian distribution with variance proportional to $P(k)$. We refer

to this stage as the Initialization module.

The procedure closely follows the implementation adopted in initial-condition (ICs) generators such as N-GenIC (Springel 2005) and 2LPTic[2] (Crocce et al. 2006), ensuring that simulations initialized with the same random seed reproduce identical large-scale modes. To guarantee full reproducibility across different domain decompositions, PINOCCHIO adopts a deterministic loop over Fourier modes: each process initializes a 2D array of random seeds, one for each "pencil" of cells along the simulation box, and then populates k-space sequentially by drawing random phases and amplitudes from these predefined sequences. This approach ensures that the resulting field is independent of the number of MPI tasks or their spatial domain distribution.

Fourier modes are filled starting from the fundamental mode and proceeding outward in concentric shells in k-space. As a result, increasing the grid resolution preserves all large-scale fluctuations while simply adding smaller-scale structure.

2.4.2 Fmax

The second stage of the code computes the collapse time for each grid element. This quantity determines, as already explained in Sections 2.2 and 2.3, the epoch at which a given Lagrangian region undergoes gravitational collapse and is one of the core predictive quantity of PINOCCHIO.

The calculation relies on the local tidal field, described by the eigenvalues $(\lambda_1, \lambda_3, \lambda_3)$ of the Hessian of the gravitational potential $\partial_i \partial_j \phi$. These are obtained efficiently in Fourier space using Fast Fourier Transforms (FFTs), since spatial derivatives become simple multiplications by the wavenumber components. In Fourier space, the Poisson equation reads:

$$\tilde{\phi}(\mathbf{k}) = -\frac{\tilde{\delta}(\mathbf{k})}{k^2}, \tag{2.26}$$

so that the second derivatives are computed as:

$$\frac{\partial^2 \phi}{\partial x_i \partial x_j} \longrightarrow -k_i k_j \tilde{\phi}(\mathbf{k}). \tag{2.27}$$

This approach allows all derivatives to be calculated exactly on the grid and makes the operation naturally parallelizable. The calculation is repeated over a hierarchy of 10–20 smoothing scales, enabling the identification of collapsed structures over a wide mass range.

Once the deformation tensor is computed, it is diagonalized to obtain its three eigen-

[2] https://github.com/manodeep/2LPTic

values which characterize the local tidal field and determine the collapse dynamics of the corresponding Lagrangian region. Each mass element (grid cell) is then modeled as a homogeneous ellipsoid evolving under its own gravity, with initial conditions given by the Hessian eigenvalues. The collapse time of the ellipsoid is then computed with two alternative approaches:

- *Classic* (analytic) method: the evolution of the ellipsoid is approximated using 3LPT, leading to a non-linear algebraic formulation of the collapse time in terms of the Hessian eigenvalues as described in Section 2.3.

- *Tabulated* (numerical) method: the evolution of the ellipsoid is computed without approximations by solving a system of nine Ordinary Differential Equations (ODEs), following the formulation of Nadkarni-Ghosh & Singhal (2016). The system is expressed in terms of dimensionless variables associated with the three principal axes of the ellipsoid and defined as follows:

$$\lambda_{a,i} = 1 - \frac{a_i}{a}, \tag{2.28}$$

$$\lambda_{v,i} = \frac{1}{H}\frac{\dot{a}_i}{a_i} - 1, \tag{2.29}$$

$$\lambda_{d,i} = \frac{\delta\,\alpha_i}{2} + \lambda_{\mathrm{ext},i}, \tag{2.30}$$

  where $i = 1, 2, 3$, $a_i(t)$ denote the scale factors of the three principal axes of the ellipsoid, while $a(t)$ is the background cosmological scale factor. The variables $\lambda_{a,i}$ quantify the deformation of the ellipsoid along its principal axes and thus characterize its evolving shape. The variables $\lambda_{v,i}$ measure the deviation of the expansion or contraction rate of each axis from the background Hubble flow, while $\lambda_{d,i}$ correspond to the eigenvalues of the tidal tensor and describe the local gravitational field driving the collapse. The terms $\lambda_{\mathrm{ext},i}(t)$ model the contribution of the external tidal field acting on the ellipsoid, while the coefficients α_i encode the dependence of the internal gravitational potential on the ellipsoidal geometry. The collapse time is defined as the epoch at which the first axis collapses, i.e. when $\lambda_{a,i} \to 1$.

  Within PINOCCHIO framework, this numerical approach is mainly used in the case of modified gravity (see, for example Song et al. 2022), where the analytic solution based on LPT is not applicable. Since this approach is significantly slower, collapse times are precomputed on a grid of eigenvalues and subsequently interpolated. For computational convenience eigenvalues are sampled in the space defined by the variables $\delta = \Sigma_i \lambda_i$, $x = \lambda_1 - \lambda_2$ and $y = \lambda_2 - \lambda_3$, with eigenvalues sorted in descending order. Since the collapse time function exhibits non-trivial dependence

along the δ direction, requiring non-uniform sampling, the interpolation on this grid is performed using a cubic spline, whereas a simpler (and faster) bi-linear interpolation is performed in the x and y variables.

In both approaches, the code returns the value of the linear growth rate $D(a)$, with a being the scale factor, at the time of collapse $t_{\rm coll}$. To obtain the corresponding scale factor $a_{\rm coll}$, the function $D(a)$ must be inverted. To this end, the $D(a)$ function is first computed by solving the ODE governing its evolution, then interpolated with a cubic spline to obtain its inverse $a(D)$. The collapse redshift is finally obtained from the relation $z_{\rm coll} = 1/a_{\rm coll} - 1$. Consequently, spline interpolation is required in both cases.

The collapse-time calculation is implemented within the broader `Fmax` module, which also includes the FFT operations mentioned above that relies on the `pFFT` library[3].

2.4.3 Fragmentation

The third and final stage assembles the collapsed particles into gravitationally bound structures, such as halos and filaments, and reconstructs the halos hierarchical evolution. This process is handled by the `Fragmentation` module. Using the collapse times computed in the previous stage, the code builds merger trees and traces the growth of dark matter halos through accretion and merging, while also identifying the filamentary network connecting them.

Each collapsed mass element can either seed the formation of a new halo, accrete onto an existing halo, or become part of a filament. The algorithm proceeds as follows:

1. For each collapsing particle, the code inspects its six nearest neighbors in Lagrangian space.

2. If a particle collapses before any of its neighbors, it becomes a one-particle halo seed.

3. A collapsing particle may accrete onto an existing halo if it is adjacent to it in Lagrangian space. The accretion condition is verified by moving both the particle and the halo to their predicted Eulerian positions using the LPT displacement field and comparing their distance to a threshold value.

4. Particles that collapse but do not meet the accretion condition are temporarily classified as *filaments*. They may later accrete if one of their neighboring particles joins a halo.

[3] https://github.com/mpip/pfft

5. Whenever a collapsing particle simultaneously "touches" two halos in Lagrangian space, a *merger check* is triggered. If, after displacing both halos to their predicted positions, their distance falls below the merger threshold, the halos are merged into a single structure. The final halo mass and position are computed as the averages over all the particles composing it.

6. If the particle has not been accreted previously, it is re-evaluated. If none of its neighboring particles belong to a halo, the particle is tagged as a filament.

The particle and halo displacements used in these steps are computed in Fourier space using Fast FFTs, in the same way as in the `Fmax` module. In Fourier space, the displacement field is obtained from the density field as:

$$\mathbf{\Psi}(\mathbf{k}) = i\,\frac{\mathbf{k}}{k^2}\,\tilde{\delta}(\mathbf{k}), \tag{2.31}$$

which allows accurate and efficient reconstruction of particle trajectories before collapse.

The accretion and merger thresholds are defined in terms of characteristic halo radii and a set of calibration parameters tuned against N-body simulations. The threshold distance for a collapsed particle to accrete onto a halo of mass M_h and radius $R = M_h^{1/3}$ is defined as:

$$d^2_{\rm thr} = \begin{cases} (f_a R^e)^2 + (f_{200} R)^2, & D_0\sigma \le \sigma_0, \\ \left[f_a R^e\,(1 + f_{\rm ra}\,(D_0\sigma - \sigma_0))\,\right]^2 + (f_{200} R)^2, & D_0\sigma > \sigma_0, \end{cases} \tag{2.32}$$

Similarly, the threshold distance for merging between two halos is given by:

$$d'^2_{\rm thr} = \begin{cases} (f_m R^e_{\rm lar})^2 + (f_{200} R_{\rm lar})^2, & D_0\sigma \le \sigma_0, \\ \left[f_m R^e_{\rm lar}\,(1 + f_{\rm rm}\,(D_0\sigma - \sigma_0))\,\right]^2 + (f_{200} R_{\rm lar})^2, & D_0\sigma > \sigma_0, \end{cases} \tag{2.33}$$

where, D_0 is the normalized growth factor at final time, σ is the variance of the linear density contrast on the grid and is a function of time, while σ_0 is a free parameter controlling the change of the virial radius. The parameters $\{e, f_a, f_m, f_{\rm ra}, f_{\rm rm}, f_{200}\}$ are additional free parameters calibrated against N-body simulations in ΛCDM. These parameters ensure that `PINOCCHIO` reproduces the halo mass function, merger histories, and clustering properties measured in full cosmological simulations, accurately capturing the hierarchical growth of structure in the cosmic web.

Applications of `PINOCCHIO` to extended cosmological models, including massive neutrino cosmologies (Rizzo et al. 2017; Adamek et al. 2023) and modified gravity scenarios such as *f(R)* (Moretti et al. 2020) and cubic Galileon models (Song et al. 2022), have

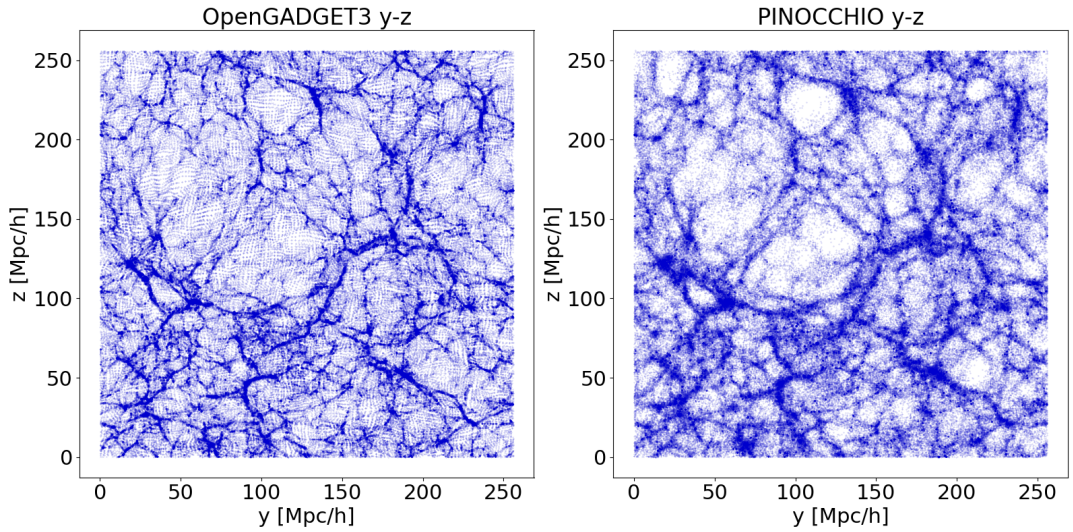


Figure 2.1: Comparison of the displacement fields in a 256^3 particle simulation box of side $256\,h^{-1}$ Mpc. The figure shows a $5\,h^{-1}$ Mpc-thick slice. Left: displacement field from a full N-body simulation obtained with the `OpenGADGET3` code. Right: corresponding 3LPT displacement field computed with `PINOCCHIO`.

shown that retaining the ΛCDM calibration does not lead to significant degradation in the accuracy of halo statistics when compared to the corresponding N-body simulations (see Section 2.5.2 for more details). Physically, the calibrated parameters primarily regulate aspects of halo identification in the nonlinear regime, typically after turn-around and shell-crossing. While extensions beyond ΛCDM can modify the background expansion and the linear growth of perturbations, and in some cases introduce additional scale- or environment-dependent effects, the impact of these modifications on the collapse stage controlled by the calibrated parameters is generally subdominant on the scales relevant for halo formation. For this reason, a recalibration of these parameters are not generally required,

This step represents another core component of the code. Without the `Fragmentation` stage, and by using only the Lagrangian displacements described in Eq. (2.31), the output would simply consist of a displacement field similar to that shown in Figure 2.1. This example shows the displacement field, at $z = 0$, within a thin slice of the simulation box, with a depth of $5\,h^{-1}$ Mpc. The left panel displays the displacement field obtained from a full N-body simulation using the `OpenGADGET3` code (described later in Section 6.4.1), while the right panel shows the corresponding field computed with `PINOCCHIO` using third-order Lagrangian perturbation theory (3LPT). Despite the absence of non-linear mode coupling in the latter, the large-scale flow patterns and coherent structures are clearly reproduced, with multi-streaming regions appearing significantly thickened, demonstrating the accuracy of the 3LPT displacement field in the quasi-linear regime.

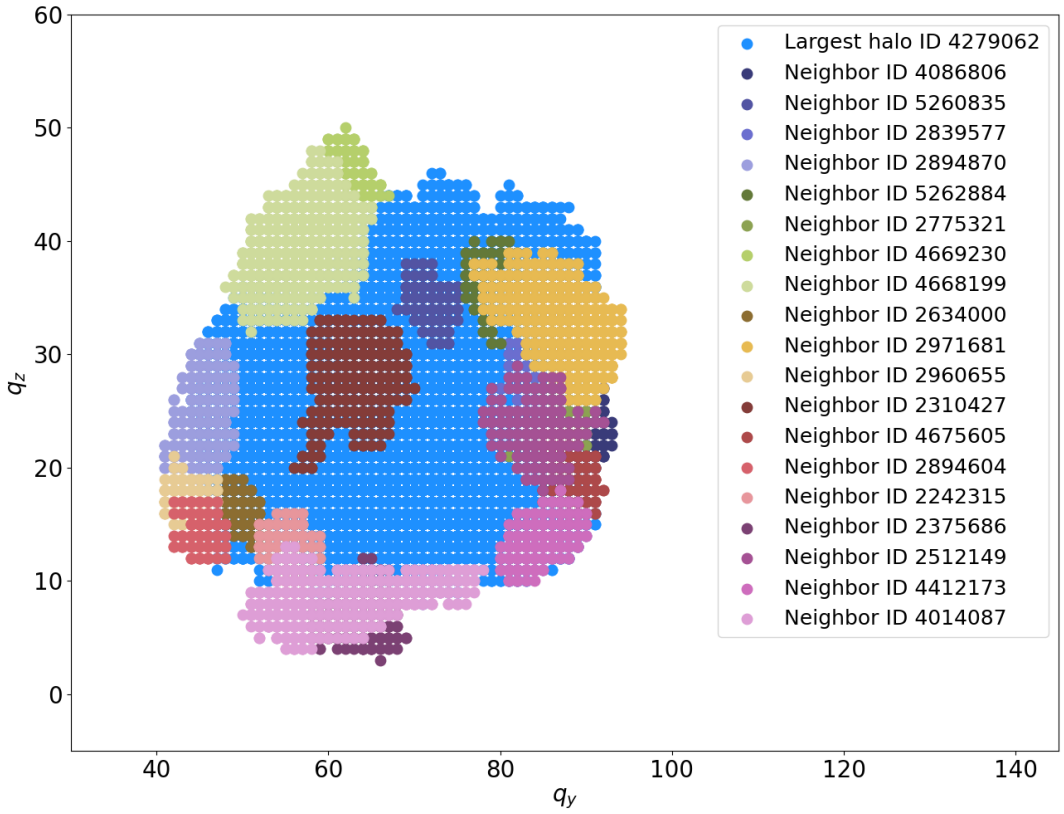


Figure 2.2: Halos identified by the `Fragmentation` procedure directly on the initial Lagrangian grid. The figure shows a fixed-depth slice of the simulation box, highlighting one of the most massive halos (in blue) and its 20 nearest neighboring halos. Each halo corresponds to a connected region of collapsed particles as defined by the local-neighbour accretion and merger criteria described in Eq. (2.32) and (2.33).

By applying the `Fragmentation` procedure directly on the Lagrangian grid, as described above, using collapse times and local-neighbour checks, `PINOCCHIO` converts the diffuse, continuous matter field into a discrete population of halos. The LPT displacement field is used only to predict Eulerian positions for accretion/merger decisions and to place halos in the final catalog. In particular, halo construction relies on 2LPT, whereas halo displacements are computed using 3LPT. As an illustration, Figure 2.2 shows halos identified on the initial grid within a slice of fixed depth, highlighting one of the most massive halos and its 15 nearest neighbours.

Within this framework, once the `Fragmentation` procedure is applied to the entire simulation box, the diffuse displacement field shown in Figure 2.1 collapses into well-defined structures, as illustrated in Figure 2.3. This figure is directly comparable to Figure 2.1: while the previous one shows the continuous displacement field, here we display the resulting collapsed matter distribution after `Fragmentation`. The left panel corresponds to the result obtained with `OpenGADGET3`, whereas the right panel shows the

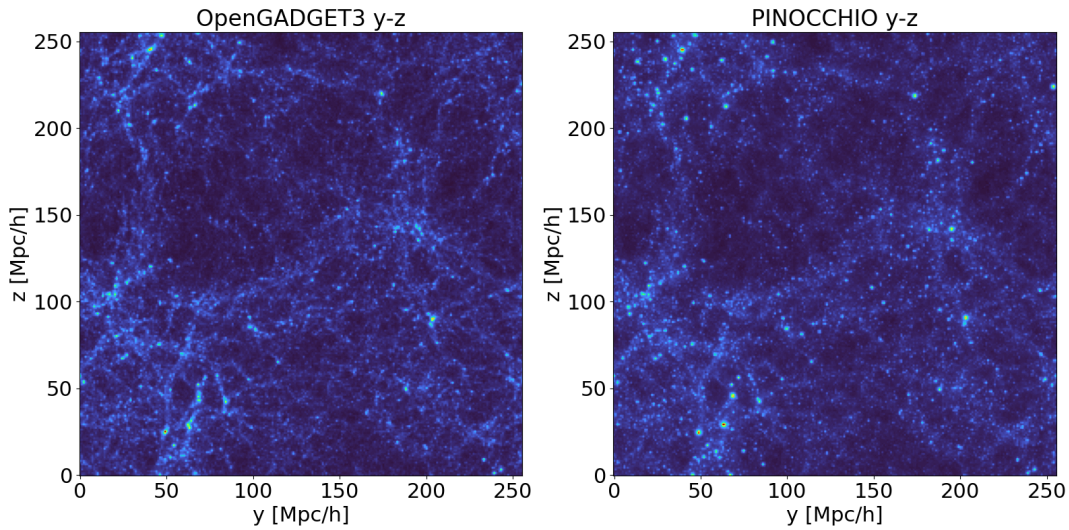


Figure 2.3: Same as Figure 2.1, but showing the collapsed density field after applying the `Fragmentation` algorithm to the full simulation box for `PINOCCHIO`.

equivalent prediction from `PINOCCHIO`, where collapse is modeled analytically through ellipsoidal dynamics and implemented via the Lagrangian `Fragmentation` algorithm.

2.5 Cosmological applications of `PINOCCHIO`

This Section briefly highlights some of the main cosmological applications of `PINOCCHIO`. Beyond its role as a fast and accurate generator of dark matter halo catalogs, the code has been employed to explore a variety of physical scenarios, from standard ΛCDM structure formation to modified gravity and massive neutrino cosmologies, as well as early-Universe supermassive black hole seeding. Its computational efficiency also makes it an ideal tool for producing large ensembles of mock realizations for covariance estimation and validation, such as those required for the *Euclid* mission.

However, while its performance in predicting halo abundances and clustering is well established, its effectiveness in modeling other components of the cosmic web, such as cosmic voids, has remained unexplored. This gap has been recently addressed in Lepinzan et al. (2025), and is presented in detail in Chapter 6.

In addition, because `PINOCCHIO` is fundamentally a Lagrangian method, evaluating its accuracy at the particle level provides a crucial, complementary validation. A dedicated particle-by-particle comparison with a full N-body simulation is therefore performed, offering a direct test of the predicted collapse ordering and Lagrangian dynamics

2.5.1 Cluster cosmology

In previous works, such as Monaco et al. (2013) and Munari et al. (2017), it has been shown that PINOCCHIO accurately reproduces key overdensity summary statistics, such as the HMF, halo bias, and two-point statistics, when compared to full N-body simulations. In particular, Munari et al. (2017) demonstrated that using 3LPT for halo displacements and 2LPT for halo construction yields agreement within 10% in the halo power spectrum $P(k)$ (up to $k_{\rm max} \sim 0.5h\,{\rm Mpc}^{-1}$) and 5–10% in the HMF (up to $\sim 10^{15} M_\odot$).

Building upon these results, Fumagalli et al. (2024) used 1000 PINOCCHIO simulations, designed to mimic the expected *Euclid* cluster sample, to calibrate a model for the clustering covariance of galaxy clusters. Salvalaggio et al. (2024) validated an analytical model for the galaxy bispectrum covariance and the power spectrum-bispectrum cross-covariance in redshift space by comparing it with numerical estimates obtained from 10.000 PINOCCHIO mock halo catalogs. More recently, Fumagalli et al. (2025a) extended this validation by showing that PINOCCHIO can reproduce the two-point correlation function (2PCF) covariance measured in a full N-body code, namely OpenGADGET3 described in Section 6.4.1, at the few-percent level, provided that HMF are carefully rescaled.

Going beyond two-point statistics, PINOCCHIO has also been extensively employed in the analysis of higher-order clustering statistics. In particular, Oddo et al. (2020) used large ensembles of PINOCCHIO mock catalogues to estimate the covariance of the halo bispectrum, while Veropalumbo et al. (2022) validated covariance estimates for the three-point correlation function (3PCF) by comparing numerical results obtained from PINOCCHIO catalogues with those measured in full N-body simulations.

2.5.2 Extension to scale-dependent growth cosmologies: $\nu\Lambda$CDM and modified gravity

The study described in Rizzo et al. (2017), further extends PINOCCHIO to cosmologies with scale-dependent growth, such as the $\nu\Lambda$CDM scenario. Validation against N-body simulations shows that the code reproduces the main halo statistics, including the HMF and halo power spectrum, with a precision of 5–10%, comparable to the original implementation. It also captures key physical effects, such as the Ω_ν–σ_8 degeneracy, confirming the reliability of PINOCCHIO in modeling structure formation in neutrino cosmologies.

In the context of neutrino cosmology PINOCCHIO was also included in the *Euclid* code-comparison study of Adamek et al. (2023) as a fast approximate method for modelling massive-neutrino effects on halo clustering.

The modified gravity scenario explored within the PINOCCHIO code involves extending

standard cosmological models to include modified gravitational theories, such as $f(R)$ (Hu & Sawicki 2007) gravity, cubic Galileon (Nicolis et al. 2009), and the Dvali-Gabadadze-Porrati (nDGP) model (Dvali et al. 2000).

The first study, described in Moretti et al. (2020), introduces a new numerical method based on 2LPT to determine the Lagrangian displacement field in modified gravity. The method is tested using N-body simulations and found to provide accurate results for halo power spectra $P(k)$, with precision within 10% for scales up to $k \approx 0.2$–$0.4\,h\,\mathrm{Mpc}^{-1}$, as well as for halo positions, with an error that is a fraction of the inter-particle distance.

Building on this methodology, Song et al. (2022) extends the 2LPT and ellipsoidal collapse prescriptions to the cubic Galileon model by explicitly incorporating Vainshtein screening. The resulting implementation is shown to reproduce the trends observed in N-body simulations for the halo mass function, halo bias, and matter power spectrum, while retaining a computational cost that is orders of magnitude lower than that of full simulations.

Finally, Song et al. (2024) applies the same Lagrangian-based framework to the nDGP model. The nGDP-PINOCCHIO implementation is tested against N-body simulations, showing that the resulting halo power spectrum $P(k)$ agrees with simulations within 5% for $k < 0.3h\mathrm{Mpc}^{-1}$ at $z = 0$, with improved agreement at higher redshifts. The cumulative HMF is also consistent with simulation results within the statistical scatter.

2.5.3 Galaxy clustering

Monaco et al. (2025), presents two large suites of light-cone simulations and mock galaxy catalogues generated with the latest version of the PINOCCHIO code. These include over 4500 realizations designed to reproduce the *Euclid* spectroscopic sample by populating halos with galaxies through an halo occupation distribution (HOD) model calibrated on the *Euclid* Flagship simulation. The two main sets, *Geppetto* and *EuclidLargeMock*, cover areas of 2763 deg^2 and half the sky, respectively. Validation through number densities, power spectra $P(k)$, and correlation functions shows good agreement with the Flagship catalogue, with minor deviations at $k > 0.2\,h\,\mathrm{Mpc}^{-1}$, as illustrated in Figure 2.4. Owing to the wavelength coverage of the NISP red grism, described in Section 1.2, the Hα emission line can be observed over the redshift range $z \simeq 0.9$–1.8. Galaxies with reliable spectroscopic redshift measurements are therefore expected to have line fluxes above a fiducial threshold of $f_0 = 2 \times 10^{-16}\mathrm{ergs}^{-1}\mathrm{cm}^{-2}$. In Figure 2.4, we show the $P(k)$ multipoles measured for galaxies satisfying this flux selection in four redshift bins within the aforementioned range, comparing the Flagship mock catalogue (Section 1.2.1) with the *EuclidLargeMocks* realizations. Cosmological parameter inference confirms that replacing

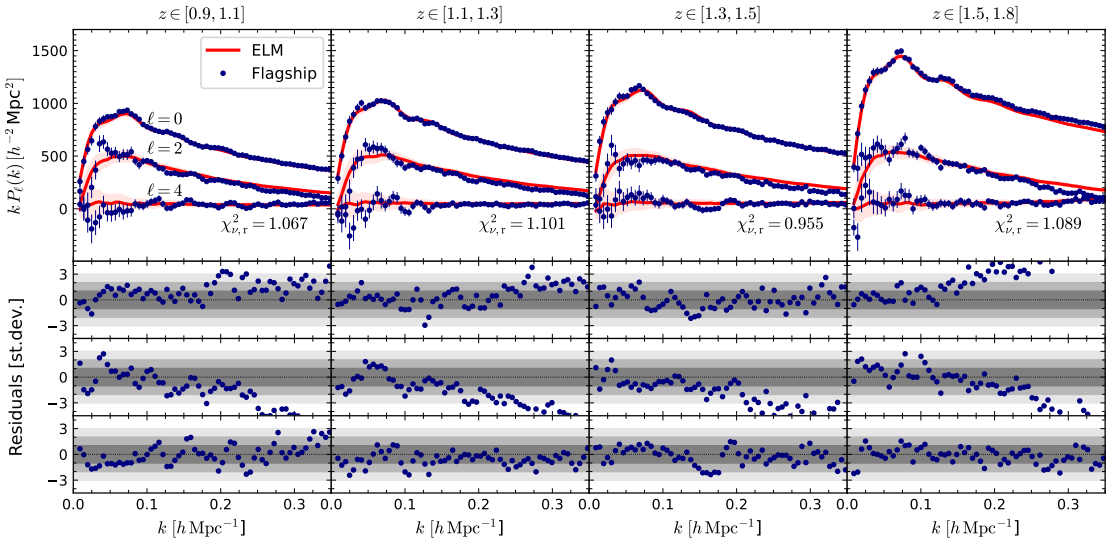


Figure 2.4: Power spectrum multipoles of galaxies with Hα line flux $f_{\mathrm{H}\alpha} > f_0$, shown for the Flagship mock catalog (blue symbols) and the *EuclidLargeMocks* catalog (red curves) in four redshift bins, as indicated at the top of each column. From top to bottom, the panels display the monopole, quadrupole, and hexadecapole moments. The lower panels show the residuals of the Flagship measurements relative to the mean of the *EuclidLargeMocks*, expressed in units of the standard deviation of the latter.

the Flagship mock with a single *EuclidLargeMock* realization yields consistent posteriors, within the expected sample variance. These simulations will support the Euclid Data Release 1 (DR1) galaxy clustering analysis.

I was personally involved in this work, contributing in particular to the transition of the code from version 5.0 to the latest version 5.1. My work focused on improving CPU parallelization of the collapse-time calculation via OpenMP, fixing the OpenMP parallelization of the FFT computation using pFFT, cleaning and optimizing the collapse time module to enhance code readability and maintainability, and updating and modernizing the official GitHub repository of the code [4].

2.5.4 Supermassive black hole seeding

PINOCCHIO has also been employed to investigate the formation of supermassive black holes (SMBHs) from Population III.1 remnants in the very early Universe. In this scenario, SMBH seeds form in isolated, metal-free minihaloes at $z \gtrsim 25$, where the lack of radiative feedback allows the formation of supermassive protostars that collapse into black holes of $\sim 10^5\,M_\odot$.

[4] https://github.com/pigimonaco/Pinocchio

In Singh et al. (2023), PINOCCHIO merger trees were used to trace the cosmological evolution of such seeds down to $z = 0$, predicting their number densities, host halo occupation fractions, and clustering properties. The study shows that the SMBH population forms very early and evolves with only modest merger-driven losses, retaining a characteristic spatial scale inherited from the initial seeding conditions.

In a follow-up work, Cammelli et al. (2025) coupled these PINOCCHIO-based halo merger trees with a semi-analytic galaxy formation model, namely GAEA (Fontanot et al. 2020), to study the joint growth of SMBHs and their host galaxies. The results demonstrate that the Pop III.1 seeding channel is able to reproduce several key observables, such as the local SMBH mass function and the occupation fraction in galaxies, for plausible values of the isolation scale.

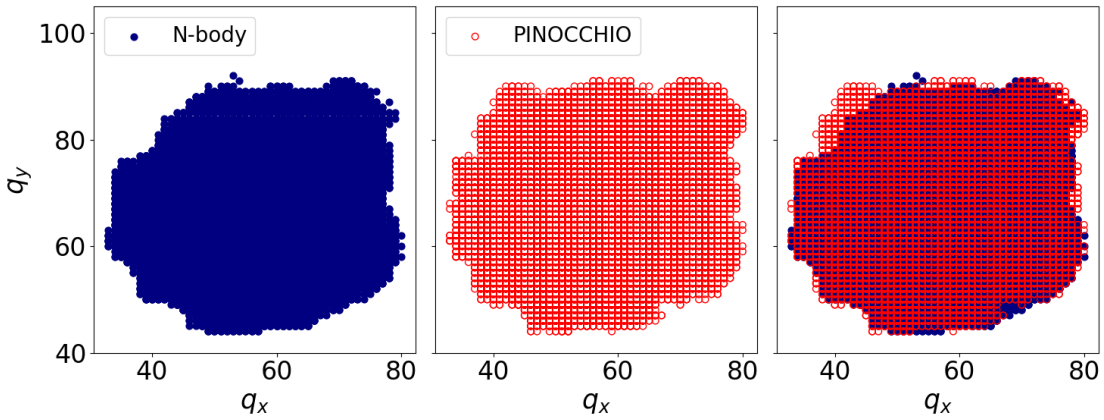
In both works, PINOCCHIO was run on a $60\,\mathrm{Mpc}^3$ box sampled with 4096^3 particles, corresponding to a particle mass of $1.23 \times 10^5\,M_\odot$ and a minimum resolved halo mass of $1.23 \times 10^6\,M_\odot$ (assuming 10 particles per halo). This represents exceptionally high mass resolution for a semi-analytic Lagrangian code, and demonstrates that PINOCCHIO can also be effectively used to investigate early-Universe structure formation and SMBH seeding scenarios.

In a broader context, PINOCCHIO merger trees have also been coupled to semi-analytic galaxy formation models to study the joint evolution of galaxies and active galactic nuclei. An example is the MORGANA model (Monaco et al. 2007), which follows the exchange of mass, metals, and energy between the different baryonic components of galaxies and includes physically motivated prescriptions for gas cooling, star formation, feedback processes, and black-hole accretion. When combined with PINOCCHIO merger trees, MORGANA has been used to investigate the co-evolution of galaxies and AGN, highlighting both the successes and remaining tensions of current models in reproducing observed galaxy and SMBH populations.

2.5.5 Lagrangian particle-level comparison with N-body simulations

To complement the halo-level comparisons outlined earlier, we further assess the predictive accuracy of PINOCCHIO at the particle level. Since PINOCCHIO identifies halos in Lagrangian space, a direct particle-by-particle comparison with an N-body simulation provides a stringent and physically meaningful validation.

To enable this comparison we run a dedicated N-body simulation starting from the same initial conditions (ICs) generated by PINOCCHIO. The test setup consists of cosmological box of side $256\,h^{-1}$Mpc, sampled with 256^3 particles and evolved in an Einstein-de-Sitter cosmology in order to exploit the scale-free and self-similar nature of structure formation



PINOCCHIO	N-body	Overlap	Overlap/PINOCCHIO	Overlap/N-body
28308	28522	22621	0.799103	0.793107

Figure 2.5: 2D projection of the largest halo identified in the N-body simulation (blue) and in PINOCCHIO (red). The table reports the particle counts in each halo, the size of their intersection, and the corresponding overlap fractions.

in this background. By generating the N-body ICs directly from the PINOCCHIO , we ensure a strict one-to-one correspondence between particle IDs in the two simulations. This allows us to identify, for each particle, the halo to which it is assigned in the N-body and in the PINOCCHIO outputs. By cross-matching these halo memberships, we can quantify the degree of overlap between individual halos in the two simulations.

An illustrative example is shown in Figure 2.5, where we display a 2D projection, in Lagrangian space, of the largest halos identified in both simulations: the N-body halo is shown in blue, while the corresponding PINOCCHIO halo is shown in red. The table below the plot reports, for this halo pair, the total number of particles in the PINOCCHIO halo, the number of particles in the N-body halo, the size of their intersection (Overlap), and the overlap fraction. In this specific case the overlap fraction is approximately 80% in both directions, indicating strong consistency in the recovered lagrangian patches.

Repeating this procedure for the full halo population, we compute the particle–level overlap matrix between every PINOCCHIO halo and every N-body halo. The resulting matrix is shown in Figure 2.6. The y–axis reports the masses of the N-body halos, while the x–axis shows the masses of the PINOCCHIO halos. The color scale encodes the overlap level with respect of the PINOCCHIO size.

This matrix allows us to classify the correspondence between halos in the two simulations. Following the criterion introduced in Monaco et al. (2002b), we consider only halo pairs with an overlap fraction greater than 0.3. For each PINOCCHIO halo passing this threshold, we identify the N-body halo with which it shares the largest overlap. If this maximum is also the largest overlap fraction for the corresponding N-body halo, the pair

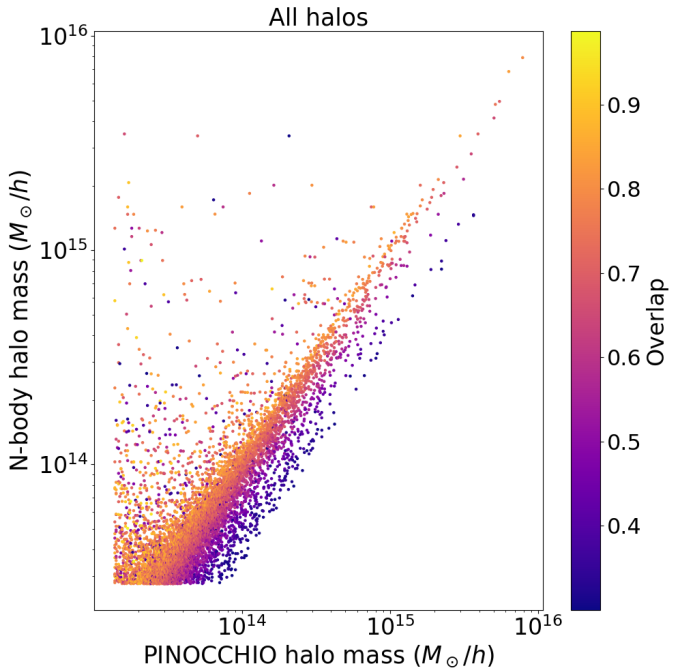


Figure 2.6: Overlap matrix. The axes show halo masses in the two simulations, and the color encodes the fraction of overlapping particles. This matrix is used to classify halo correspondences into *clean assigned* and *split* halos. Clean assignments correspond to PINOCCHIO–N-body pairs where the overlap fraction is mutually maximal; split halos are those for which the overlap is maximal only from the N-body side. The remaining fraction $f_{\rm unassigned} = 1 - f_{\rm clean} - f_{\rm split}$ represents halos with no significant particle overlap.

is labeled as a *clean assigned*. Paired-up halos that are not cleanly assigned are labeled as *split*. This classification is illustrated in Figure 2.7, where the overlap matrix is separated into *clean assigned* halos (left panel) and the *split* halos (right panel).

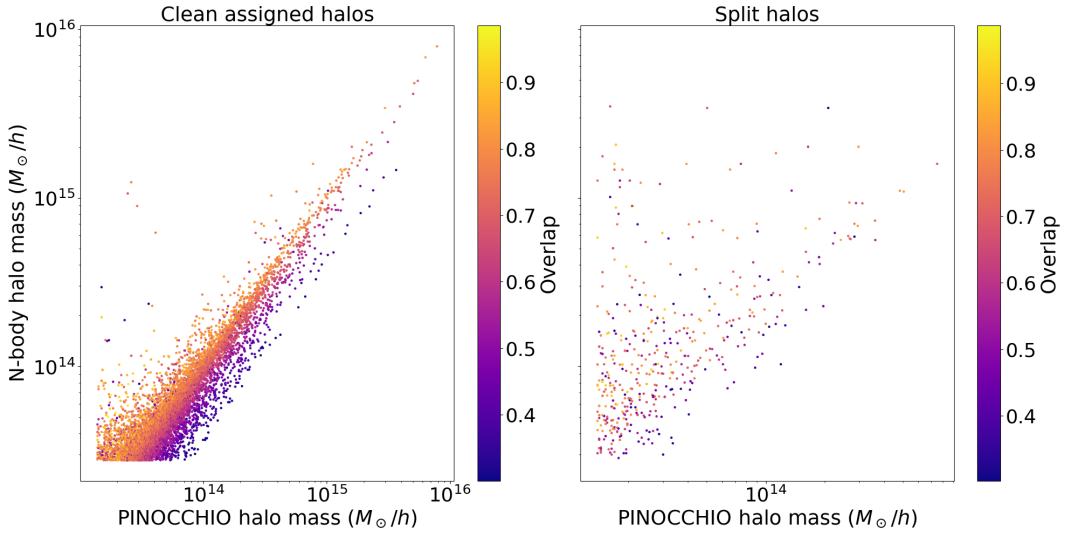


Figure 2.7: Overlap matrix separating cleanly assigned halos (left) and split halos (right).

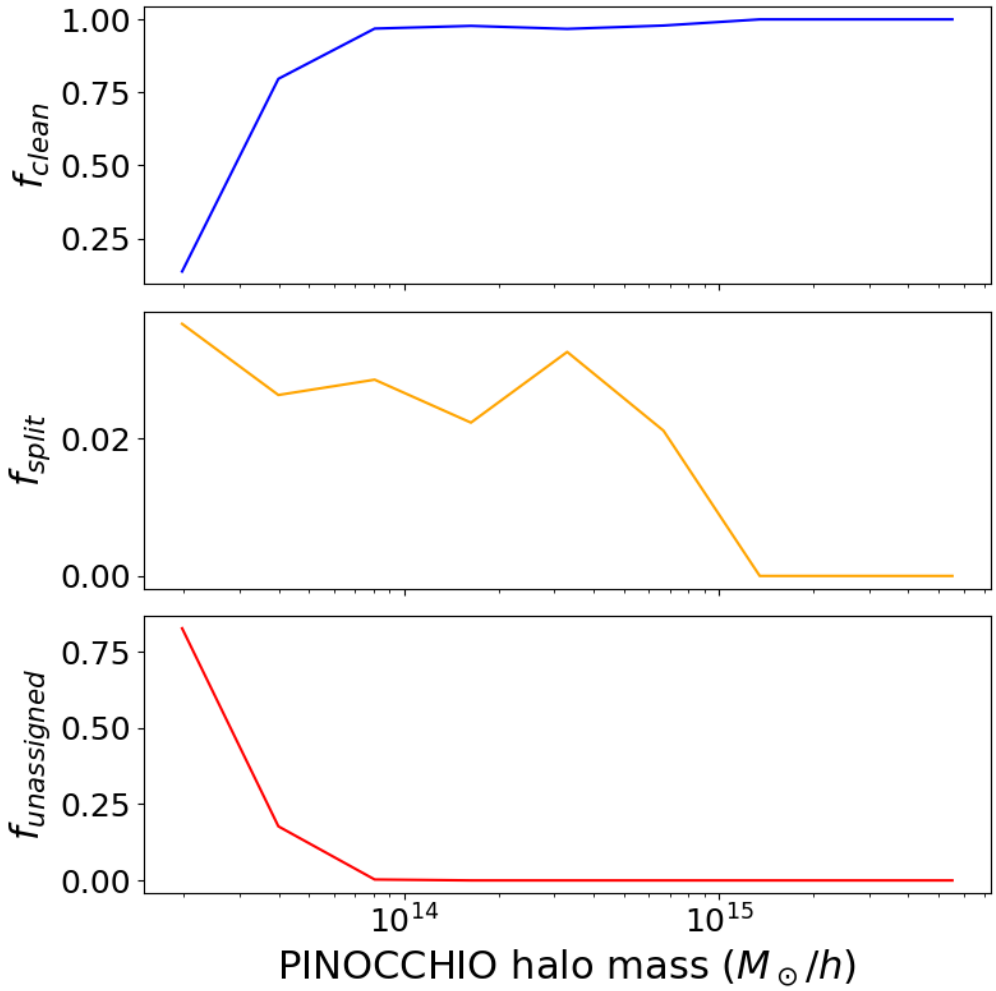


Figure 2.8: Fractions of cleanly assigned halos ($f_{\rm clean}$), split halos ($f_{\rm split}$), and unassigned halos ($f_{\rm unassigned}$) as a function of PINOCCHIO halo mass. For the overall analysis we considered only pairs for which the overlap is mutual higher than 0.3.

Denoting by $f_{\rm clean}$ the fraction of *clean assigned* halos and by $f_{\rm split}$ the fraction of *split* halos, the fraction of halos that are not paired with any N-body counterpart is simply $1 - f_{\rm clean} - f_{\rm split}$. Having classified each PINOCCHIO–N-body halo pair as either *clean*, *split*, or *unassigned*, we can now quantify how these categories depend on halo mass. This provides a global assessment of the reliability of PINOCCHIO across the full mass range.

The resulting trends are shown in Figure 2.8. For halos above $M \gtrsim 4 \times 10^{13}\, h^{-1} M_\odot$, the fraction of cleanly assigned objects already exceeds $f_{\rm cl} \simeq 0.8$, indicating that the vast majority of N-body halos can be unambiguously matched to a unique PINOCCHIO counterpart.

This represents a significant improvement over the early results of Monaco et al. (2002a), where comparable clean–assignment levels were reached only for masses $M \gtrsim 10^{14}\, h^{-1} M_\odot$

(roughly ~ 40 particles in their setup). The improved performance can be attributed to the adoption of 3LPT in the modern version of PINOCCHIO, which enhances the accuracy of collapse-time predictions and displacements field and, consequently, the ability of the Fragmentation module to recover the correct halo progenitor patches. The fraction of split halos, $f_{\rm split}$, remains small across all masses, and decreases further at the high-mass end, indicating that only a minor subset of N-body halos fragment into multiple PINOCCHIO counterparts. The remaining fraction, $1 - f_{\rm cl} - f_{\rm split}$, corresponding to halos without a significant overlap, is very small for massive systems (well below the per-cent level) and increases only toward the low-mass end, where discreteness effects naturally become more relevant.

Chapter 3

GPU porting of collapse time calculation

This Chapter presents the technical developments carried out to accelerate collapse time calculation of PINOCCHIO on modern HPC computing platforms. We begin by outlining the rationale for GPU porting, followed by the design choices that led to select OpenMP as the offloading model. We describe the key characteristics of the HPC systems used in this work, together with the methodology adopted for GPU offloading using OpenMP target directives, including the development of custom GPU-native interpolation routines required to replace the non-portable GSL-based counterparts. The validation of these routines, as well as the resource configurations adopted for both single-node tests and large-scale production runs, is also discussed. We then present the results of the GPU porting, including standalone tests of the custom interpolation kernels and single-node benchmarks of PINOCCHIO runs on both NVIDIA- and AMD-based systems. The impact of GPU acceleration on large-scale production runs is evaluated on the NVIDIA-LEONARDO supercomputer. The conclusions of this study are reported in Chapter 7.

The content of this Chapter is based on the manuscript Lepinzan et al. (2025), submitted to Astronomy & Computing special issue "VSI: HPC in Cosmology and Astrophysics" currently under revision.

3.1 Rationale for GPU porting

The GPU porting effort began from the legacy code publicly available on GitHub at the link https://github.com/pigimonaco/Pinocchio.git. The legacy version is designed exclusively for multi-core CPU architectures, using a hybrid MPI/OpenMP approach. A detailed timing breakdown of a full production run for this CPU version is reported in Table 3.1.

Stages	Time [s]	Fraction [%]
`Initialization`	103.05	3.28
`Fmax`	637.61	20.30
Collapse times (Classic)	*108.70*	*3.46*
`Fragmentation`	2398.90	76.38
Total	3140.89	100.00

Table 3.1: Timing breakdown of the main stages of a full `PINOCCHIO` production run. Reported times correspond to wall-clock runtime, and percentages are given relative to the total runtime. Collapse times are reported explicitly as part of the `Fmax` calculation, since this step is the target of our GPU porting.

Although the collapse time calculation represents only a modest fraction of the total runtime, it was chosen as the first target for GPU acceleration. Thanks to the modular structure of `PINOCCHIO`, computational kernels can be ported incrementally, and the collapse time computation, based on local operations on the Hessian of the gravitational potential described in Section 2.4.2, is an ideal first candidate, being an embarrassingly parallel problem.

Since the original CPU-based implementation relies on the GNU Scientific Library (GSL) [1] for interpolation routines, which does not support GPU offloading, we developed custom GPU-native interpolation routines to enable full device execution. In the following Sections, we will refer to the first approach described in Section 2.4.2, as the *Classic* collapse time calculation, which only requires the spline interpolation of the inverse collapse time $a(D)$. The second, *Tabulated* collapse-time calculation, additionally involves cubic-spline and bilinear interpolation over a precomputed table of eigenvalues. The *Classic* approach corresponds to the configuration adopted in standard production runs, and the timings reported in Table 3.1 refer to this configuration.

3.2 Design choices: OpenMP

To enhance performance and portability, we offload the collapse time module of the code to GPUs using OpenMP target directives. OpenMP (Martineau et al. 2016), with GPU support introduced in version 4.0, provides a directive-based programming model that enables the offloading of computations to GPUs while maintaining cross-platform portability and preserving code readability. Moreover, it avoids the need to introduce C++ dependencies typically required for CUDA (Farber 2011) and HIP [2] models. This approach allows for gradual integration into existing CPU-based code, making it especially

[1] https://www.gnu.org/software/gsl/doc/html/interp.html#
[2] https://rocm.docs.amd.com/projects/HIP/en/docs-develop/what_is_hip.html

suitable for modernizing legacy scientific applications. While the directive-based model facilitates integration within the existing code, achieving optimal performance, particularly in the presence of complex control flow, often requires careful adaptation to GPU-friendly execution patterns.

Through comparative benchmarking on two major supercomputing platforms, LEONARDO (NVIDIA-based) and SETONIX (AMD-based), we demonstrate both the portability and the significant performance gains of our approach. As will be shown and discussed in detail later in this Chapter, we achieve substantial speed-ups from GPU offloading, reaching up to a factor of $\sim 8\times$, depending on the hardware platform. Detailed, system-specific performance results are presented in Section 3.4.3, where it is shown that the AMD-based platform consistently achieves higher speed-ups owing to superior architectural features, as detailed in Section 3.3.3.

Taken together, these results highlight the potential of OpenMP-based GPU offloading to provide effective and portable acceleration of scientific simulations across diverse hardware accelerators.

3.3 Computing platforms

To assess the portability and performance of our GPU porting strategy, we conducted benchmarks on two state-of-the-art supercomputing platforms, described in detail below. These systems, LEONARDO and SETONIX, represent distinct GPU architectures (NVIDIA and AMD, respectively), offering an ideal environment to evaluate cross-platform compatibility and scalability.

3.3.1 NVIDIA-LEONARDO cluster

Leonardo Booster, part of the CINECA [3] Italian national HPC facility, is ranked as 6^{th} in the November 2024 HPCG500 list and 52^{th} in the November 2024 Green500. The platform consists of 3456, 32-cores, Intel Xeon Platinum 8358 CPU nodes equipped with 4 NVIDIA Tesla Ampere 100 GPUs. Each CPU node has 512 GB of DDR4 memory, while each GPU provides 64 GB of HBM2 memory. The CPU and the GPU are interfaced by a PCIe Gen4 interconnection.

The `PINOCCHIO` code was compiled on this system using the NVC/NVC++ compilers v24.3 and OpenMPI library v5.1 was employed.

[3] https://www.hpc.cineca.it/systems/hardware/leonardo/

3.3.2 AMD cluster

Setonix-GPU, part of the HPC facilities at the Pawsey Supercomputing Centre [4] in Perth, Western Australia, is ranked as 45^{th} in the November 2024 HPCG500 list and 20^{th} in the November 2024 Green500 list. The partition comprises 154 compute nodes with 256 GB memory each, equipped with 1 × AMD optimized 3rd Gen EPYC “Trento" 64 cores and 8 GCDs (from 4x “AMD MI250X" cards, each card with 2 GCDs), 128 GB HBM2e. CPU-GPU and GPU-GPU interconnections within each node are guaranteed by the InfinityFabric technology. Inter-node connectivity is established via a HPE Slingshot-11 network fabric, leveraging a DragonFly+ topology to achieve a 200 GB/s bidirectional bandwidth between distinct computing nodes.

The `PINOCCHIO` code was compiled on this system with the amdclang-18 compiler and MPICH-8.1.31 implementation was employed.

3.3.3 Architectural comparison and performance implications

To better understand the performance differences observed in our benchmarks, we provide a detailed analysis of the GPU architectures and their interconnection topologies on both platforms.

The NVIDIA A100 GPU is based on the Ampere architecture (7nm process), featuring:

- **Compute Units**: 108 Streaming Multiprocessors (SMs) with 6912 FP64 CUDA cores
- **FP64 Performance**: 9.7 TFLOPS peak theoretical performance
- **Memory Subsystem**: 64 GB HBM2 with 1555 GB/s bandwidth per GPU
- **Cache Hierarchy**: 40 MB L2 cache, 192 KB L1 cache per SM
- **Tensor Cores**: 432 third-generation Tensor Cores (not utilized in our application)

The AMD MI250X represents a unique multi-chip module (MCM) design (6nm process), where each card contains two Graphics Compute Dies (GCDs):

- **Compute Units**: Each GCD has 110 Compute Units (CUs) with 7040 stream processors
- **FP64 Performance**: 47.9 TFLOPS peak theoretical performance per GCD (95.8 TFLOPS per card)

[4] https://pawsey.org.au/systems/setonix/

- **Memory Subsystem**: 64 GB HBM2e per GCD (128 GB per card) with 3276 GB/s bandwidth per GCD
- **Cache Hierarchy**: 8 MB L2 cache per GCD, 16 KB L1 cache per CU
- **Matrix Cores**: 440 Matrix Core Engines per GCD (not utilized in our application)

The FP64 peak performance differs by approximately *5x* between AMD GCDs (47.9 TFLOPS) and NVIDIA A100s (9.7 TFLOPS). Such a disparity is expected to influence the execution of compute-bound kernels, with its impact examined later in Section 3.5.

3.3.4 CPU-GPU interconnection topology

LEONARDO architecture

The LEONARDO compute nodes employ a traditional PCIe-based topology, organized as follows:

```
CPU Socket 0 (16 cores) <-PCIe Gen4-> GPU 0, GPU 1
CPU Socket 1 (16 cores) <-PCIe Gen4-> GPU 2, GPU 3
```

- **Host-to-Device Interconnect**: Communication between CPU and GPU is established via PCI Express Generation 4 (PCIe Gen4) $\times$16 links, providing a bidirectional bandwidth of approximately 31.5 GB/s per GPU. The associated transfer latency typically ranges from 1 to 2 µs.
- **Non-Uniform Memory Access (NUMA) Considerations**: Each pair of GPUs is directly attached to a specific CPU socket. Consequently, optimized Message Passing Interface (MPI) rank placement is required to minimize cross-socket traffic and maintain balanced memory access performance.
- **Device-to-Device Interconnect**: NVLink 3.0 interconnects paired GPUs, delivering up to 600 GB/s bidirectional bandwidth. However, this link is not utilized in the present embarrassingly parallel workload.

SETONIX architecture

The SETONIX compute nodes are architecturally based on the AMD unified Infinity Fabric (IF) architecture, which governs the intricate interconnectivity.

Each AMD Optimized 3rd Generation EPYC “Trento" CPU (comprising 64 total cores) is logically segmented into eight chiplets (eight cores per chiplet). The specific coupling between CPU chiplets and their respective GCDs is mediated via the IF, as mapped below:

```
Chiplet 0 (8 cores) <-Infinity Fabric-> GCD 4
Chiplet 1 (8 cores) <-Infinity Fabric-> GCD 5
Chiplet 2 (8 cores) <-Infinity Fabric-> GCD 2
Chiplet 3 (8 cores) <-Infinity Fabric-> GCD 3
Chiplet 4 (8 cores) <-Infinity Fabric-> GCD 6
Chiplet 5 (8 cores) <-Infinity Fabric-> GCD 7
Chiplet 6 (8 cores) <-Infinity Fabric-> GCD 0
Chiplet 7 (8 cores) <-Infinity Fabric-> GCD 1
```

- **Bandwidth Specification**: The Infinity Fabric link provides a high-throughput channel with a bandwidth ranging from 36 to 50 GB/s per link, subject to the specific configuration utilized.
- **Data Transfer Latency**: Due to the tightly integrated system design, the interconnect achieves sub-microsecond latency for data transfers.
- **Memory Coherence**: The system implements a unified memory architecture, which inherently supports coherent memory access between the CPU’s host memory space and the GPU’s device memory space.
- **Topology and Locality**: This architecture establishes a natural affinity where each 8-core CPU chiplet maintains a direct, dedicated connection to its paired GCD. This tight integration eliminates conventional cross-socket NUMA access concerns for compute tasks utilizing the local CPU-GCD pair.

3.3.5 Performance implications for our workload

The disparities between the two computing architectures introduce several critical factors impacting the performance and efficiency of the *kernel* workload:

1. **Compute Density**: The AMD MI250X’s higher FP64 throughput (47.9 vs 9.7 TFLOPS) directly translates to faster kernel execution for our compute-bound workload, as confirmed by our roofline analysis showing > 80% of peak utilization.
2. **Memory Bandwidth**: While AMD offers higher memory bandwidth (3276 vs 1555 GB/s), this advantage is less relevant for our compute-bound kernel. However, it does benefit the interpolation table loading phase.

3. **Host-Device Data Transfer Overhead**: The interconnect technology dictates the latency and bandwidth characteristics for host-to-device communication:

   - LEONARDO: The PCI Express Generation 4 (PCIe Gen4) interface constrains peak transfer rates and incurs higher communication latency.
   - SETONIX: The tightly integrated Infinity Fabric architecture effectively minimizes transfer overhead, which is particularly advantageous for workloads characterized by frequent, low-volume data exchanges.

4. **Scalability and Multi-Accelerator Performance**: The intra-node interconnect topology affects performance consistency under full utilization:

   - LEONARDO: The 4 GPU (8 GPU) per node configuration, relying on a PCIe root complex, is prone to interconnect congestion and performance degradation when all accelerators attempt concurrent data movement operations.
   - SETONIX: The 8 GCD configuration, leveraging dedicated CPU-GPU Infinity Fabric links, facilitates more consistent performance scaling under maximum node load.

5. **Toolchain Maturity and Optimization Efficacy**: The state of the compiler ecosystem introduces a developmental factor:

   - NVIDIA: Benefits from a highly mature compiler toolchain (NVC++ 24.3), offering established, extensive optimization for scientific HPC workloads.
   - AMD: Employs a newer toolchain (amdclang-18). Despite the last improvements, its overall optimization maturity and stability may not yet be equivalent to the NVIDIA standard for all compiling scenarios.

The combination of superior FP64 performance, better host–device integration, and our code segment compute-bound nature suggests that the AMD platform may exploit its theoretical advantages more effectively. The performance implications of these characteristics are discussed in Section 3.5.

Overall, this architectural analysis underscores the importance of portable programming models such as OpenMP, which enable scientific applications to leverage the strengths of diverse GPU architectures without platform-specific optimizations.

3.4 Offloading strategy and resource configuration

Our effort targets the second step of PINOCCHIO, described in Section 2.4.2, specifically the calculation of the collapse times at each grid point (hereafter we refer to it as the target *kernel*, in both its *Classic* and *Tabulated* variants), after the Hessian of the potential has been computed. These calculations are inherently independent, making them well-suited for GPU offloading by assigning the computation at each grid point to an individual GPU-thread.

However, the legacy *kernel* logic introduces conditional branches, which can lead to divergent execution paths among GPU threads. It is well established that thread divergence degrades GPU performance by reducing the throughput. To mitigate this issue, we eliminate

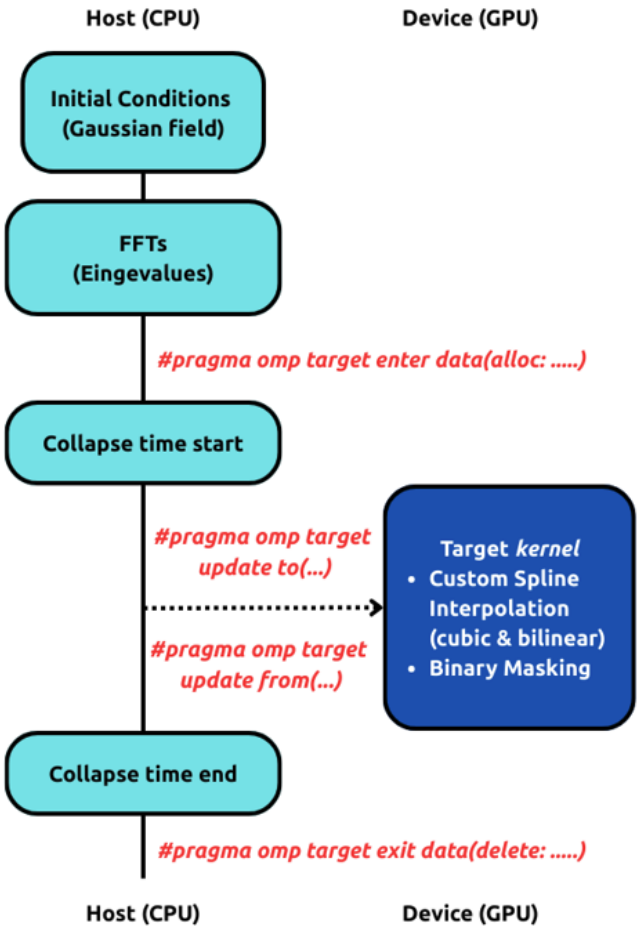


Figure 3.1: Overview of the GPU offloading strategy used in the PINOCCHIO collapse time kernel. Host-side memory is allocated and selectively transferred to the GPU using OpenMP data mapping directives. The target *kernel*, including the custom GPU-native interpolation routines and binary-masked control flow, runs entirely on the device. Only minimal, necessary data is transferred back to the host, improving performance by reducing memory traffic.

the conditional branches by adopting a masked-based approach (i.e. using "predicated execution" to avoid branching entirely. The compiler is very good at this, and often a simple if statement can be compiled to use predicated instructions). This ensures that all GPU threads execute the same instructions at the warp/wavefront level, thereby improving computational efficiency. In addition, we reorganized the data layout into a struct-of-arrays (SoA) format, so that consecutive GPU threads access consecutive elements of each array. This improves memory coalescing and further increases throughput.

We manage host-device memory transfers explicitly using standard OpenMP data mapping directives as shown in Figure 3.1. In particular, we use `pragma omp target enter data map(alloc:...)` to allocate required quantities on the device at the beginning of the computation, followed by targeted updates using `pragma omp target update` to transfer only the necessary data between the host and device. This approach avoids unnecessary memory movement and ensures efficient data locality during *kernel* execution. Once the computation is complete, the GPU memory is explicitly released using `pragma omp target exit data map(delete:...)`.

3.4.1 Custom interpolation routines implementation

A critical part of the *kernel* calculations involves interpolating precomputed quantities required to estimate the collapse time. In the legacy CPU-based implementation, this is handled by GSL library, using cubic spline and bilinear interpolation routines.

This is necessary because, as discussed Section 2.4.2, the linear growth rate $D(a)$ as a function of scale factor a is computed by numerically solving an ODE over a predefined grid. These results are then stored and accessed via cubic spline interpolation, with the spline object constructed once and reused throughout the simulation. Additionally, one option for computing collapse times involves tabulating their values on a grid of the eigenvalues λ_i. The final collapse time is then obtained through interpolation using a mixed scheme: spline interpolation along the dimension with the most complex variation (δ), and bilinear interpolation in the other two dimensions.

For the interpolation of $D(a)$, building the spline on the CPU and transferring it only once (*Classic kernel*) to the GPU is not performance-critical. However, in the second case, where the interpolation table must be updated at every smoothing radius iteration (*Tabulated kernel*), constructing the spline on the host and repeatedly transferring data to the GPU would become a major performance bottleneck. This repeated movement of large data tables would severely impact throughput, making GPU-resident interpolation routines essential for maintaining efficiency.

Since GSL does not support GPU offloading, we developed two custom GPU-native

interpolation routines, implemented using OpenMP target constructs. These routines are fully device-resident and designed to avoid host-device transfers during kernel execution. Their numerical behavior closely reproduces the GSL results, as verified in a standalone toy code, while also providing substantial performance improvements. A detailed analysis of accuracy and performance is presented in Section 3.5.

3.4.2 Custom interpolation validation against GSL

In order to validate the accuracy of the custom GPU-native interpolation routines, we construct a synthetic dataset independent from the main `PINOCCHIO` workflow.

In particular, we use simple analytical functions with known behavior: a 1D function $y(x) = x^2$ for validating the cubic spline routine, and a 2D function $z(x, y) = x^2 + y^2$ for validating the bilinear interpolation. For the cubic spline test, we use 512 randomly distributed points for building the spline object, consistent with what is typically used within the `PINOCCHIO` code. For the bilinear case, we construct a 64×64 interpolation table, representative of the structure used in `PINOCCHIO` runs, where each grid point is associated with a spline of size 128.

The tabulated input points are randomly distributed, reflecting the non-uniform sampling in realistic collapse time tables and highlighting the robustness of our interpolation routines. In both cases, we generate a set of randomly distributed evaluation points, ranging from 100 to 10^7, with the upper end corresponding to the number of interpolation calls typically encountered in a small-box simulation (e.g., 256^3 particles).

We then compare the interpolated values produced by the GPU-native routines against those obtained using the original GSL-based CPU implementation. The comparison is quantified by computing the mean residuals across all evaluation points. This procedure allows us to confirm that the numerical differences introduced by the GPU implementation are negligible with respect to the original GSL routines.

In addition to validating numerical accuracy, we also benchmark the performance of the custom GPU-native interpolation routines against their GSL counterparts as a function of the number of evaluation points. This allows us to estimate the threshold at which GPU acceleration becomes advantageous.

Validation results and performance tests for the cubic spline and bilinear interpolation routines are presented respectively in Section 3.5.1 and Section 3.5.2. These tests are performed on the NVIDIA platform only, as their purpose is to validate the correctness and assess the relative performance of the GPU-native routines, without requiring cross-platform comparison.

3.4.3 Porting benchmarks: single-node

To ensure a meaningful comparison across computing platforms, we perform initial benchmarks using the same number of computational units, hereafter referred to as `CUs`. Given the heterogeneity of the platforms in terms of the underlying hardware architecture, we define the `CU` to be $\frac{1}{4}$ of the node, corresponding to eight cores for CPU and one GPU for LEONARDO, and sixteen cores for CPU and two Graphics Compute Dies (GCDs)[5] for SETONIX. This assumption enables a uniform metric for scaling analysis across different configurations.

The choice of using $\frac{1}{4}$ of the node per `CU` is motivated by the following considerations:

- On LEONARDO, each node hosts four GPUs, and the performance is maximized when each GPU is managed by an MPI process using eight CPU cores (i.e. spawning eight OMP threads);
- On SETONIX each GCDs is physically paired with a chiplet of eight CPU cores, supporting a natural 1:8 GCD-to-core mapping.

While the current *kernels* implementation allows multiple GPUs to be driven by a single MPI task when needed, this definition of `CU` provides a practical baseline for fair performance comparisons between CPU and GPU configurations.

All single-node tests on LEONARDO are performed using one computational node, with up to 4 MPI processes (corresponding to 32 cores for CPU runs and 4 GPUs for accelerator runs). All single-node tests on SETONIX are performed using one computational node, with up to 8 MPI processes (corresponding to 64 cores for CPU runs and 8 GCDs (4 GPUs) for accelerator runs). This setup ensures that performance measurements are not affected by inter-node communication.

Single-node benchmarks are performed for both the *Classic* and *Tabulated* collapse time *kernels*. In all cases, the *kernel* time-to-solution is defined as the wall-clock time required for the actual computation. For GPU runs, this includes both the device execution time and the overhead from host-device data transfers, providing a realistic assessment of end-to-end performance.

3.4.4 Production run setup

While the single-node benchmarks focus on a controlled environment for platform comparison, production runs adopt a different configuration optimized for the overall application

[5] A Graphics Compute Die (GCD) is essentially an independent GPU chip; the MI250X integrates two such dies into a single package.

performance. Specifically, we increase the number of MPI tasks per node and reduce the number of OpenMP threads per task. This configuration better matches the scaling characteristics of other parts of `PINOCCHIO`, such as the FFT computations and the `Fragmentation`, which benefit from a finer-grained domain decomposition.

In these tests, we run with 16 MPI tasks per node (4 per GPU, illustrating the flexibility of the MPI–GPU mapping while preserving correctness), each using 2 OpenMP threads, for a total of 2880 MPI tasks across 180 compute nodes (using 32 cores per node).

These production-scale benchmarks are performed exclusively on the NVIDIA platform, where such a large allocation of resources is available, and are carried out with the *Classic kernel*, which corresponds to the standard configuration in production runs not involving modified gravity (which lies beyond the scope of this work).

To ensure that the GPU offloading does not impact the scientific correctness of the results, we compare the distribution of collapsed particle, as a function of redshift z, produced by the GPU and legacy CPU implementations. This quantity is a direct output of the code and encapsulates the physics modeled by the *kernel*.

3.5 Collapse time porting results

This Section presents the validation and performance results of the GPU porting. We validate the GPU-native interpolation routines, analyse single-node performance (Sections 3.3–3.4.3), provide a roofline analysis of the *Classic kernel*, and evaluate its impact in full production runs (Section 3.4.4). All results rely on single-run benchmarks, with stable timings ensured by repeated evaluations over multiple smoothing radii (Section 2.4.2).

3.5.1 Cubic spline validation

To evaluate the accuracy and performance of the cubic spline interpolation, we test the GPU-native routine against the GSL implementation using the synthetic 1D function described in Section 3.4.2.

Figure 3.2 summarizes the accuracy results. The left panel displays the interpolated values obtained from both the GPU-native and GSL-based routines, alongside with the analytical reference function. For visual clarity, only 35 randomly distributed evaluation points are displayed. As shown, the GPU custom implementation closely follows both the analytical reference and the well-established GSL results, confirming the numerical accuracy of our custom routine. The right panel shows the histogram of the residuals between the GPU and GSL interpolations. The dashed line marks the average residual,

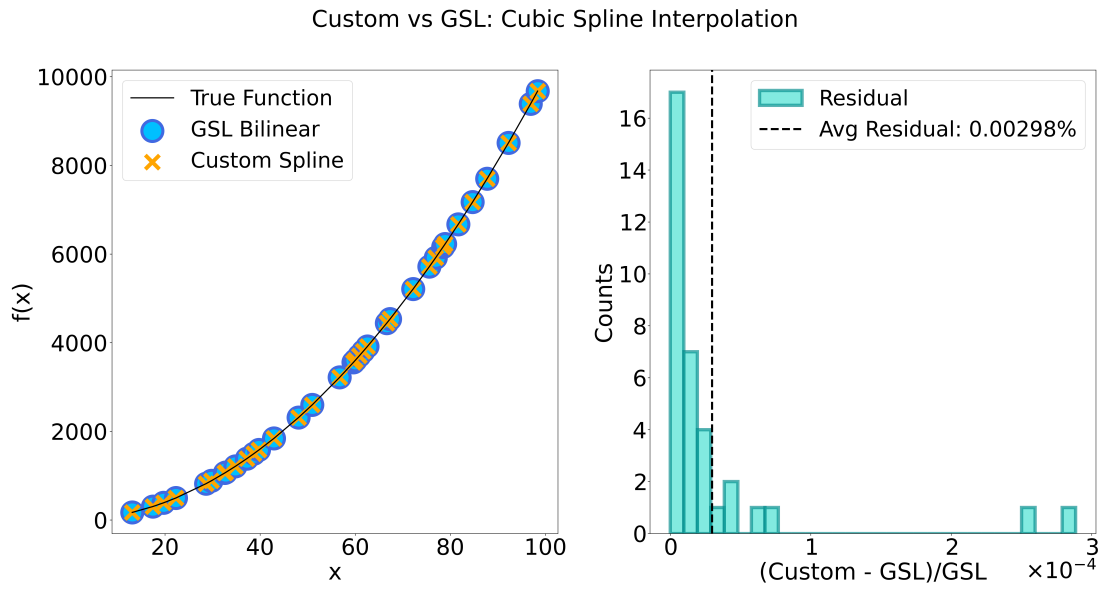


Figure 3.2: Left panel displays the comparison between the GPU-custom cubic spline interpolation routine, the GSL implementation, and the analytical 1D reference function described in Section 3.4.2. A total of only 35 evaluation points are shown for clarity. Right panel shows the histogram of the residuals between the GPU-custom and GSL interpolations, aggregated over the 35 evaluation points. Residuals are shown in absolute units, while the dashed line indicates the average residual expressed as a percentage.

which remains at the level of sub-percent level ($\sim 0.003\%$), indicating excellent agreement between the two methods. A slight systematic overestimation is observed, with residuals consistently positive across the evaluation points. This mild systematic, likely introduced by the numerical treatment of second derivatives in the custom GPU implementation, remains well below the percent level. As will be shown in Section 3.5.5, the impact of these deviations on the final scientific output is negligible.

In addition of validating numerical accuracy, we evaluate the performance of the custom GPU-native cubic spline interpolation routine against the GSL implementation in Figure 3.3. The GPU-custom interpolation is shown in blue, while the GSL-based implementation is shown in red. At small numbers of evaluations, the CPU implementation remains faster due to lower memory management overheads. However, as the number of evaluation points increases, the GPU routine scales more efficiently, becoming faster beyond approximately 10^4 evaluations and reaching a speed up factor of $\sim 12x$ relative to the GSL. The GPU timings include memory transfers between the host and device, providing a realistic estimate of performance in practical settings. To avoid unrepresentative timing fluctuations, each interpolation benchmark was performed by looping the routine 10 times over the same data and averaged. This approach ensures that timing results reflect a stable and representative performance not affected by transient runtime variations. Such

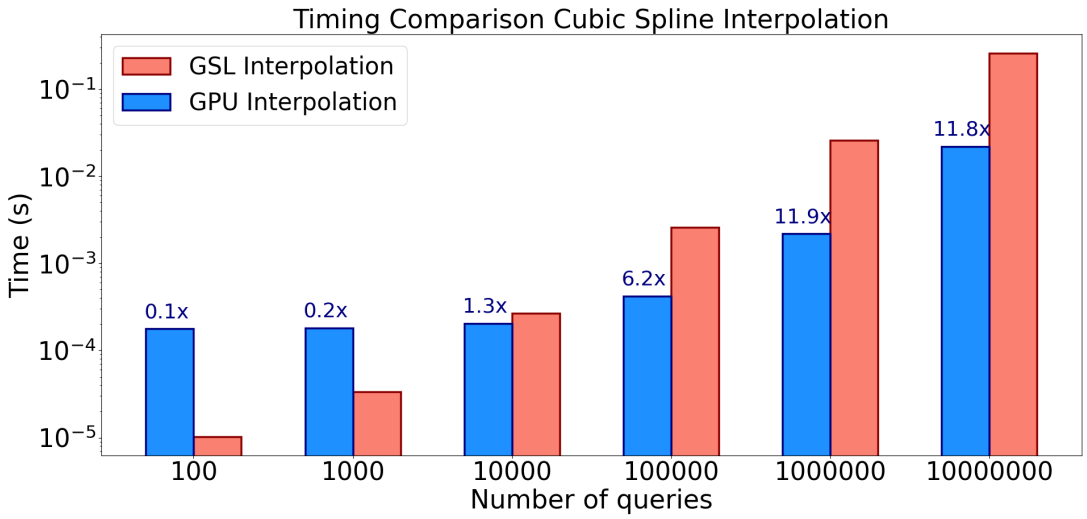


Figure 3.3: Comparison of interpolation wall-time between the GPU-native cubic spline routine and the GSL implementation, as a function of the number of evaluation points. GPU timings include memory transfers between the host and device. Speedup values relative to the GSL implementation are reported above each GPU bar.

performance gains become particularly relevant in `PINOCCHIO` runs, where collapse times must be evaluated for billions of particles across multiple smoothing radii.

3.5.2 Bilinear validation

Following the same approach adopted for the cubic spline interpolation described in Section 3.5.1, we assess the numerical accuracy and performance of the GPU-native bilinear interpolation by comparing its output with the GSL implementation across a set of randomly distributed evaluation points.

The left panel of Figure 3.4 shows the interpolated values from both methods, compared against the ground truth values derived from the analytical function, demonstrating visually consistent results. The right panel presents the histogram of the residuals between the two methods. The numerical accuracy ($\sim 0.003\%$) and the mild overestimation trend is consistent with what is observed for the cubic spline interpolation.

Figure 3.5 shows the interpolation performance comparison, following the same setup and evaluation strategy described for the cubic spline in Figure 3.3. Interestingly, in contrast to the cubic spline case, the GPU-custom bilinear interpolation routine exhibits nearly constant execution time across varying evaluation size. This behavior suggests efficient parallel utilization and early kernel saturation, where the GPU is able to fully occupy its compute units even at modest workloads. However, due to the higher per-query complexity

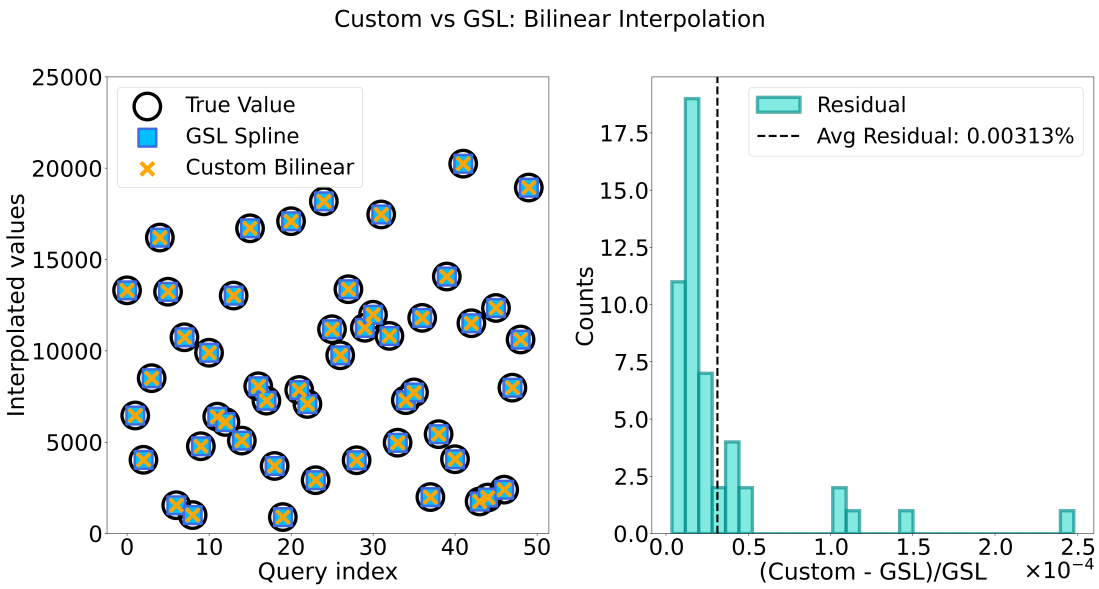


Figure 3.4: Same as Figure 3.2 but for the bilinear interpolation routines. In this case 50 evaluation points are shown in the left panel.

of the bilinear interpolation, involving four spline evaluations per call and more scattered memory access patterns, the speedup relative to the GSL implementation becomes evident only at larger evaluation point ($\geq 10^6$). This scaling behavior illustrates the efficiency of the GPU implementation in scenarios involving heavy interpolation workloads.

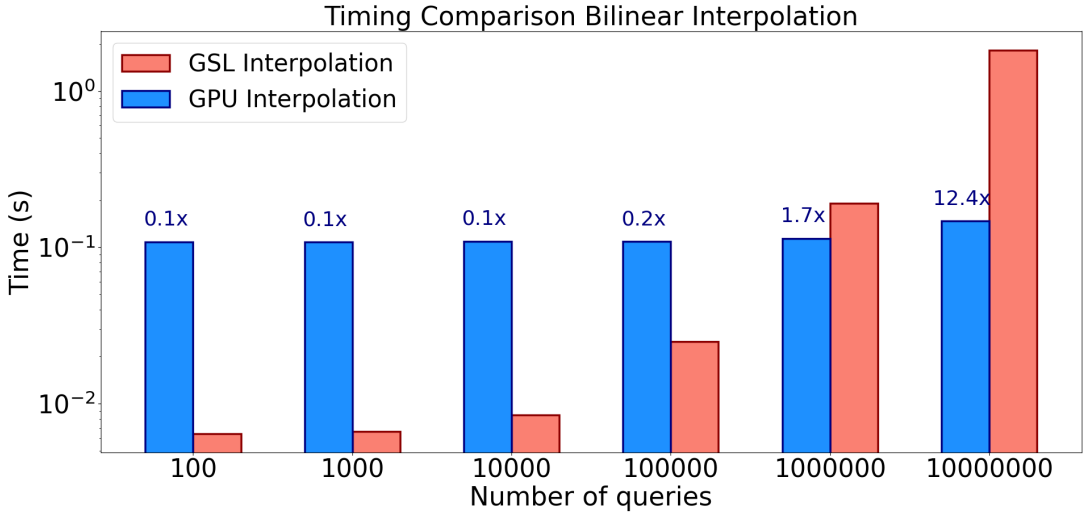


Figure 3.5: Same as Figure 3.3 but for the bilinear interpolation routines.

3.5.3 Single-node speedup comparison

We begin our performance analysis of the PINOCCHIO *kernels* with strong-scaling benchmarks on a single node, comparing GPU and CPU implementations across different platforms and grid sizes. This analysis is performed for both collapse time computation methods described in Section 2.4.2, namely the *Classic* and *Tabulated kernels*. CPU speedup is defined as the ratio between the time-to-solution using 1, 2, and 4 CPU CUs and that using a single CPU CU. GPU results are normalized in the same way for visual comparison, but labels above GPU bars indicate the speedup relative to the CPU implementation using the same number of CUs. This enables a direct, per-CU comparison of GPU versus CPU performance at each scale.

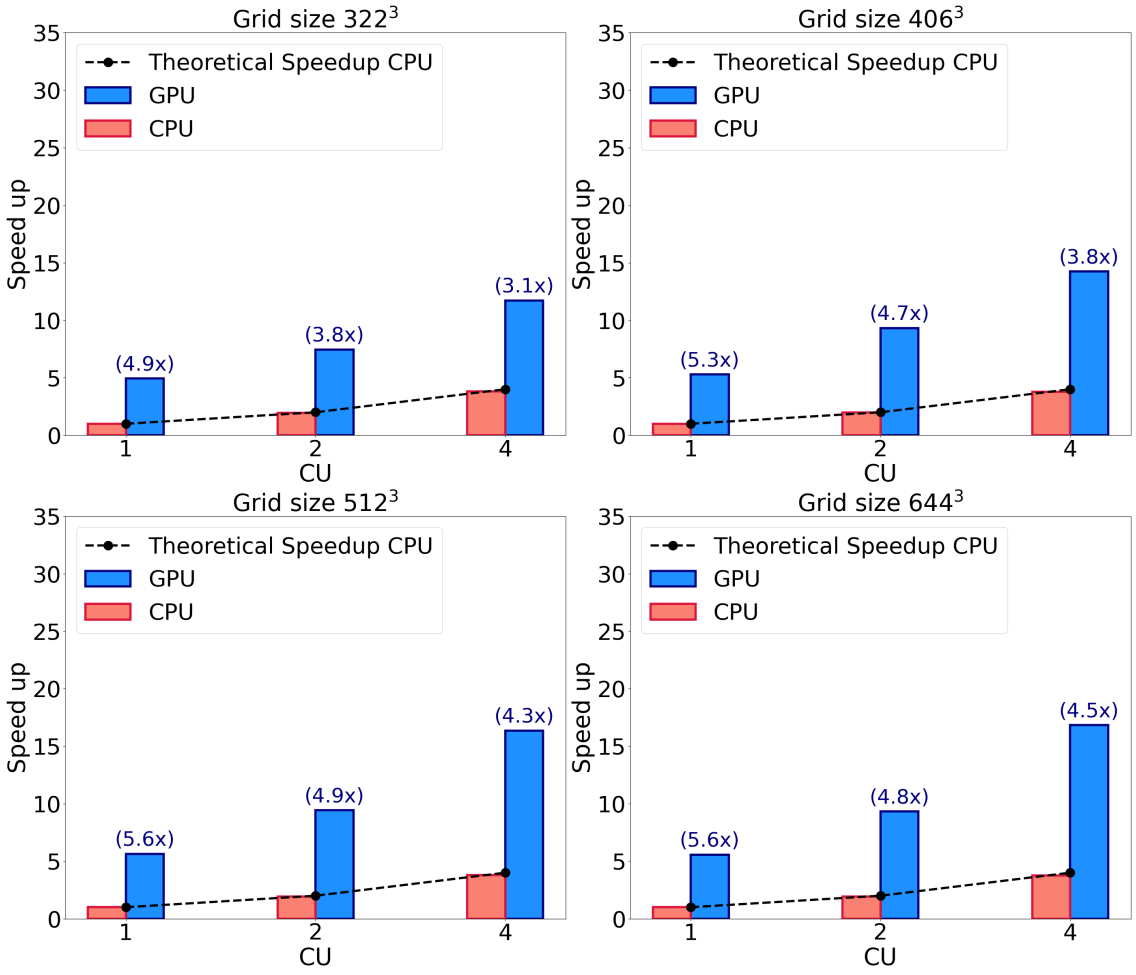


Figure 3.6: Speedup in strong-scaling tests for NVIDIA platform for different grid sizes, using the *Classic kernel*. The plots display performance as grid resolution increases from the upper-left to the lower-right. Bar heights represent the normalized speedup relative to the 1 CPU CU baseline. Labels above GPU bars indicate the speedup of the GPU implementation relative to the CPU implementation at the same number of CU.

Figure 3.6 and 3.7 illustrates the results for NVIDIA and AMD platforms respectively, with increasing grid resolution moving from the upper-left to the lower-right for the *Classic kernel*. For completeness, the CPU performance on both platforms is included, alongside with the expected theoretical speedup for CPU calculations. The results validate the parallel efficiency of the *Classic* kernel computations. CPU speedup closely follows the theoretical expectation across all grid sizes, confirming that the workload scales efficiently and evenly across the available CU. Additionally, the GPU implementation consistently achieves a speedup at least *4x* higher than the CPU across all CUs.

This consistent gain confirms the suitability of GPUs in handling massively parallel workloads such as the collapse time computation. Notably, the advantage of GPU offloading becomes more pronounced at higher grid resolutions (512^3 and 644^3), where the increased computational load leads to better GPU utilization. At smaller grid sizes, the workload may be insufficient to fully saturate the GPU, which limits the achievable speedup.

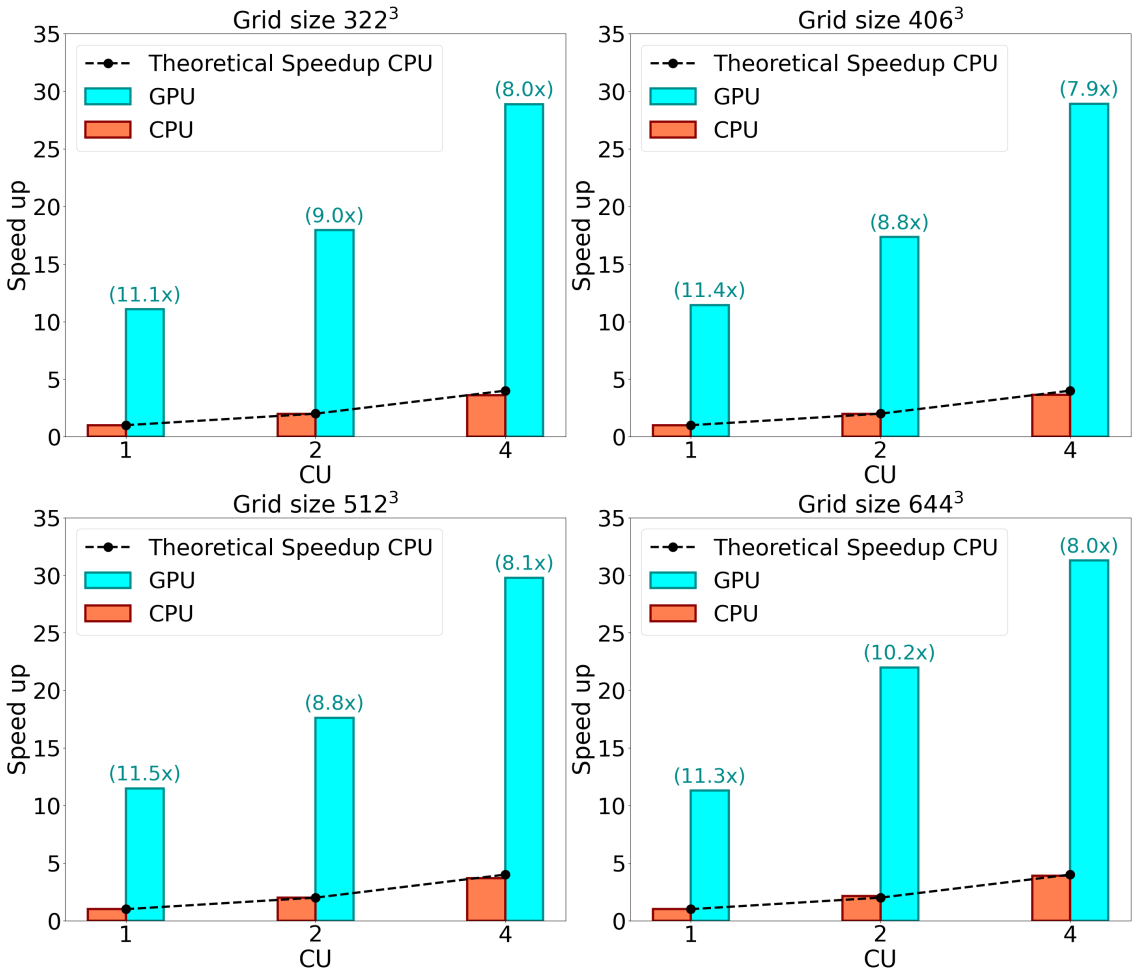


Figure 3.7: Same as Figure 3.6 but for the AMD platform.

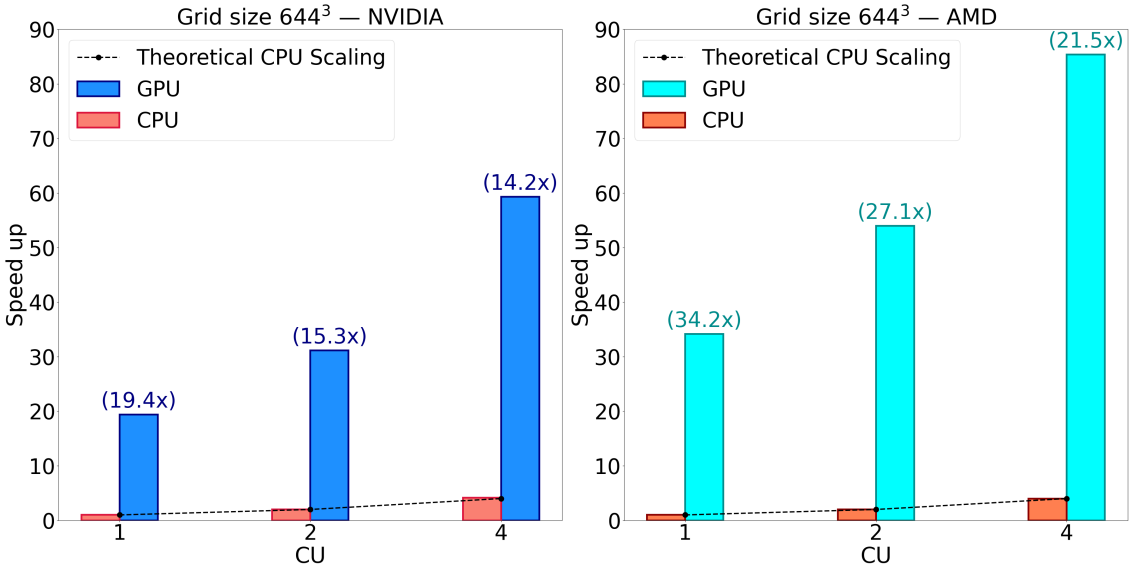


Figure 3.8: Same as Figure 3.6, but using the *Tabulated kernel*. Results are shown for the NVIDIA platform (left) and AMD platform (right), and refer exclusively to the 644^3 grid size.

For this reason, in the case of the *Tabulated kernel*, we report performance results only for the 644^3 grid, where the computational intensity is sufficient to highlight the scaling behavior and hardware differences. The results are shown in Figure 3.8, where the left panel presents the NVIDIA case, and the right panel corresponds to the AMD platform.

The larger GPU speedup observed for the *Tabulated kernel* (up to *4*× higher than in the *Classic kernel*) is primarily due to the absence of OpenMP parallelization in the corresponding CPU implementation. While both tests use the same number of CPU cores per CU, the *Classic kernel* benefits from hybrid MPI + OpenMP parallelism, whereas the *Tabulated kernel* relies solely on MPI. As a result, the CPU baseline is less optimized in the latter, amplifying the apparent GPU performance gain. It is also worth noting that the *Tabulated* approach is mainly used in runs involving modified gravity, as explained in Section 2.4.2, which are beyond the scope of this work. For this reason, our large-scale production tests in Section 3.5.5 focus on the *Classic* variant only.

Notably in both approaches cases, we also observe that the speedup achieved on the AMD platform is roughly twice that of the NVIDIA platform. As discussed in Section 3.3.3, the combination of approximately *5x* difference in FP64 peak performance (47.9 vs 9.7 TFLOPS) between AMD GCDs and NVIDIA A100s, better host-device integration, and our kernel's compute-bound nature allows the AMD platform to more effectively utilize its theoretical advantages. These findings confirm that our GPU implementations deliver both substantial performance improvements and excellent portability. The consistent scaling behavior across NVIDIA and AMD platforms highlights the robustness of our OpenMP

offloading approach for heterogeneous computing environments.

3.5.4 Roofline analysis

To quantitatively assess the computational efficiency of our GPU porting strategy, we perform a roofline analysis on the NVIDIA platform using NVIDIA Nsight Compute tool[6]. This allows us to evaluate how close the GPU *Classic kernel* operates to the theoretical hardware performance limits and to confirm that thread divergence has been effectively mitigated by the masked-based implementation described in Section 3.4.

This analysis was performed on the NVIDIA platform only, and for the *Classic kernel*, which reflects the setup adopted in our production simulation runs. The *Tabulated kernel*, while also GPU-accelerated, is primarily used in simulations involving modified gravity, which are beyond the scope of the present work. For this reason, performance profiling of the latter was not pursued here. Nevertheless, similar behavior is expected, as the *Tabulated kernel* builds upon the *Classic* one with the addition of a bilinear interpolation routine, which is lightweight and free of conditional branching.

Figure 3.9 illustrates the results. The *Classic kernel* achieves over 80% of the theoretical peak performance in FP64, indicating efficient use of available GPU resources. The

[6] https://developer.nvidia.com/nsight-compute

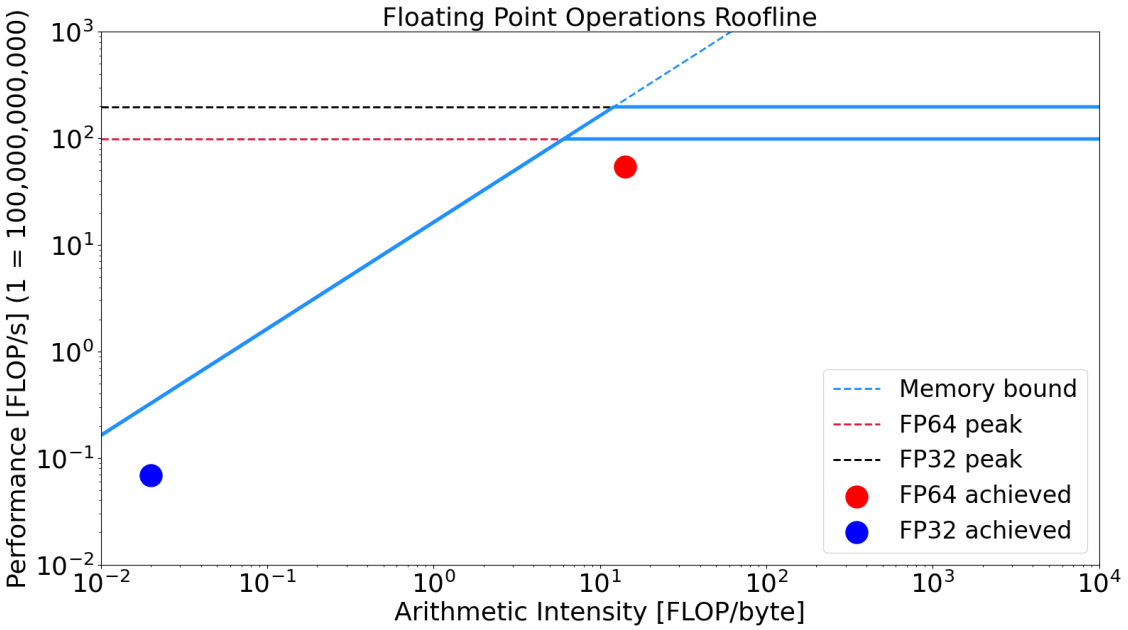


Figure 3.9: Roofline analysis of the GPU-accelerated *Classic kernel* on the NVIDIA platform. The *kernel* achieves over 80% of the theoretical FP64 peak performance, operating in the compute-bound regime near the performance plateau (red dot).

measured arithmetic intensity places the kernel well within the compute-bound regime (red dot in Figure 3.9), close to the performance plateau, confirming that the GPU is effectively saturated and that the implementation makes good use of available compute throughput. This high efficiency aligns with the speedup trends observed in our timing benchmarks and further validates the quality of the GPU porting.

3.5.5 Production run

While benchmarks presented in Section 3.5.3 focused on controlled single-node conditions to ensure platform comparability, production-scale runs involve a configuration tailored to the overall performance of the entire `PINOCCHIO` application. In these cases, domain decomposition, FFT scalability and the halo construction workflow play a critical role, motivating a shift toward a more MPI-intensive configuration. To reflect realistic usage scenarios, we adopt the hybrid parallel configuration described in Section 3.4.4. These production-scale tests were performed on the NVIDIA platform, which currently provides access to a sufficient number of compute nodes required for large-scale simulations.

We evaluate the GPU-accelerated version of the *Classic kernel* within the full production workflow, measuring the time-to-solution. The direct comparison with the CPU implementation shows a consistent performance gain of approximately $6x$, in line with the results observed in the controlled single-node benchmarks (Section 3.5.3). This confirms that the benefits of GPU offloading persist under realistic, large-scale workloads, validating the scalability of the implementation. In practical terms, this translates into a net saving of approximately 100 seconds per simulation. Given that typical cosmological campaigns involve thousands of `PINOCCHIO` runs, this results in a cumulative saving of over 160000 Standard-h, a significant reduction in computational cost that enables more efficient use of HPC resources. Such a saving also implies a non-negligible reduction in energy consumption, as will be presented and discussed in detail in Section 4.4.

In addition to performance, we validate the scientific consistency of the GPU implementation by comparing the distribution of collapsed particles, as a function of redshift z, obtained in the GPU and CPU runs. As shown in Figure 3.10, the two distributions are in excellent agreement, confirming that the ported *Classic kernel* accurately reproduces the expected collapse statistics. The relative residuals remain within $\pm 1\%$ across most of the redshift range, demonstrating the robustness of the GPU interpolation scheme. For $z \gtrsim 0.6$, the agreement is consistently better than 1%. At $z \lesssim 0.5$, the GPU version exhibits a small systematic underestimation in the number of collapsed particles, reaching up to $\sim 2.5 - 3\%$. Interestingly, this trend is interrupted by a localized overestimation around $z \sim 0.5 - 0.6$, where the GPU result overshoots the CPU reference. This anomaly may be

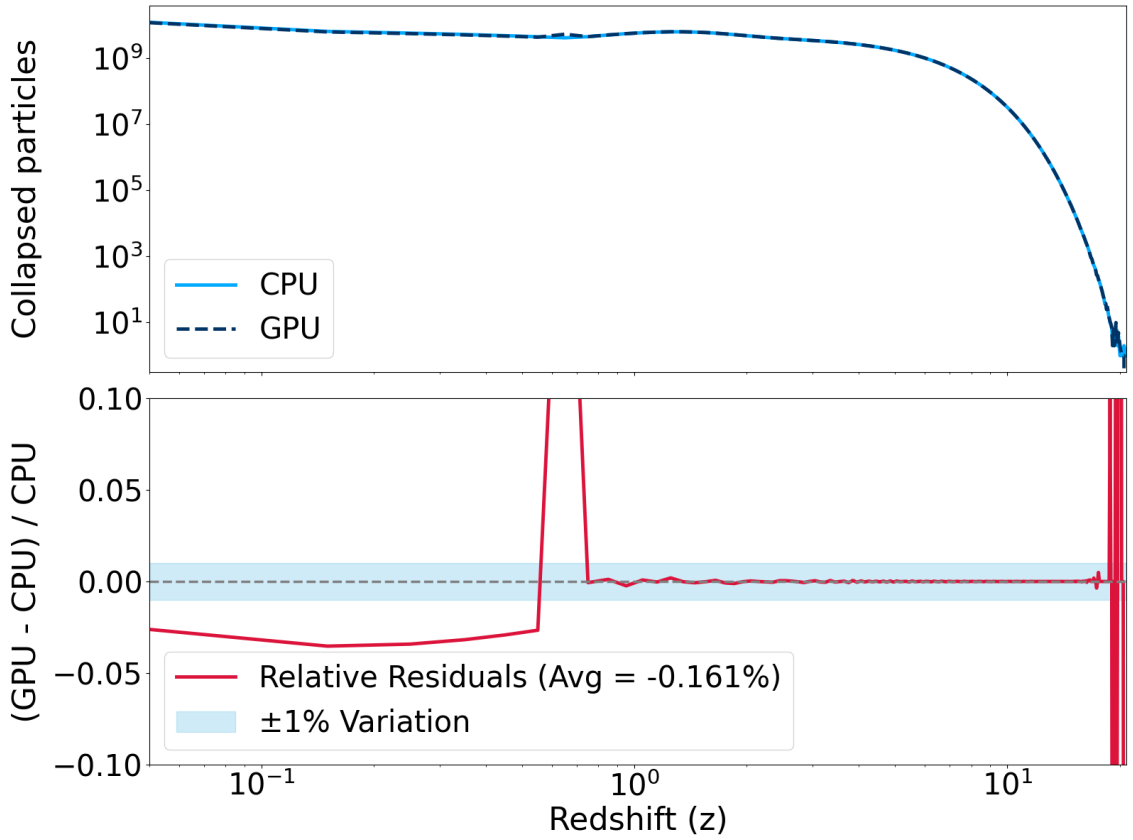


Figure 3.10: Comparison of the number of collapsed particles per redshift bin obtained using the CPU and GPU implementations of the *Classic kernel*. The upper panel shows the total number of collapsed particles as a function of redshift z, while the lower panel presents the relative residuals between the GPU and CPU results. The shaded region highlights the ±1% deviation band and the dashed lined indicates the level of perfect agreement.

attributed to a subtle interpolation artifact in the inverse growing mode near the transition between rapid and slow growth, where the cubic spline response becomes more sensitive to local curvature.

In the current implementation, this interpolation is performed using a custom cubic spline solver where the same calculation performed using the GSL spline interpolation on the CPU does not exhibit the anomaly. This strongly suggests that the effect is not intrinsic to the physical model, but rather related to implementation-specific aspects of the spline construction, such as boundary conditions or numerical conditioning in the tridiagonal solver. To further test this hypothesis, one could compare different spline boundary prescriptions (natural vs. clamped). More specifically, one possible test consists in replacing the natural spline boundary conditions (which impose vanishing second derivatives at the endpoints) with clamped boundary conditions (which instead enforce specified first derivatives at the endpoints). Enforcing physically motivated endpoint slopes

may reduce the sensitivity of the interpolation to artificial curvature suppression near the domain boundaries and could mitigate the observed anomaly.

Importantly, we have confirmed that this interpolation artifact does not propagate to the final halo properties, as the halo mass function. This indicates that the anomaly affects only an intermediate numerical quantity and does not significantly impact the physical outputs of PINOCCHIO. We stress that the custom cubic spline solver was developed to enable GPU execution, as publicly available interpolation libraries with native GPU support are limited. We are therefore currently refining the GPU implementation to improve its numerical robustness while preserving computational efficiency.

At high redshifts ($z \gtrsim 10$), discrepancies are negligible and statistically irrelevant due to the very low number of collapsing particles. Overall, the results confirm the scientific fidelity of the GPU implementation, with only minimal deviations from the CPU reference. Although the scientific outcomes obtained with the GPU version are in very good agreement with those produced by the reference CPU implementation, ongoing work is focused on

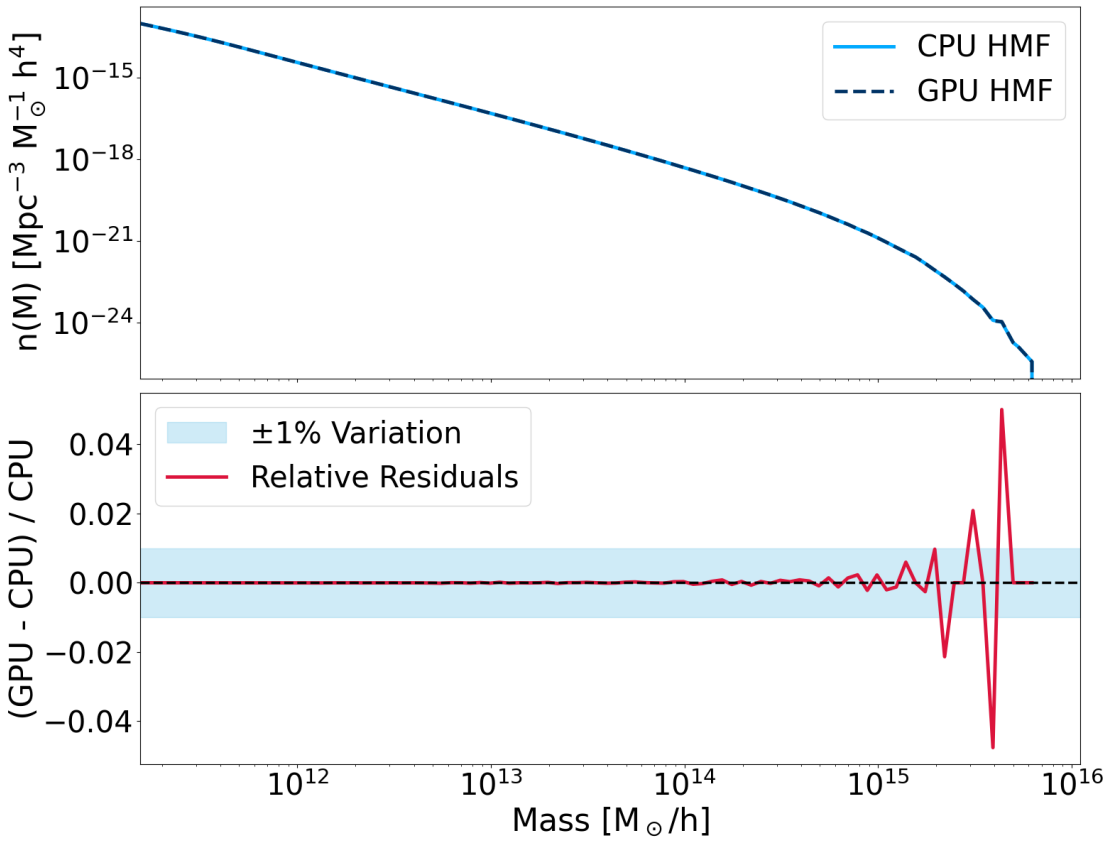


Figure 3.11: Comparison of the HMF obtained from the GPU-accelerated and CPU-based implementations of the PINOCCHIO *Classic kernel* in a full production run. The top panel shows the HMFs from both versions, while the bottom panel displays the relative difference between them.

further mitigating these differences in order to achieve an even closer agreement.

To further validate the scientific consistency of our GPU-accelerated implementation of the *Classic kernel*, we compare the HMFs obtained from both the CPU and GPU versions of `PINOCCHIO` in a full production run scenario as described in Section 3.5.5. Figure 3.11 presents the results. The agreement between the two implementations is remarkably good, with relative differences well below the 1% level across the entire mass range. This confirms that the overall mild underestimation of number of collapse particle ($\sim 0.16\%$) is effectively absorbed by the `Fragmentation` algorithm. Furthermore, as shown in Munari et al. (2017), the intrinsic deviation of `PINOCCHIO` relative to full N-body simulations is typically in the range of $5 - 10\%$ for the HMF.

In this context, the overall underestimation of collapse times at the level of $\sim 0.16\%$, as illustrated in Figure 3.10 is well within acceptable tolerances and has no significant impact on the physical predictions. This confirms the numerical consistency of the GPU porting in terms of its impact on `PINOCCHIO` scientific final output. At the high-mass end, slightly larger fluctuations are observed, which are expected due to the inherently lower number statistics in that regime. These deviations do not indicate a systematic bias and are fully consistent with stochastic sampling noise.

The overall agreement for both number of collapsed particles and HMF reinforces the reliability of the GPU-native implementation not only at the interpolation level, but also in preserving the physical accuracy of the full pipeline when applied to large-scale cosmological workflows.

Chapter 4

Energy measurement and efficiency analysis

Building on the GPU porting effort described in Chapter 3, we evaluate the energy impact of GPU offloading by measuring and comparing the energy consumption of CPU- and GPU-based of collapse time calculation. For this purpose, we integrate the Power Measurement Toolkit (PMT), a high-level library designed to monitor power consumption on heterogeneous systems, enabling a quantitative assessment of the energy efficiency of the optimized code. We begin by introducing the key characteristics of a different NVIDIA-based system (KAROLINA) adopted for the energy-efficiency analysis, while the same AMD-based platform (SETONIX) used in Chapter 3 is retained. This choice is motivated by the fact that on the NVIDIA-LEONARDO system it was not possible to access the hardware-level counters required to measure energy consumption on both CPUs and GPUs. We then introduce PMT library and its new parallel implementation for MPI-based applications. We define the energy-profiling strategy and the metrics used to assess energy efficiency, and outline the resource configurations adopted for both strong- and weak-scaling (multi-node) tests on NVIDIA-KAROLINA and AMD-based platforms. Finally, we present the energy-efficiency results obtained from these measurements. The conclusions of this study are reported in Chapter 7.

An additional consideration is essential to properly frame the discussion and the results presented in this Chapter. All energy benchmarks are conducted using a full compute node as the basic computational unit, rather than a quarter node as used in the porting experiments. This choice is motivated by the fact that the primary interest lies in the energy consumed by the entire node. Using partial-node configurations would require non-trivial modeling to estimate the idle power of non-active components (CPUs or GPUs), which must still be accounted for in a reliable energy-efficiency analysis.

The content of this Chapter is based on the manuscript Lacopo et al. (2025.), submitted to Astronomy & Computing special issue "VSI: HPC in Cosmology and Astrophysics", accepted for publication.

4.1 Additional computing platform

Following the procedure adopted in Chapter 3, we perform the energy-efficiency analysis on two state-of-the-art supercomputing platforms representative of NVIDIA and AMD GPU architectures. In this Chapter, the NVIDIA-based system LEONARDO is replaced by KAROLINA, while the AMD-based platform SETONIX is retained. This choice reflects the different measurement requirements of the present study and enables a consistent evaluation of energy consumption across heterogeneous GPU architectures.

4.1.1 NVIDIA-KAROLINA cluster

KAROLINA is part of the IT4I [1] Czech national HPC facility. The platform consists of 72, 64-cores, 2 × AMD EPYC 7H12 CPU nodes equipped with 8 NVIDIA Tesla Ampere 100 GPUs. Each CPU node has 1 TB of DDR4 memory, while each GPU provides 40 GB of HBM2 memory. The CPU and the GPU are interfaced by a PCIe Gen4 interconnection. Computing nodes are connected through a Mellanox HDR Infiniband network with DragonFly+ topology, with 200 GB/s bidirectional bandwidth.

The `PINOCCHIO` code was compiled on this system using the NVC/NVC++ compilers v24.3 and OpenMPI library v5.1 was employed.

4.1.2 Additional architectural comparison and performance implications

The architectural considerations discussed in Section 3.3.3 remain fully applicable in the present analysis, as the NVIDIA GPUs deployed on KAROLINA are identical to those used on LEONARDO. Consequently, the performance implications associated with the NVIDIA architecture are unchanged.

It is only worth stressing that the FP64 peak performance differs by approximately *5x* between AMD GCDs (47.9 TFLOPS) and NVIDIA A100s (9.7 TFLOPS). Such a disparity is expected to influence the execution of compute-bound kernels, with its impact examined later in Section 4.4.

[1] https://www.it4i.cz/en/infrastructure/karolina

4.1.3 Additional CPU-GPU interconnection topology

KAROLINA architecture

The KAROLINA compute nodes feature a PCIe-based interconnect with a bipartite organization, described as follows:

```
CPU Socket 0 (64 cores) <-PCIe Gen4-> GPU 0, GPU 1, GPU 2, GPU 3
CPU Socket 1 (64 cores) <-PCIe Gen4-> GPU 4, GPU 5, GPU 6, GPU 7
```

- **Host-to-Device Interconnect**: Similar to LEONARDO, communication relies on PCIe Gen4 ×16 links, each providing approximately 31.5 GB/s bidirectional bandwidth, with a nominal latency between 1 and 2 µs.
- **Non-Uniform Memory Access (NUMA) Considerations**: GPUs are grouped under specific CPU sockets. As with LEONARDO, careful MPI rank placement is essential to reduce cross-socket communication and ensure optimal data locality.
- **Device-to-Device Interconnect**: The nodes are equipped with NVLink 3.0 links offering up to 600 GB/s inter-GPU bandwidth between paired devices. These links are not leveraged in the current embarrassingly parallel configuration.

4.1.4 Power efficiency considerations

While the performance implications for our workload follow the same considerations discussed in Section 3.3.5 and will be examined in detail in Section 4.4, additional power-efficiency considerations are required to properly interpret the energy behavior of the two platforms.

While the AMD MI250X has a higher Thermal Design Power (TDP) of 560W compared to the NVIDIA A100's 400W, the power efficiency story is more nuanced when considering FP64 performance per watt. The MI250X delivers approximately 85.5 GFLOPS/W (47.9 TFLOPS at 560W per card, considering both GCDs), while the A100 provides 24.3 GFLOPS/W (9.7 TFLOPS at 400W), resulting in a 3.5× advantage in raw FP64 efficiency for AMD. This theoretical advantage is validated by the two platforms positions in the Green500 ranking, with SETONIX placed 20th versus KAROLINA's 57th position.

For our compute-bound kernel achieving 8× speedup on AMD versus 2× on NVIDIA (compared to CPU baseline), we can theoretically expect the AMD platform to complete the same workload using approximately (1.4 × power) × (0.5 × time) = 0.7× the energy of the NVIDIA platform, representing a 30% energy saving despite the higher instantaneous

power draw. However, this efficiency advantage must be weighed against practical considerations including cooling infrastructure requirements and the higher power delivery demands of the MI250X.

4.2 Parallel PMT

Power Measurements Toolkit [2] (Corda et al. 2022) is a C++ library that allows energetic profiling of code portions in a plethora of architectures, such as Intel and AMD CPUs through Running Average Power Limit (RAPL) counters (David et al. 2010), NVIDIA GPUs through Nvidia Management Library (NVML) (Kasichayanula et al. 2012), and AMD GPUs through Radeon Open Compute platform (ROCM) [3] and AMD System Management Interface (SMI)[4]. For the sake of completeness, PMT also provides an opportunity to measure the energy for other architectures, like Xilinx Field-Programmable Gate Arrays (FPGAs).

PMT is compiled as a standard shared object (.so) that can be linked to any C/C++ code. To use it, the header must be included:

```
#include "pmt.h"
```

and compilation requires specifying the appropriate PMT *include* and *library* paths as follows:

```
CC/CXX -I$(PATH_TO_PMT)/include -L$(PATH_TO_PMT)/lib64
my_code.c/my_code.cpp -lpmt
```

PMT is perfectly suitable for serial codes, when the main process profiles the relevant *kernel* or code portion and outputs the resulting energy, runtime, and power. However, the situation is more complex for MPI codes, such as `PINOCCHIO`. Allowing every process to read hardware counters and output their own results would be redundant and inefficient. To address this limitation, we developed a parallel version of PMT that collects the results pertaining to all MPI processes and writes a final report for all CPUs/GPUs profiling.

The Parallel PMT is a C++ library with a C wrapper, making it usable within any C/C++ application. It relies on the standard PMT library and therefore cannot be compiled unless PMT is already available on the system. The current implementation supports

[2] https://git.astron.nl/RD/pmt.git
[3] https://www.amd.com/en/products/software/rocm.html
[4] https://rocm.docs.amd.com/projects/amdsmi/en/latest/

RAPL counters for CPUs and provides a flag to specify whether the GPUs (if present) are NVIDIA or AMD.

Once the library is compiled, the file:

```
libpmt_parallel.so
```

is created, and the header:

```
#include "energy_pmt.h"
```

must be included the application.

When the application starts, PMT is initialized through the function call:

```
PMT_CREATE(MPI Comm comm, int rapl, int *devID, int numGPUs);
```

where `comm` is the MPI communicator of the original application, `rapl` is an integer that can be either 0 (`rapl` reading not active) or 1 (`rapl` reading active), `devID` is a pointer to the array of devices (per task) and `numGPUs` is the number of GPUs associated to each MPI task. In this work, we set `numGPUs` equal to 1, so that each MPI task is associated with a single GPU. In a future release of the library, we plan to extend the support to non trivial cases to where more than one GPU is exclusively assigned to a taks.

The `PMT_CREATE` function initializes the PMT counters for the specific hardware and the MPI communicators required by the library. Indeed, the library is MPI- and NUMA-aware: it automatically detects the number of computing nodes and to which node the specific task belongs.

For multi-socket nodes, the library was extended to create communicators at the socket level, collecting all the MPI tasks running on the same socket. In addition to the single sockets and single nodes communicator, each socket and each node elects a master process (the rank 0 in the specific communicator). Higher-level communicators are then created to gather these socket and node masters. This structure enables meaningful statistical analyses, for example when running an application on many nodes and the goal is of computing averages and standard deviations across them. In the case of multi-socket nodes, statistics are available at both node and socket levels.

In Sections 3.3 and 4.1, we discussed the possibility of having either single GPUs devices, such as the NVIDIA Tesla A100 (LEONARDO and KAROLINA), or multi-GCD GPUs, such as the AMD Radeon Instinct MI250X (SETONIX). In the former case, the setup is straightforward: one MPI task is assigned to each GPU, and the task is responsible for accessing the hardware counters and exchanging information with all the

other processes. The latter case is more complex, since the original application assigns one MPI task per GCD, but only one task per physical GPU should access the hardware counters. On AMD GPUs, this translates into two MPI ranks per GPU (one per GCD), with only the first rank reading the GPU counters and sharing the results with the other ranks. Consequently, in this specific case, only even MPI ranks initialize the PMT, while a dedicated communicator is initialized in the original code and passed to the library as `comm` argument.

Once PMT is initialized, the code segment or the `kernel` to be profiled is selected and bracketed as shown in the following pseudo-code:

```
PMT_CPU_START("Kernel_name");
PMT_GPU_START("Kernel_name", int devID);

kernel execution

PMT_CPU_STOP("Kernel_name");
PMT_GPU_STOP("Kernel_name", int devID);
```

where `devID` is the device number assigned to the specific MPI task. The tag must be the same for the `PMT_*_START` and `PMT_*_STOP` functions, otherwise PMT will return an error.

At the end of the run the global master process (the master of all PMT communicators), generates the results using the following functions:

```
PMT_CPU_SHOW("Kernel_name");
PMT_GPU_SHOW("Kernel_name", int devID);
```

Once this is done, the library automatically writes two output files per tag, one for CPUs and one for GPUs, reporting the total energy consumed by the entire *kernel* as well as the statistics at the socket/node level. For GPUs, additional statistics are provided at the individual GPU level. Finally, the following function must be called to free the memory allocated from the MPI communicators:

```
PMT_FREE();
```

For completeness, here is an example of how to compile and link a code with the parallel PMT:

```
CC/CXX -I$(PATH_TO_PARALLEL_PMT) -L$(PATH_TO_PARALLEL_PMT)
my_code.c/my_code.cpp -lpmt_parallel
```

This parallel version of the PMT library is expected to be released in the near future. Unless otherwise specified, all energy measurements presented in this work were obtained using this specific version.

4.3 Energy profiling strategy and resource configuration

From energy-to-solution and time-to-solution considered separately, it is possible to infer which configuration is the greenest and which is the fastest. However, it is more infor-

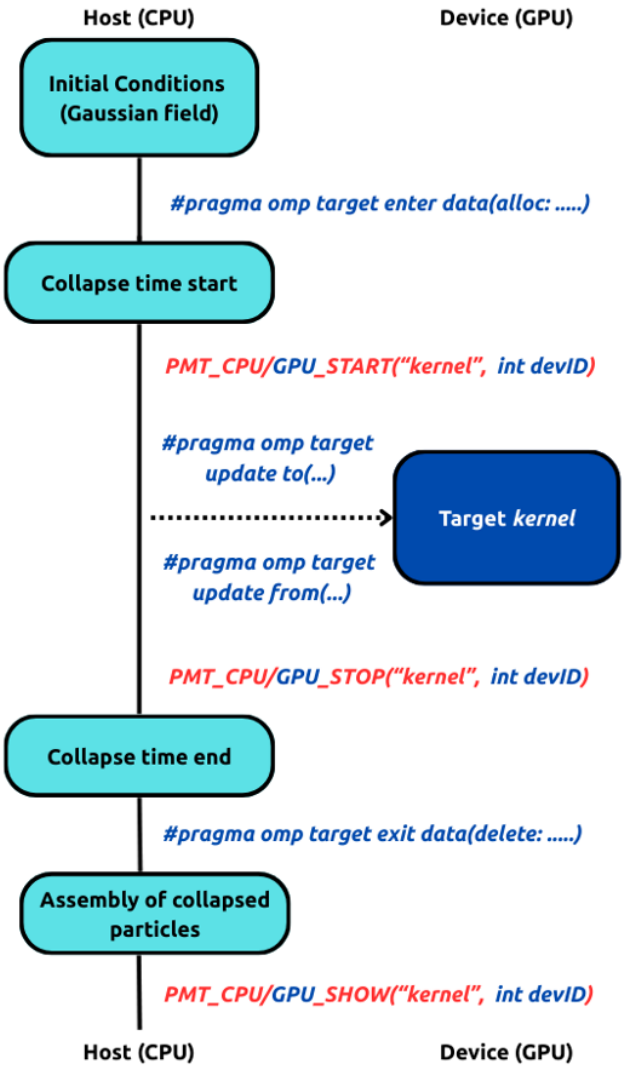


Figure 4.1: Same as Figure 3.1, but highlighting the PMT function calls used for energy profiling. The data management and execution flow on both the host and the device are reported for completeness.

mative to combine these into a single figure of merit that identifies the most efficient configuration, where efficiency refers to the simultaneous minimization of runtime and energy consumption.

For this purpose, we adopt two complementary metrics: the Energy–Delay Product (EDP, Goz et al. 2020; Taffoni et al. 2020b) and the Green Productivity (GP, Lacopo et al. 2025a; De Rubeis et al. 2025). Throughout this analysis the term *configuration* will be assigned to a specific CPU or GPU run with N CUs. After introducing these metrics, we detail the resource configurations used in our benchmarks. A schematic overview of the energy-profiling workflow employed to evaluate EDP and GP, including the PMT function calls discussed in Section 4.2, is shown in Figure 4.1.

4.3.1 Energy-Delay Product

The EDP is defined as the product of the energy consumed by a specific configuration and the corresponding time-to-solution, as defined below:

$$EDP = E \times T^w \tag{4.1}$$

where w is a weight factor which determines the importance given to time-to-solution compared to energy-to-solution. In this work we present our results with $w = 1, 2, 3$. The configuration which minimizes the EDP is the most efficient one.

4.3.2 Green Productivity

By examining the EDP for different configurations, it is possible to identify which one has the lowest EDP. However, it is also useful to define a metric that relates a given configuration to a reference one, in order to asses which one is the most efficient. This motivates the introduction of the GP, defined as follows:

$$GP = \frac{T_0/T_N}{\alpha E_N/E_0} \tag{4.2}$$

where T_0 and E_0 are time-to-solution and energy-to-solution of the reference configuration, respectively, and T_N and E_N are the corresponding quantities for the tested configuration. The parameter α specifies the relative weight assigned to energy-to-solution compared to time-to-solution. From this definition, it follows that when the same code implementation is tested by varying only the number of CUs, the numerator reduces to the runtime speedup, while the denominator corresponds to the inverse of energy speedup. For GP analyses, we assign equal importance the two quantities by setting $\alpha = 1$. In this way, the configuration

that maximizes GP is the most efficient one.

4.3.3 Energy benchmarks: multi-node

To ensure a meaningful comparison across the different computing platforms, we perform the benchmarks by choosing as a CU the full node, rather than $\frac{1}{4}$ of a node, as employed in the porting benchmarks (Section 3.4.3). This choice is motivated by the fact that, for most applications, the primary interest lies in the energy which is consumed by the entire node. Otherwise, non-trivial models would be required to estimate the idle consumption of non-active components (CPUs or GPUs), which must still be accounted for. In pure CPU runs, the idle GPU consumption is included automatically since each MPI task activates PMT GPU profiling to account for the imprint of accelerators. Finally, since the parallel PMT library relies on MPI, it is also relevant to test its behavior across multiple computing nodes.

For our specific purpose, we adopted the following configurations. These choices is motivated by the heterogeneity of the target architectures, as detailed below:

- On KAROLINA, each MPI process can manage one GPU out of eight using at most sixteen cores (i.e. spawning sixteen OMP threads). The CU will be the node, with 8 MPI tasks, 16 OpenMP threads and one GPU per task;
- On SETONIX, each chiplet, to which MPI processes are binded, consists of eight cores and is associated with one GPU. The CU will again be the node, with 8 MPI tasks, 8 OpenMP threads and one GPU per task.

As discussed in Section 4.2, the PMT library does not account for the energy consumed by inter-node communication. This omission does not impact our treatment since the `kernel` computation is embarrassingly parallel and no MPI communication is needed during its execution. In all runs, the `kernel` time-to-solution is the wall-clock time required by the actual computation, as measured by the slowest MPI task. The energy-to-solution is always the one read by the highest energy demanding MPI task. For GPU runs, both time-to-solution and energy-to-solution include the cost of pure GPU computation as well as host–device data transfers.

Strong scaling tests keep the number of PINOCCHIO particles fixed while increasing the CUs at each step. In these tests we use up to 16 CUs. For both CPU and GPU runs, we set the PINOCCHIO number of particles fixed to 768^3. Configurations for the SETONIX cluster are summarized in Table 4.1. Since the number of CPU cores differs from SETONIX to KAROLINA, which is constituted by dual-socket nodes, we also summarize the configuration for the latter cluster in Table 4.3.

	Nodes	MPI(threads)	GPUs	Grid size
CPU	1	8 (8)	0	768^3
CPU	2	16 (8)	0	768^3
CPU	4	32 (8)	0	768^3
CPU	6	48 (8)	0	768^3
CPU	8	64 (8)	0	768^3
CPU	12	96 (8)	0	768^3
CPU	16	128 (8)	0	768^3
GPU	1	8 (8)	8	768^3
GPU	2	16 (8)	16	768^3
GPU	4	32 (8)	32	768^3
GPU	6	48 (8)	48	768^3
GPU	8	64 (8)	64	768^3
GPU	12	96 (8)	96	768^3
GPU	16	128 (8)	128	768^3

Table 4.1: SETONIX configurations for CPU only and GPU runs adopted for *strong* scaling tests. The first column indicates the number of nodes. The second column reports to the total number of MPI tasks and the number of threads spawned by each task. The third column specifies the number of GPUs, while the last column gives the fixed box size adopted in the simulations. For GPU runs, the number of MPI tasks corresponds to the number of GPUs.

	Nodes	MPI(threads)	GPUs	Grid size
CPU	1	8 (8)	0	512^3
CPU	2	16 (8)	0	644^3
CPU	4	32 (8)	0	812^3
CPU	8	64 (8)	0	1024^3
GPU	1	8 (8)	8	512^3
GPU	2	16 (8)	16	644^3
GPU	4	32 (8)	32	812^3
GPU	8	64 (8)	64	1024^3

Table 4.2: SETONIX configurations for CPU only and GPU runs adopted for *weak* scaling tests. The column structure is the same as in Table 4.1.

Weak scaling tests, double both the total number of particles and `CUs` at each step, ranging from 512^3 up to 1024^3. The tests started from the 1 `CU` configuration and scaled up to 8 `CUs`. All configurations, together with the corresponding particle counts, are summarized in Table 4.2 and Table 4.4, for the SETONIX and KAROLINA clusters, respectively.

	Nodes	MPI(threads)	GPUs	Grid size
CPU	1	8 (16)	0	768^3
CPU	2	16 (16)	0	768^3
CPU	4	32 (16)	0	768^3
CPU	6	48 (16)	0	768^3
CPU	8	64 (16)	0	768^3
CPU	12	96 (16)	0	768^3
CPU	16	128 (16)	0	768^3
GPU	1	8 (16)	8	768^3
GPU	2	16 (16)	16	768^3
GPU	4	32 (16)	32	768^3
GPU	6	48 (16)	48	768^3
GPU	8	64 (16)	64	768^3
GPU	12	96 (16)	96	768^3
GPU	16	128 (16)	128	768^3

Table 4.3: Same as Table 4.1 but for KAROLINA.

	Nodes	MPI(threads)	GPUs	Grid size
CPU	1	8 (16)	0	512^3
CPU	2	16 (16)	0	644^3
CPU	4	32 (16)	0	812^3
CPU	8	64 (16)	0	1024^3
GPU	1	8 (16)	8	512^3
GPU	2	16 (16)	16	644^3
GPU	4	32 (16)	32	812^3
GPU	8	64 (16)	64	1024^3

Table 4.4: Same as Table 4.2 but for KAROLINA.

4.4 Energy measurements results

In this Section we present the results of strong- and weak-scaling tests, with a focus on the scalability of energy-to-solution. Since in many cases the combination of energy-to-solution and time-to-solution is more informative, we also report the `EDP` (Section 4.3.1) and `GP` (Section 4.3.2) results. Results obtained on the SETONIX cluster are presented in Section 4.4.1, while those for the KAROLINA cluster are given in Section 4.4.2.

4.4.1 Setonix cluster

Strong scaling tests

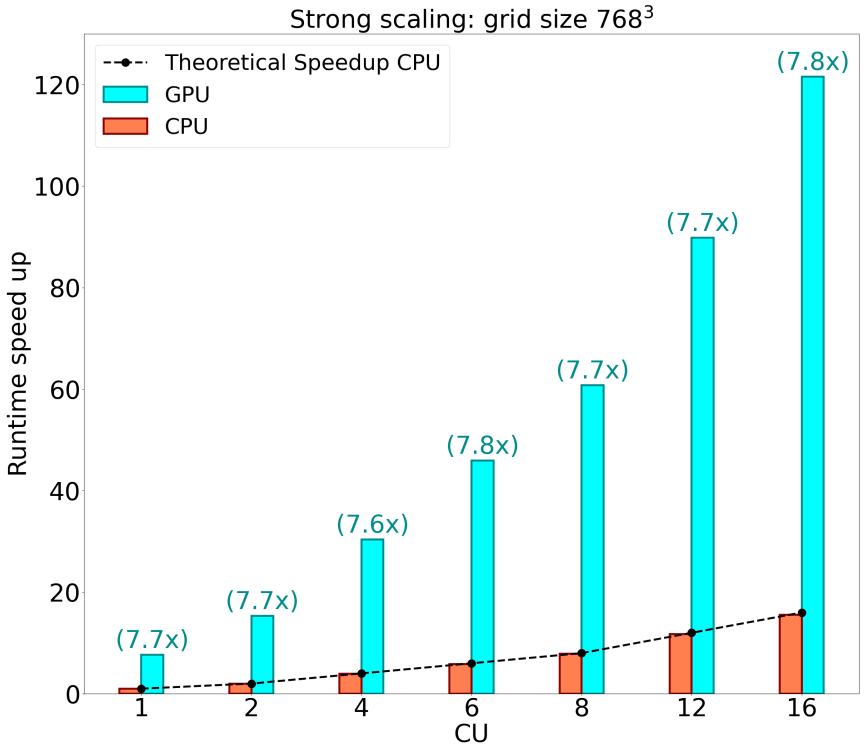


Figure 4.2: Strong scaling runtime speedup for the SETONIX cluster. All the histograms are normalized to the 1 CPU `CU` reference. The numbers above the GPU bar indicate the gain factor compared to the same `CU` CPU configuration.

The runtime strong scaling results for the SETONIX cluster are shown in Figure 4.2. All histograms are normalized to the 1 CPU `CU` configuration, so their height represents the relative increase with respect to the reference. The numbers above each GPU bar indicate the speedup of GPUs compared to CPUs with exactly the same number of `CUs`. Across all runs, GPU runtime gain remains stable around $7.7 - 7.8\times$ times over the corresponding CPU configuration, consistent with those reported in Section 3.5.

Figure 4.3 shows the energy speedup results on the SETONIX cluster, where all histograms are normalized to the reference configuration (1 CPU `CU`). The speedup is nearly flat because the problem (box) size is fixed while the computing resources increase at each step. For an embarrassingly parallel code, doubling the `CUs` reduces the runtime, as shown in Figure 4.2, but the energy remains constant, since the decrease in runtime is balanced by the increase in allocated resources. Again, the numbers above the GPU

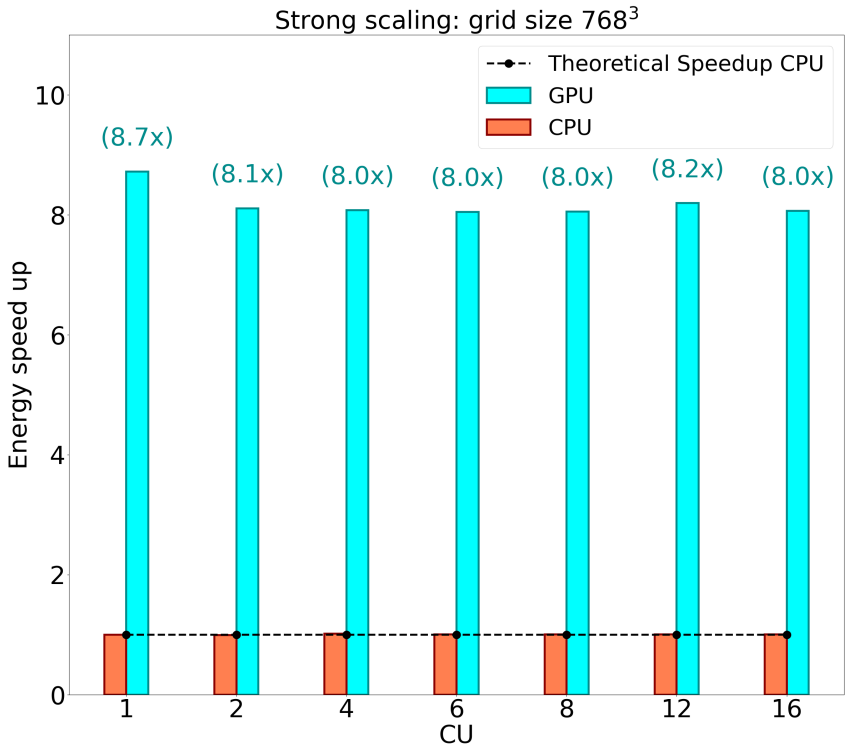


Figure 4.3: Strong scaling energy speedup for the SETONIX cluster. All histograms are normalized and annotated as in Figure 4.2.

histograms indicates the speedup relative to the corresponding CPU configuration. Overall, the gain in energy-to-solution from using GPUs exceeds 8×.

Figure 4.4 shows the EDP results in strong scaling tests on the SETONIX cluster, with weight factors $w = 1, 2, 3$. These tests were performed for all configurations reported in Table 4.1. Cyan stars denote pure CPU runs, while dark red Greek crosses correspond to GPU runs. The y-axis is in logarithmic scale. For a fixed number CUs, even when $w = 1$ GPUs are more efficient than their CPU counterpart by almost a factor of $\sim 64\times$. This is consistent with Figures 4.2 and 4.3, which show that GPUs are both faster and greener by a factor of $\sim 8\times$. EDP plot shows the dramatic advantage of GPUs when combining energy-to-solution and time-to-solution. When a greater emphasis is placed on runtime gains (i.e. $w = 2, 3$), the gap between CPU and GPU runs becomes increasingly pronounced. The EDP decreases as a function of CUs, indicating that employing more resources in these strong-scaling tests improves system utilization: energy consumption remains constant, while runtime halves at each step.

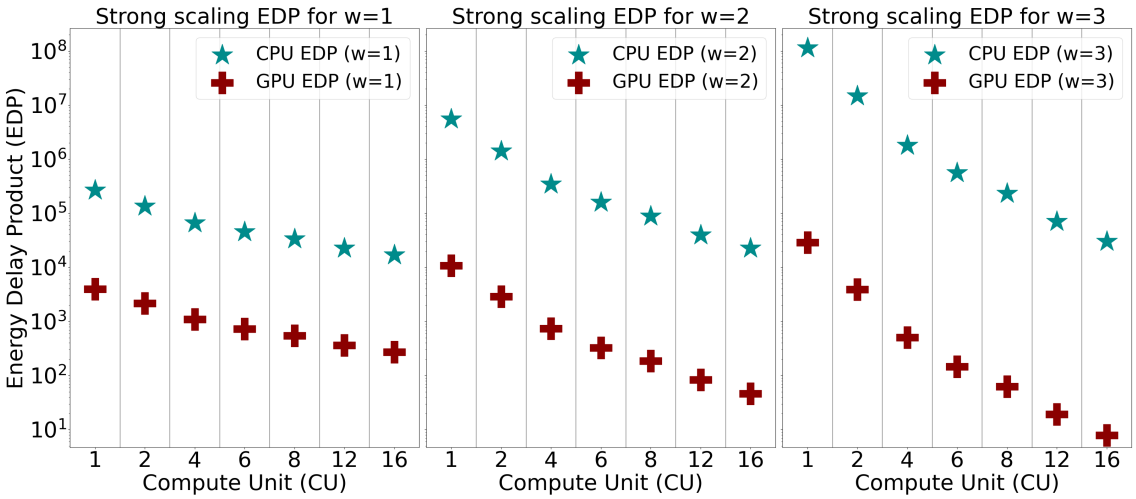


Figure 4.4: EDP for the SETONIX cluster in strong scaling tests, for all CPU and GPU configurations. Results are shown for $w = 1, 2, 3$.

Weak scaling tests

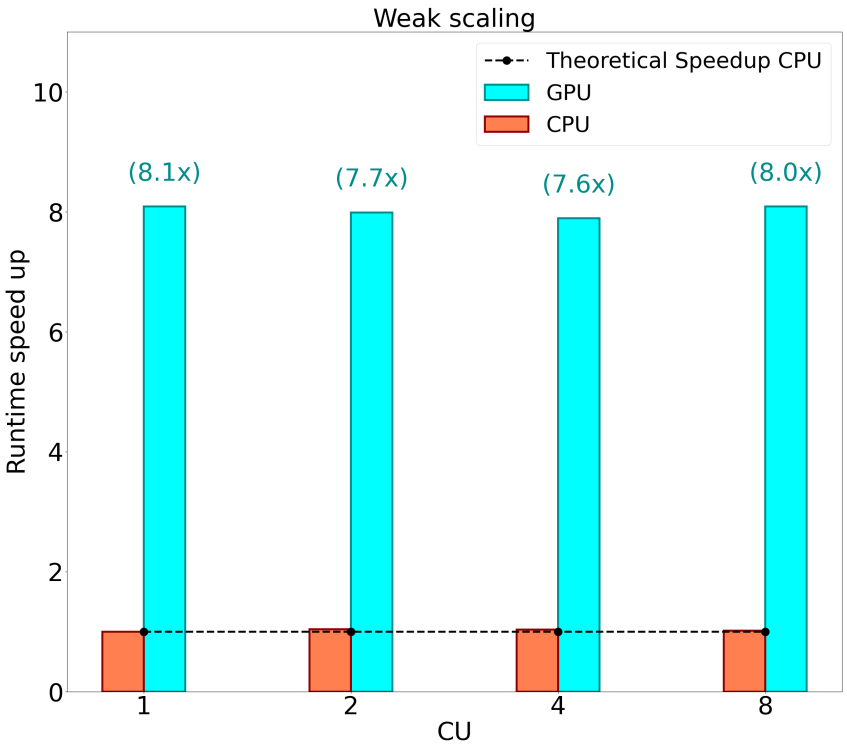


Figure 4.5: Same as Figure 4.2, but for the weak scaling.

Figures 4.5 and 4.6 show the weak-scaling results for runtime and energy consumption on the SETONIX cluster. All histograms are normalized and annotated as in Figure 4.2.

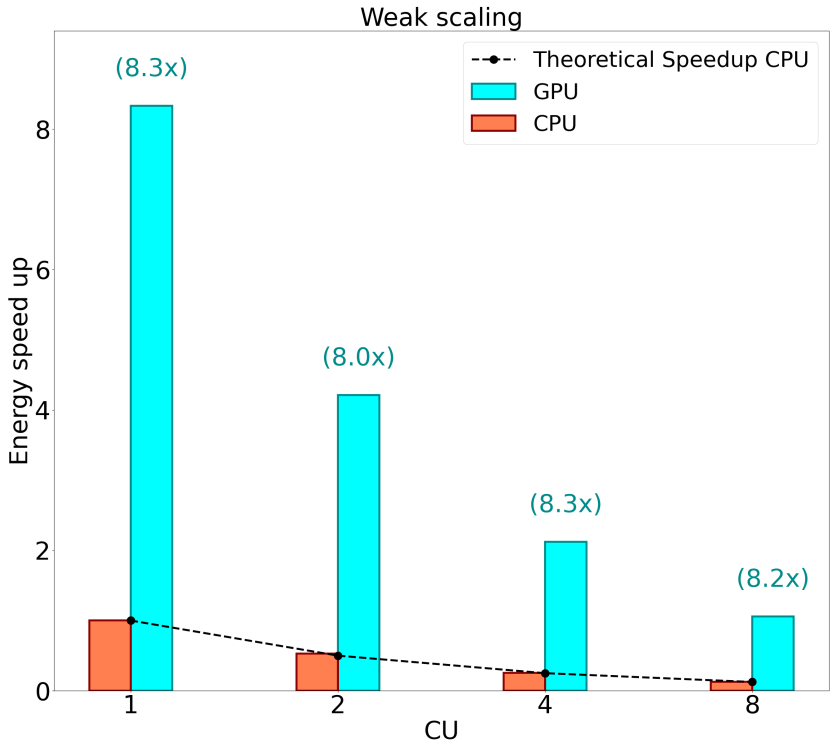


Figure 4.6: Same as Figure 4.3, but for the weak scaling.

Runtime scaling is essentially flat, as expected for embarrassingly parallel codes where both the problem size and the number of `CUs` increase together, with GPUs performing $\sim 8\times$ faster than CPUs, consistent with the strong-scaling results. Energy consumption, doubles at each step because the workload per `CU` is fixed while runtime remains flat: running the code for the same time while doubling resources implies a theoretical doubling of energy usage.

In Figure 4.7 we show the results as in Figure 4.4, but for weak scaling. Unlike the trend we observed in Figure 4.4, the `EDP` increases as a function of `CUs` in this case. This is because the runtime remains constant while energy doubles at each step, leading to an overall increase of the `EDP` whenever the number of `CUs` is doubled. For $w = 1$, GPUs are more efficient by a factor of $\sim 64\times$, consistent with the strong scaling results. For $w = 2, 3$, which correspond to cases where more weight is given to time-to-solution than energy-to-solution, the gap between GPUs and CPUs becomes increasingly pronounced.

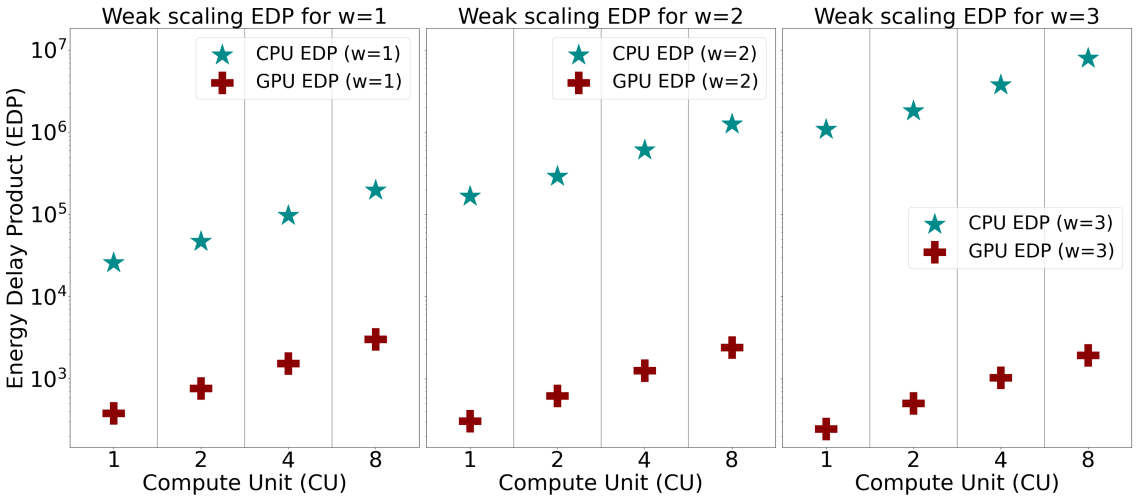


Figure 4.7: Same as Figure 4.4, but for weak scaling.

Green Productivity

GP provides insights into how the increasing resources affects overall system utilization. Here, we focus on the strong scaling tests discussed in Section 4.3.3 and summarized in Table 4.1, adopting $\alpha = 1$ to give equal weight to energy-to-solution and time-to-solution.

Results for the SETONIX system are shown in Figure 4.8, where cyan stars denotes CPU runs and dark red Greek crosses correspond to GPU runs. CPU and GPU runs are self-normalized: for CPU runs the reference configuration is the 1 CPU CU, while for GPU runs the reference is 1 GPU CU. Since the code is embarrassingly parallel, increasing the number CUs should in principle lead to higher GP. However, this does not generally hold

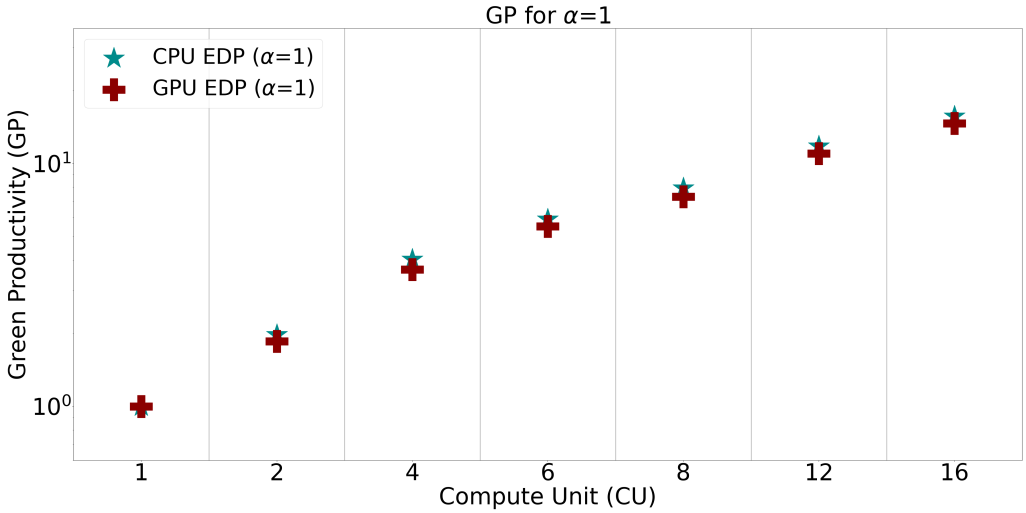


Figure 4.8: GP for the SETONIX cluster strong scaling tests, for all CPU and GPU configurations. For CPU runs, the reference is 1 CPU CU, while for GPU ones the reference is 1 GPU CU. Results are shown for $\alpha = 1$.

for non-compute bound algorithms, where the configuration that maximizes GP is often the one with the lowest CUs fitting the problem, as presented in Lacopo et al. (2025a). Nevertheless, the results in Figure 4.8 shows that GP does not increase linearly, with a shallow “knee" appearing around 6 CUs. This behavior indicates that the problem size becomes too small to fully utilize the available computing resources, especially for GPU runs.

4.4.2 Karolina cluster

Strong scaling tests

Similar to Figure 4.2, Figure 4.9 presents the runtime strong-scaling results for KAROLINA. In this case, the runtime gain from GPU runs is about 4× smaller than that observed on SETONIX. This is due both to higher performance of the dual-socket CPUs on KAROLINA (essentially twice the computing power of SETONIX CPUs) and to the difference in FP64 computing power between AMD and NVIDIA GPUs, with the latter offering lower peak performance (see Section 3.3).

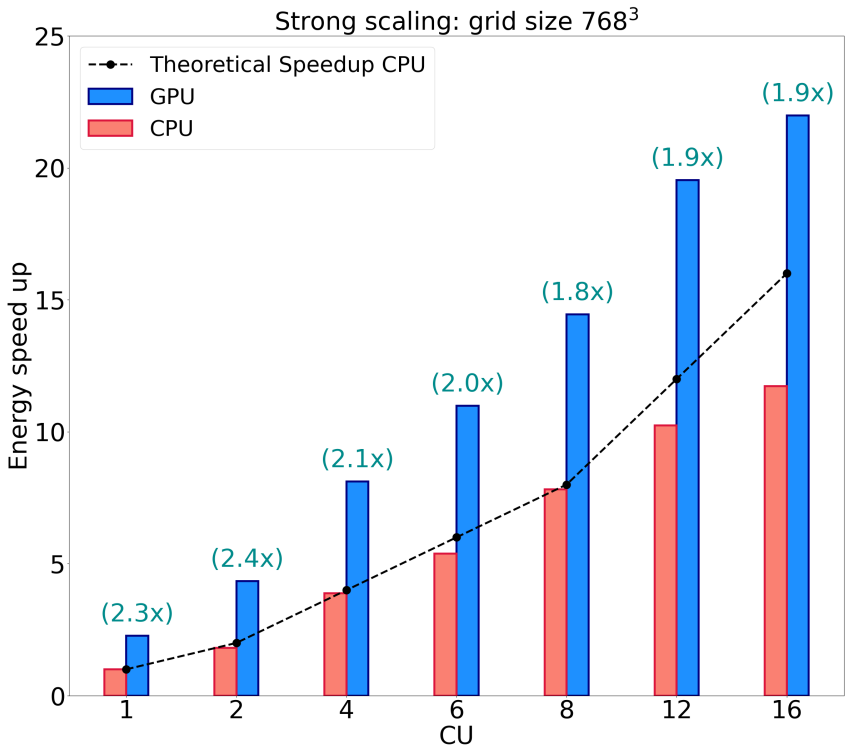


Figure 4.9: Same as Figure 4.2, but for the KAROLINA cluster.

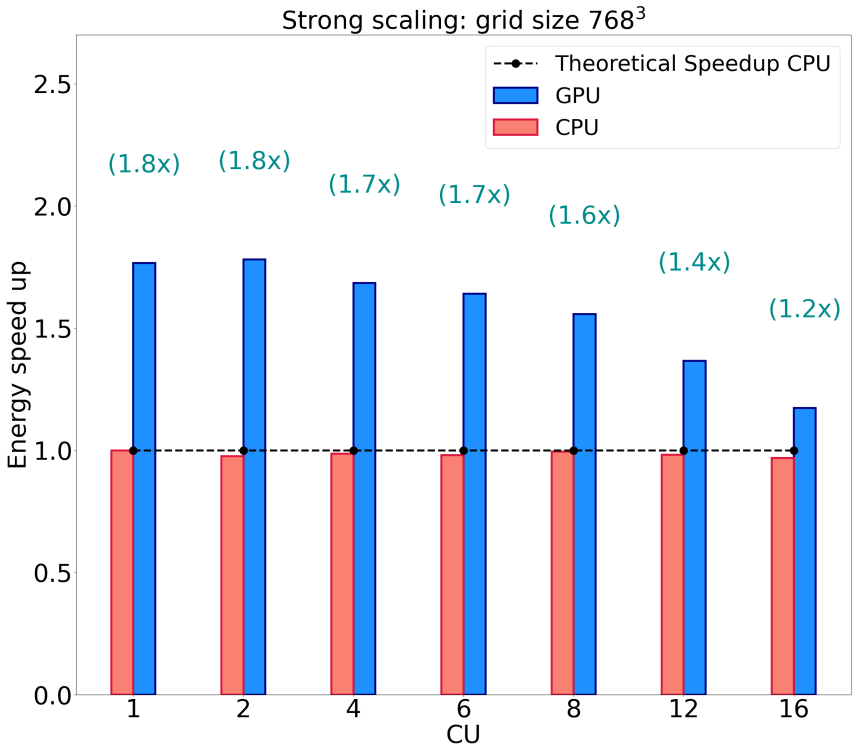


Figure 4.10: Same as Figure 4.3, but for the KAROLINA cluster.

As a result, the runtime gain for GPU configurations is limited to about 2×, compared to the homologous CPU configurations. Interestingly, KAROLINA exhibits worse strong scaling behavior than SETONIX for both CPU and GPU runs, with a significant deviation from the ideal scaling. However, GPU configurations show an even worse speedup than CPU configurations, as is particularly evident in the energy speedup result shown in Figure 4.10. For runs with 12 or 16 `CUs`, the energy gain factor approaches a 1× factor, indicating that the GPU utilization becomes inefficient in these cases. This behavior can be explained follows: unlike the AMD+AMD hardware on SETONIX, the AMD+NVIDIA combination on KAROLINA lacks a fast InfinityFabric CPU-GPU connection, as discussed in Section 3.3. Ideal strong scaling is achieved only as long as hardware is fully utilized. Beyond 12 `CUs`, this condition is no longer met for GPU tests. As a result, GPUs remain underutilized while the runtime is dominated by CPU-GPU latency. The faster interconnection available on SETONIX partially mitigates for this under-utilization effect.

`EDP` results for KAROLINA are shown in Figure 4.11. Red stars refer to pure CPU runs, blue Greek crosses refer to GPU ones. In Figure 4.4 we observed a 64× advantage of GPUs over CPUs runs in the combined energy-runtime efficiency for $w = 1$. ON KAROLINA, this gap shrinks to 4× and decreases further for larger numbers of `CUs`. As expected, the

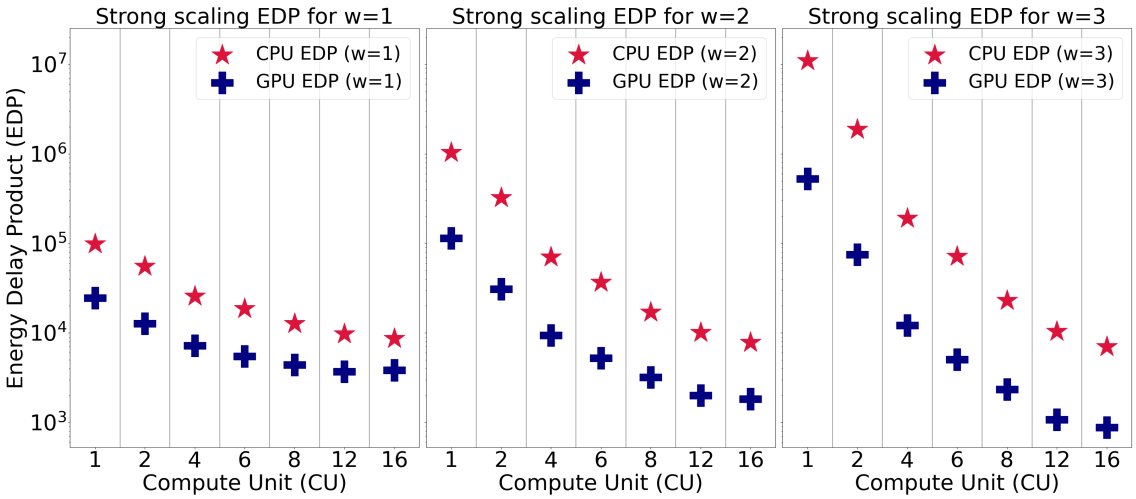


Figure 4.11: Same as Figure 4.4, but for the KAROLINA cluster.

gap increases when $w = 2, 3$, but it remains less pronounced than in Figure 4.4.

Weak scaling tests

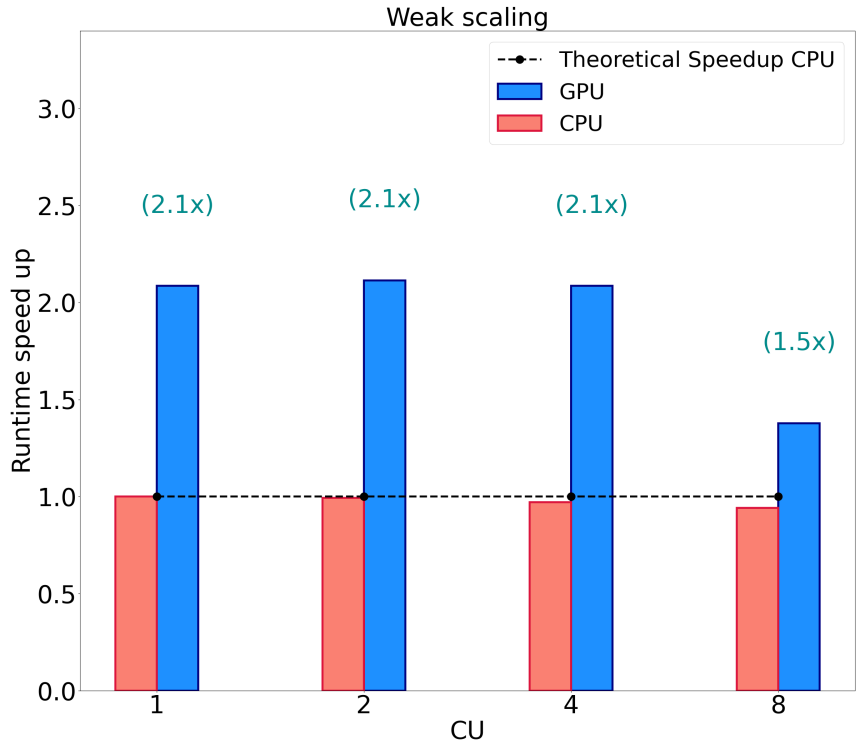


Figure 4.12: Same as Figure 4.5, but for the KAROLINA cluster.

Similar to Figure 4.5, Figure 4.12 presents the runtime weak-scaling results for KAROLINA.

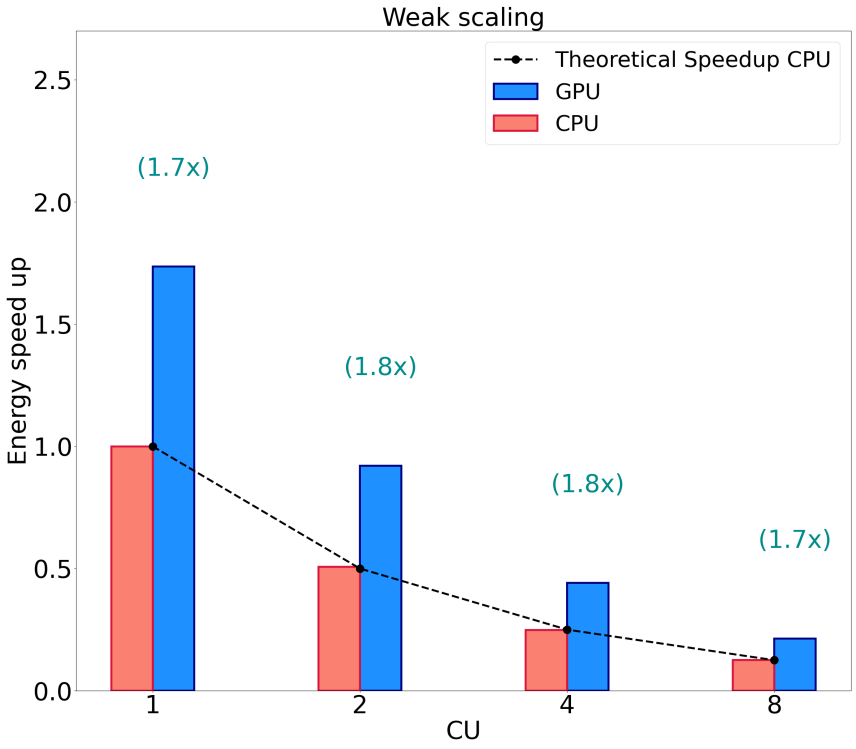


Figure 4.13: Same as Figure 4.6, but for the KAROLINA cluster.

While CPU runtimes scale almost as expected, with only a slight deviation for 8 `CUs`, the behavior with GPUs is more interesting. Specifically, when 8 `CUs` are used, the runtime gain factor drops from 2× to 1.5×, following a trend similar to the strong scaling tests. This represents a significant deviation from theoretical scaling, indicating that GPUs become less efficient in terms of runtime when many computing nodes are used.

Similar to Figure 4.6, Figure 4.13 presents the energy weak-scaling results for KAROLINA. Interestingly, the energy gain factors of each GPU configurations relative to the corresponding CPU configurations are stable, without the significant drop observed in Figure 4.12. This suggests that energy is not simply an integral over time. On some system, faster runtimes do not necessarily mean greener execution, and vice-versa. Overall, weak scaling tends to behave as expected in both CPU and GPU runs.

Weak scaling `EDP` results for KAROLINA are shown in Figure 4.14. The drop in GPU efficiency from 64× to 4× is consistent with strong scaling results in Figure 4.11 for $w = 1$, with a drop at 8 `CUs`. As expected, the gap between GPUs and CPUs widens for $w = 2, 3$.

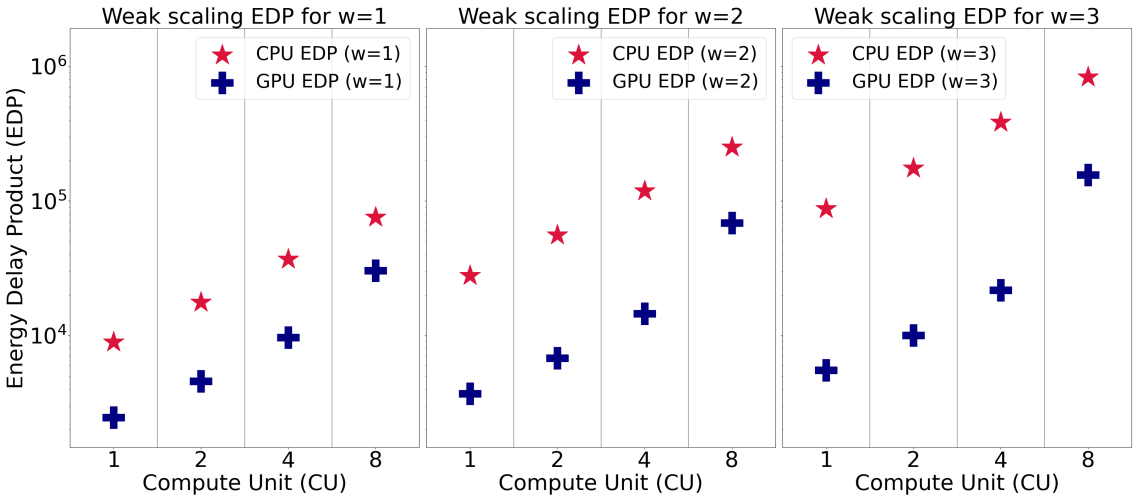


Figure 4.14: Same as Figure 4.7, but for the KAROLINA cluster.

Green Productivity

GP results for the KAROLINA cluster, shown in Figure 4.15, were obtained following the same approach described in Section 4.4.1, but using the configurations listed in Table 4.3. Unlike in Figure 4.8, where GP steadily increasing with CUs, on KAROLINA the non-ideal scaling results in a plateau as the number of computing resources increases. This effect is even more pronounced in GPU runs, where a drop is observed for runs with more than than 12 CUs.

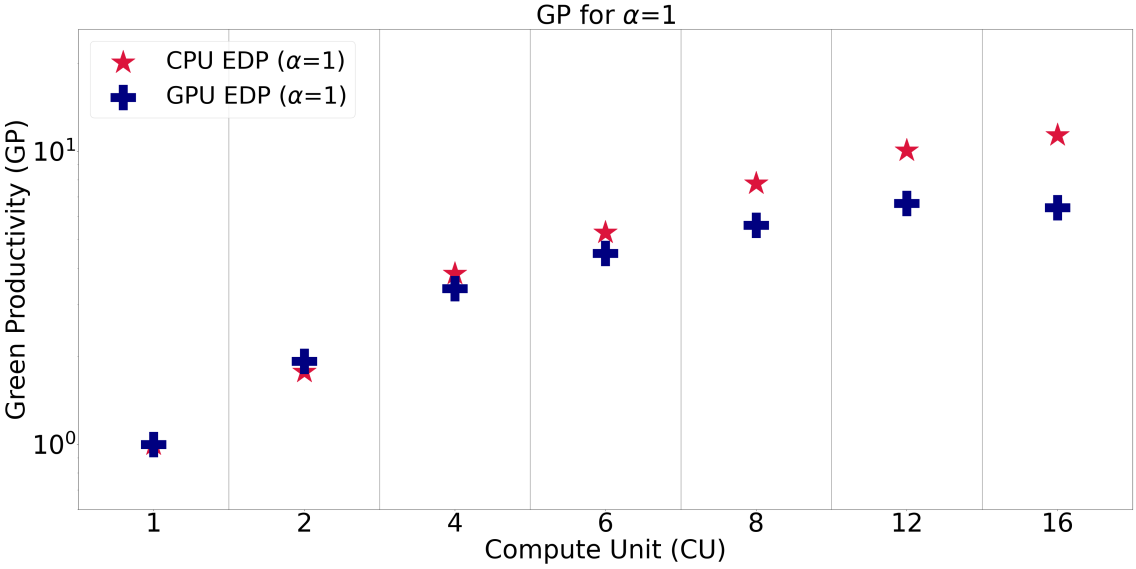


Figure 4.15: Same as Figure 4.8, but for the KAROLINA cluster.

4.5 KAROLINA vs SETONIX EDP

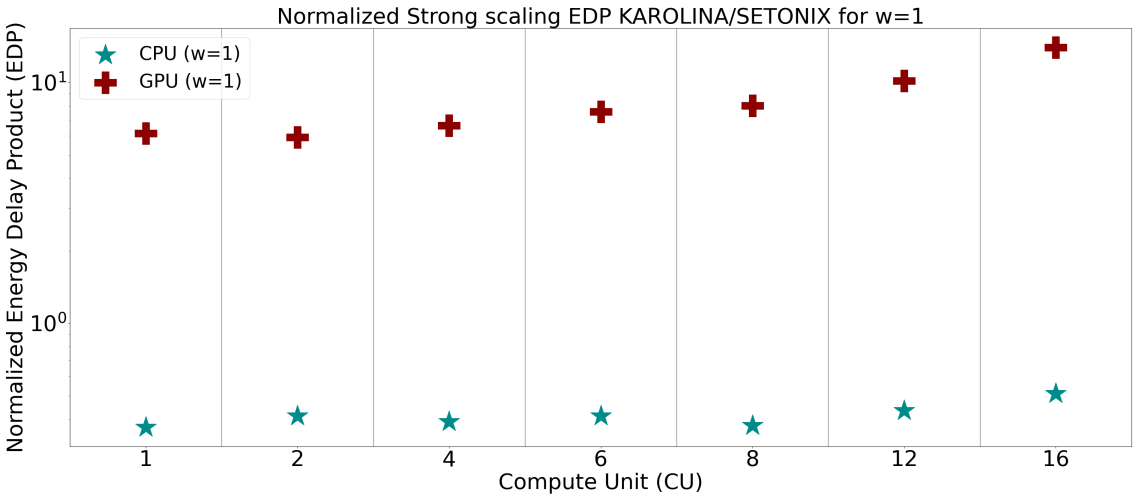


Figure 4.16: Normalized strong scaling EDP of the two clusters, obtained with the ratio between KAROLINA EDP and SETONIX EDP, for CPU and GPU configurations.

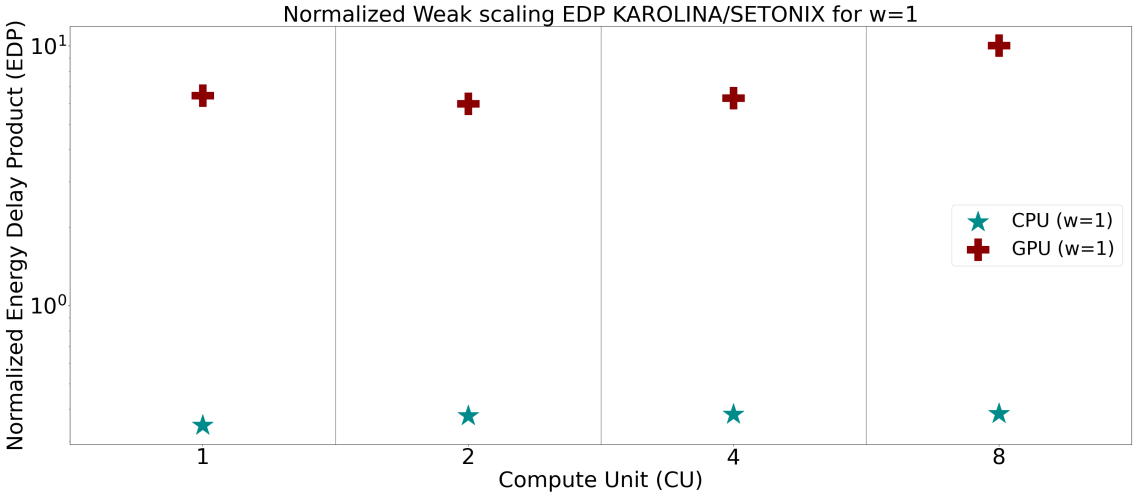


Figure 4.17: Same as Figure 4.16, but for the normalized weak scaling EDP.

To infer which out of the two architectures is best suited for the *kernel* under investigation, it is useful to analyzed the normalized EDP, i.e. the ratio between KAROLINA and SETONIX EDPs.

Figure 4.16 shows the normalized EDP, with $w = 1$, for strong scaling tests. The CPU EDP ratio is lower than one, meaning that KAROLINA is more efficient for CPU-only runs. This is expected, since KAROLINA has twice CPU cores than SETONIX, whereas for the GPU configurations SETONIX is more efficient by an order of magnitude. This is mainly a result of a better FP64 performance available in AMD GPUs compared to NVIDIA.

For the sake of completeness, we show the normalized weak scaling EDP, with $w = 1$, in Figure 4.17. Similar to the strong scaling case, KAROLINA best fits for pure CPU applications, while SETONIX is much better for GPU ones. However, we stress out that this specific analysis is not portable, and such plots should be provided for each couple of HPC platforms in which each user is supposed to run a code.

Chapter 5

Ongoing developments in PINOCCHIO

This Chapter presents a set of ongoing technical developments aimed at further extending the performance, scalability, and flexibility of PINOCCHIO on modern HPC architectures. Unlike the production-ready GPU porting and the validated energy-efficiency analysis discussed in the previous Chapters, the developments described here represent work that is currently being finalized and consolidated. In particular, the heFFTe-based FFT offloading is already functionally integrated with the GPU-accelerated collapse–time pipeline, while minor issues are still being addressed to enable its robust use within the full production workflow. In contrast, the new `Fragmentation` strategy described in this Chapter is still at an earlier stage.

Results from the heFFTe-based FFT offloading are presented in the first part of this Chapter, while preliminary tests of a new `Fragmentation` strategy employing a clustering algorithm are presented in the second part. These developments are intended to explore new directions for future large-scale cosmological production workflows and to address current scalability limitations and performance bottlenecks.

5.1 Preliminary results of the FFT porting

In addition to the collapse–time *kernel*, the FFT computations required for evaluating both the displacement source term and the Hessian of the gravitational potential (Section 2.4.2) were also ported to GPUs using the Highly Efficient FFT for Exascale (heFFTe) [1] library. heFFTe (Tomov et al. 2019) is a high-performance, GPU-optimized FFT framework designed for modern heterogeneous supercomputers, providing backend support for NVIDIA and AMD accelerators while ensuring excellent scalability on distributed-memory systems.

[1] https://icl.utk.edu/fft/

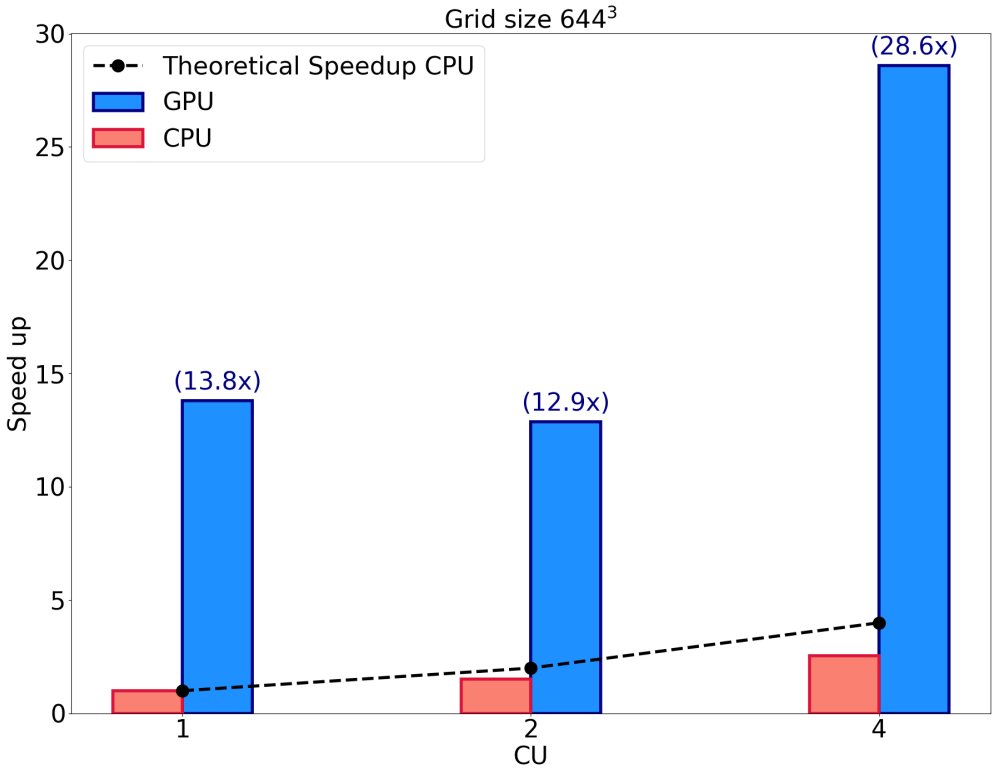


Figure 5.1: Same as Figure 3.6 but for the FFT offloading.

The FFT offloading was first implemented and tested within a dedicated PINOCCHIO mini-application developed to assess GPU performance up to the collapse-time stage. To verify consistency with the CPU implementation based on pFFT, we compared the distribution of collapsed particles, following the same methodology adopted for the collapse-time validation (Section 3.5.5). The GPU and CPU results were found to be binary identical, confirming the correctness of the heFFTe-based GPU porting. To evaluate the performance gain, we applied the same benchmarking strategy used for the collapse–time *kernel* (Section 3.4.3). The heFFTe-based FFT implementation has subsequently been integrated into the full PINOCCHIO code, which also includes the GPU-accelerated collapse time calculation and the CPU-based `Fragmentation`. While the integrated code is fully functional, discrepancies are currently observed in the total number of halos produced. Ongoing work is focused on identifying and mitigating the origin of these differences in order to achieve full consistency with the reference implementation.

The preliminary results, shown in Figure 5.1, indicate a speed-up of approximately 28× on the NVIDIA–LEONARDO system. Tests on the AMD platform are planned and will be performed once the current integration issues in the full PINOCCHIO pipeline are resolved. Notably, the configuration using two CU exhibits a slightly poorer speed-up than the single-CU case. This degradation is caused by an MPI communication issue

in heFFTe, which has already been reported to the library developers. Aside from this pathological configuration, the heFFTe backend consistently outperforms pFFT, which, for completeness, also shows imperfect scaling on the CPU.

5.2 Preliminary results of the new `Fragmentation` module

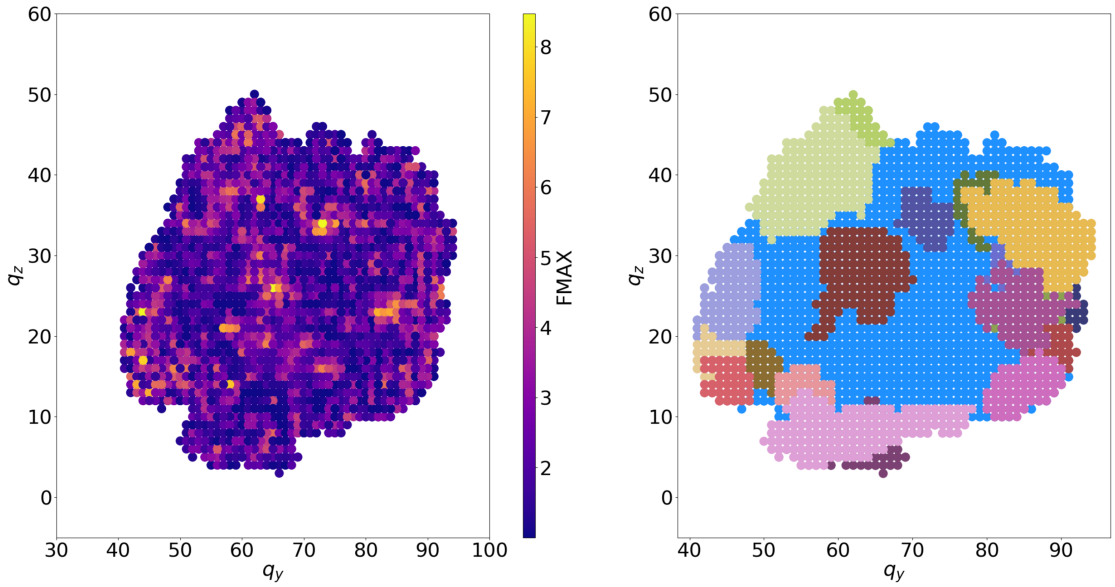


Figure 5.2: Same slice as Figure 2.2, with the corresponding collapse-time field, expressed in term of FMAX, shown in the left panel.

Given the discussion in Section 2.4.2, the halos identified by PINOCCHIO can be interpreted as clusters of particles surrounding local peaks of the collapse-time field. This is illustrated in Figure 5.2, where we show the same slice presented in Figure 2.2, now compared with the corresponding collapse-time map expressed as FMAX = $1 + z_c$. Apart from minor projection effects, it is evident that halos form around particles where FMAX takes higher values. This behavior has a clear physical interpretation: particles with larger FMAX are those predicted to collapse at earlier cosmic times.

In the vicinity of such early-collapsing regions, surrounding particles experience subsequent collapse, and the halos that form there naturally undergo mergers as the hierarchical assembly proceeds. Thus, the collapse-time field encodes part of the structure of the emerging halo network.

Motivated by the interpretation of halos as clusters around local maxima of the collapse–time field, we explored whether a density–based clustering algorithm could be

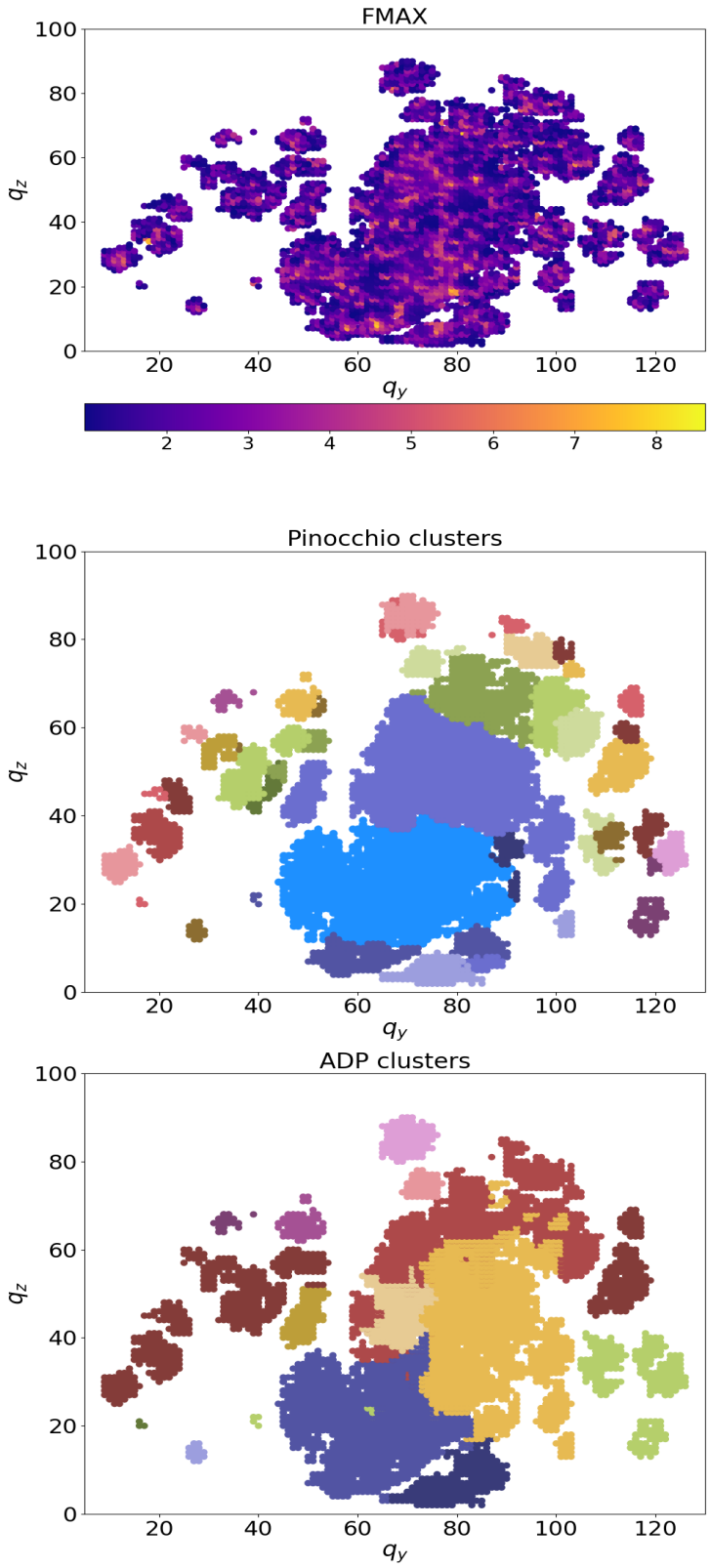


Figure 5.3: Comparison between the collapse–time field (top), the Lagrangian patches identified by the standard `PINOCCHIO` `Fragmentation` (middle), and those obtained with the 3D `ADP` algorithm (bottom).

used as an alternative Fragmentation strategy. For this purpose, we adopted a modified version of the Adaptive Density Peak (ADP) algorithm (d'Errico et al. 2021), adapted to operate directly on a regular grid in Lagrangian space. The legacy implementation of ADP, described in Appendix A, was originally designed as a standard clustering algorithm in Eulerian space (x,y,z). That implementation was later extended to a 2D grid-based version for deblending procedures in astronomical images, also documented in the Appendix A.

Building on these developments, we implemented a fully 3D version of ADP capable of working directly on the regular grid of FMAX values produced by PINOCCHIO. A preliminary test on such a grid is shown in Figure 5.3, which is composed of three panels. The upper panel displays a slice of the collapse–time field, taken from a region close to the reference slice shown in Figures 2.2 and 5.2. The middle panel shows the corresponding Lagrangian patches identified by the standard PINOCCHIO Fragmentation module, while the lower panel presents the patches obtained with the 3D ADP algorithm applied directly to the collapse–time grid. Although the comparison is only pictorial, it illustrates both the strengths and limitations of this preliminary approach. In isolated environments, where collapse–time peaks are well separated, ADP successfully reconstructs several Lagrangian patches, recovering structures broadly consistent with those produced by the standard Fragmentation module.

However, in many cases, and particularly in crowded regions, ADP tends to merge multiple nearby patches into a single, artificially large structure. This limitation arises because the collapse-time field alone does not encode the full dynamical information used by PINOCCHIO. In the standard Fragmentation algorithm, LPT displacements, or equivalently, the local velocity field, play a crucial role in determining how structures merge, separate, or accrete. When ADP is applied only to FMAX, all information about these dynamical flows is lost.

Figure 5.4 illustrates this point clearly. The figure shows a thinner slice, taken one grid cell above and below the main slice used in Figures 2.2 and 5.2. This reduced thickness was chosen to avoid overcrowding the plot with velocity vectors. The figure is composed of two panels: the collapse–time field in the left panel and the corresponding Lagrangian patches identified by PINOCCHIO, with the velocity field overplotted, in the right panel. The velocity vectors reveal infall patterns, shear, and anisotropic collapse that are completely invisible in the static collapse-time field. Importantly, these flows do not converge toward the FMAX peaks themselves. High-FMAX regions correspond to early-collapsing, already-virialized cores, which no longer act as dynamical attractors. Instead, the velocity field converges toward the current Eulerian potential minimum, effectively the present-day center of mass (COM) of the halo, where material is still undergoing collapse, often located near regions with lower FMAX. Because this crucial dynamical component is

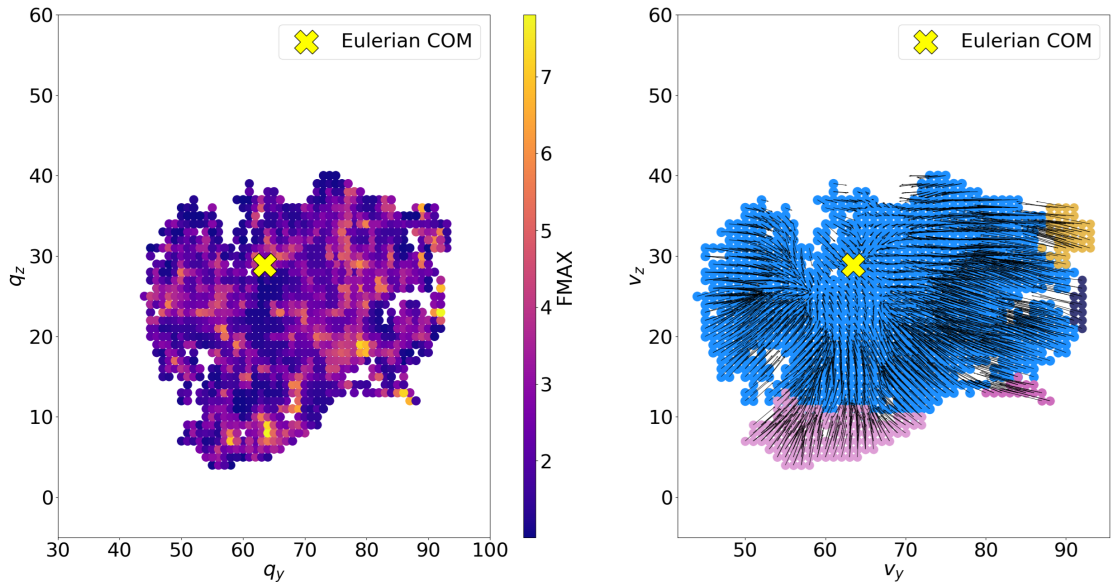


Figure 5.4: Collapse–time field (left) and corresponding Lagrangian patches identified by PINOCCHIO with the velocity field overplotted (right). The Eulerian COM is marked in yellow.

missing in the FMAX-only approach, using FMAX alone leads to ambiguous or incorrect halo boundaries in dense regions, where several collapse-time peaks may lie close together in Lagrangian space but have very different dynamical fates.

Incorporating velocity information into the 3D ADP framework is therefore the natural next step toward improving this new fragmentation scheme. Work in this direction is ongoing.

Chapter 6

Tracing cosmic voids with PINOCCHIO

This Chapter presents the novel application of PINOCCHIO to cosmic voids and assesses its ability to reproduce void statistics relative to a full N-body simulation. We begin by motivating the void statistics considered in this work and discussing their cosmological relevance. We then describe the void finder used in the analysis, followed by an overview of the void summary statistics employed in this study and the methodology adopted to measure them. The simulation setup is then outlined, including a description of the OpenGADGET3 N-body code employed in this study. We then compare the halo mass functions derived from both simulations, including the mass correction applied to OpenGADGET3 halos and the procedure used to match their number densities. Building on this foundation, we present a detailed comparison of the void statistics obtained from the two simulations. The conclusions of this study are reported in Chapter 7.

The content of this Chapter is based on the manuscript Lepinzan et al. (2025), published on A&A. 704 (2025) A5.

6.1 Cosmic voids as cosmological probes

To test the reliability of PINOCCHIO in predicting void statistics, we use the full N-body code OpenGADGET3 as a benchmark. Voids are identified using the VIDE toolkit, which is applied to the halo field (rather than the full matter density field) constructed from the halo catalogs of both simulations. By comparing results across different redshifts and resolutions, we assess how well PINOCCHIO reproduces the statistical and structural properties of voids over cosmic time. We investigate four key summary statistics that characterize cosmic voids: the void size function (VSF), void ellipticity function (VEF), core density function (CDF), and radial density profiles (RDP). The investigation into each of these statistics is motivated below.

As a complementary tool to the analysis of overdense structures such as halos, voids provide an alternative perspective on structure formation. While halos grow through mass accretion and gravitational collapse, reducing their comoving volume, voids expand. Consequently, the variation of the VSF as a function of redshift captures the evolution of the Large-Scale Structure (LSS) from a unique perspective. By tracing the size distribution of these expansive voids, the VSF contains valuable information on various cosmological parameters, complementing standard probes (Pisani et al. 2015; Contarini et al. 2019, 2023), which typically focus on overdensities.

Beyond providing insights into individual void morphologies, the resulting VEF carries crucial cosmological information. The anisotropic growth of cosmic structures, shaped by tidal forces and surrounding matter distributions (Park & Lee 2007; Schuster et al. 2023), directly influences void deformation. As matter collapses into cosmic structures the surrounding voids deform accordingly. Since void measurements are largely unaffected by systematics from baryonic physics (Schuster et al. 2024), even though minor effect are expected (Paillas et al. 2017), the redshift evolution of their shape distribution serves as a valuable tracer of dark energy (Lee & Park 2009; Bos et al. 2012; Schuster et al. 2025). Additionally, the average stretching of voids along the line of sight enables tests of cosmic expansion using the Alcock-Paczynski effect (Alcock & Paczynski 1979; Sutter et al. 2014).

The CDF encodes information about the variations in void emptiness, the processes driving matter evacuation, and the interaction of voids with their surrounding structures. Therefore the CDF provides valuable insights into the underlying cosmology, particularly in scenarios involving massive neutrinos (Schuster et al. 2019, 2023).

The stacked RDP are a powerful probe of fundamental physics, particularly in testing deviations from GR and probing the late-time evolution of the Universe. Screening mechanisms suppress the fifth force in overdense regions, but become inefficient in void interiors, and deviations from GR therefore manifest in the RDP. In particular, modified gravity models often predict enhanced void expansion, leading to deeper void centers and steeper compensation walls compared to the predictions of ΛCDM (Perico et al. 2019; Contarini et al. 2021). The shape of the RDP is also directly linked to the Integrated Sachs-Wolfe (ISW) effect (Sachs & Wolfe 1967), as voids dynamically evolve within the cosmic web. In ΛCDM, decaying void potentials cause a colder ISW imprint, modifying the expected signal (Ilic et al. 2013; Kovács 2018).

6.2 Void Identification

In this work, we employ the Void Identification and Examination toolkit VIDE [1] (Sutter et al. 2015) to identify cosmic voids from a set of biased tracers (halos) for our subsequent analysis. VIDE implements an enhanced version of the ZOnes Bordering On Voidness (ZOBOV) algorithm (Neyrinck 2008), which is based on a watershed technique (Platen et al. 2007) that identifies local watershed basins in a given $3D$ density field, and is outlined as follows. The process begins with a Voronoi tessellation of the tracer positions, which is then used to estimate the density field. The density at each point is calculated as the inverse of the volume of its corresponding Voronoi cell. Each cell is then grouped with its corresponding neighbor cell with the lowest density value of all neighbors. This step is repeated until all cells are assigned to a group, where each group contains one local minima. These groups of cells are the watershed basins, which correspond to the identified void population.

Once the watershed step is complete, each void is represented by a collection of Voronoi cells with an arbitrarily irregular overall shape. Each void's center is determined by the volume-weighted barycenter of all the Voronoi cells associated with the void. This is calculated by summing over the comoving positions x_j of the tracers, weighted by the volumes of their associated cells:

$$X_V = \frac{\sum_j x_j V_j}{\sum_j V_j} \,. \tag{6.1}$$

The total volume for each void is calculated as the sum of the volumes of the Voronoi cells j that belong to that void. The effective radius $R_{\rm eff}$ is then defined as the radius of a sphere with an equivalent volume V:

$$R_{\rm eff} = \left(\frac{3}{4\pi} \sum_j V_j \right)^{1/3} \,. \tag{6.2}$$

Additionally, VIDE quantifies the shape of the voids by computing their inertia tensor:

$$\begin{aligned} M_{xx} &= \sum_j \left(y_j^2 + z_j^2 \right) \,, \\ M_{xy} &= -\sum_j \left(x_j \, y_j \right) \,, \end{aligned} \tag{6.3}$$

with x_j, y_j and z_j representing the co-moving coordinates of the tracers relative to the void center defined in Eq. (6.1). The remaining components of the inertia tensor are calculated

[1] https://bitbucket.org/cosmicvoids/vide_public/

similarly to Eq. (6.3).

From this, the void ellipticity is then defined in terms of the smallest J_1 and largest J_3 eigenvalues of the inertia tensor:

$$\epsilon = 1 - \left(\frac{J_1}{J_3}\right)^{1/4} . \tag{6.4}$$

Another key void property, is the core density, denoted as $\hat{n}_C$:

$$\hat{n}_C = \frac{n_{\rm core}}{\bar{n}_t} , \tag{6.5}$$

which represents the density of the largest Voronoi cell of a void, $n_{\rm core}$, corresponding to the region of lowest density of a particular void. This quantity, computed by VIDE is expressed relative to the mean density of the tracer population $\bar{n}_t$.

Lastly, to calculate void radial tracer density profiles, following (Hamaus et al. 2014; Schuster et al. 2023), the (number) density within radial shells of thickness $2\delta r$ at a given co-moving distance r from the center of a single void i can be defined as:

$$\rho_V^{(i)}(r) = \frac{3}{4\pi} \sum_j \frac{\Theta(r_j)}{(r+\delta r)^3 - (r-\delta r)^3} , \tag{6.6}$$

where $\Theta(r_j)$ is defined through two Heaviside step functions ϑ, which specify the radial bin:

$$\Theta(r_j) \equiv \vartheta[r_j - (r-\delta r)]\vartheta[-(r_j - \delta r)^3] . \tag{6.7}$$

Here, r_j denotes the co-moving distance of the j-th tracer from the void center, while δr determines the shell thickness. The summation in Eq. (6.6) includes all tracers j within a specified distance from the void.

In our analysis, VIDE was run with no mergin (`mergingThreshold` = 10^{-9}), effectively disabling the merging of neighbouring watershed basins. In this configuration, each void corresponds to an individual density minimum identified by the watershed algorithm, without hierarchical merging into larger composite structures. A low threshold prevents voids from extending into overdense ridges, thereby limiting the development of a void hierarchy. Conversely, increasing the threshold (to values of order unity or higher) allows neighbouring basins to merge and produces a nested hierarchy of voids with multiple sub-void levels, although the total number of identified density minima remains unchanged.

In the literature, a commonly adopted merging threshold is $0.2\bar{n}$, where $\bar{n}$ denotes the mean tracer number density. This choice has a physical motivation: in the spherical

expansion model of an inverted top-hat perturbation in an Einstein–de Sitter universe, the matter density inside the perturbation reaches approximately 20% of the cosmic mean at the boundary when shell crossing occurs (Blumenthal et al. 1992; Sheth & van de Weygaert 2004; Neyrinck 2008). However, this interpretation strictly applies only when the threshold is defined in the full matter density field and under the assumption of spherical symmetry. When voids are identified in the tracer number-density field, additional effects must be considered. In particular, tracer bias alters the relation between tracer and matter densities, and the finite sampling of tracers introduces shot noise and an effective smoothing of the density field below the typical inter-particle separation. Both effects can modify the density ridges that determine whether neighbouring basins are merged, thereby weakening the direct physical interpretation of the $0.2\bar{n}$ criterion.

For the purposes of this work, merging was disabled to ensure a consistent and minimally model-dependent comparison between PINOCCHIO and OpenGADGET3, avoiding additional sensitivity to the ridge-density threshold that governs hierarchical merging.

6.3 Void statistics

This section introduces the key void summary statistics derived from the VIDE output: VSF, VEF, CDF and RDP. In all cases we analyze these properties at redshifts $z \in \{0.0, 0.5, 1.0, 1.5, 2.0\}$.

- VSF: similar to the HMF, which describes the number density of collapsed objects as a function of their mass and redshift (Press & Schechter 1974; Sheth & Tormen 1999; Castro et al. 2023), the VSF describes the number density of cosmic voids as a function of their size $R_{\rm eff}$ Eq. (6.2) and redshift (Sheth & van de Weygaert 2004; Jennings et al. 2013). We measure the VSF with 17 linearly spaced bins between 10 h^{-1} Mpc and 80 h^{-1} Mpc.

- VEF: although characterizing the nonspherical shape of an unbound system like a void is challenging due to the lack of a clearly defined boundary, the VEF serves as a good first-order probe of deviations from spherical symmetry (Park & Lee 2007). Void shapes are commonly characterized using the inertia tensor (Eq. (6.3)). The departure from sphericity can be quantified by measuring the ellipticity (Eq. (6.4)) of the spatial distribution of the void tracers. We measure the VEF with 11 linearly spaced bins between $\epsilon = 0$ and $\epsilon = 0.4$, as voids at higher ellipticity are extremely sparse.

- CDF: similar to the HMF and VSF, the CDF describes the number density of cosmic

voids as a function of their minimal density n_C (Eq. 6.5), offering insights into the extreme under-dense environments that define their cores (Schuster et al. 2024). We measure the CDF with 12 linearly spaced bins between $n_C = 0.05$ and $n_C = 0.65$.

- RDP: the RDP of a single void measures the density contrast relative to the mean tracer density, which is defined as:

$$RDP_i(r) = \rho_V^{(i)}(r)/\bar{\rho} - 1\,, \tag{6.8}$$

where $\rho_V^{(i)}(r)$ is the void density within a radial shell, as described in Eq. (6.6), and $\bar{\rho}$ is the mean tracer density. We emphasize that, in this work, the density field is inferred using the number density of halos, rather than the underlying full dark matter distribution. This means that the RDP characterizes the distribution of halos (tracers) around voids rather than the total mass content.

Whether measured from halos or the full density field, the RDPs of individual voids are scattered and affected by the underlying resolution. While they do not individually contain much cosmological information, they remain useful for testing specific void characteristics (Schuster et al. 2023, 2024). Furthermore, a more robust and representative characterization of the void profiles can be obtained by stacking the individual void profiles.

The stacked profiles are obtained by first computing the radial density profile for each void individually, then grouping the voids into bins based on their effective radius $R_{\rm eff}$ (Eq. 6.2). The profiles within each bin are averaged to produce a representative stacked profile for that void size range. Given this approach, the final stack is an average of Eq. (6.6):

$$\rho_V(r) = \frac{1}{N_v}\sum_i \rho_V^{(i)}(r)\,. \tag{6.9}$$

accordingly, the resulting RDP is redefined as:

$$RDP(r) = \rho_V(r)/\bar{\rho} - 1. \tag{6.10}$$

This approach allows us to dissect trends in the void halo distribution over the whole void population and over a range of scales, reducing the overall scatter from individual voids (Hamaus et al. 2014; Schuster et al. 2023). We measured the individual RDPs for each void out to $2.5 \times R_{\rm eff}$ using 12 linearly spaced bins, and grouped them into 3 linearly spaced bins of void size reported in Table 6.2.

6.4 Simulation setup

The cosmological simulations used in this study are generated from two distinct methods and codes, an N-body code and a perturbation theory based code. The first is the N-body code OpenGADGET3 (Dolag et al in prep.) described later in this Section. The latter is PINOCCHIO [2].

For both simulations we use a fixed box size (L) of 512 h^{-1} Mpc and two particle resolutions: 512^3 and 1024^3. To ensure consistency in comparing the two approaches, the initial conditions (ICs) used by both simulations were generated with PINOCCHIO at redshift $z = 50$ using 3LPT. This allows us to trace the evolution of the same initial density fields across redshift with different simulation codes.

The simulation outputs were recorded at five different redshifts $z = (0.0, 0.5, 1.0, 1.5, 2.0)$, with the goal of evaluating the accuracy with which PINOCCHIO can replicate the statistical properties of cosmic voids in a full N-body simulation, at various stages in cosmic evolution.

Both simulations assume a flat ΛCDM cosmology that matches Planck15 (Ade et al. 2016). The matter density parameter is set to $\Omega_m = 0.315$, which gives a dark energy density of $\Omega_\Lambda = 0.685$. The cosmic baryon density is $\Omega_b = 0.022\,h^{-2}$, with the dimensionless Hubble parameter $h = 0.673$. Finally, the linear power-spectrum is normalized at redshift $z = 0$ by $\sigma_8 = 0.829$, and the primordial spectral index is $n_s = 0.966$.

The largest void identified in the simulation volume has a radius smaller than 100 h^{-1} Mpc, as shown in Section 6.5.1, corresponding to a diameter well below half the box size (L/2). Therefore, even the largest structures considered remain comfortably smaller than the simulation volume and are not artificially truncated by periodic boundary conditions. The fundamental mode of the box, $k_{min} \approx 2\pi/L$, corresponds to scales significantly larger than those probed by the void radii analyzed here, indicating that the dominant large-scale modes relevant for their formation are present in the simulation. While the high-radius tail of the VSF is inevitably more sensitive to cosmic variance in a finite volume, our comparison between PINOCCHIO and OpenGADGET3 is performed within the same simulation box and from identical ICs, such that finite-volume effects largely cancel in the relative analysis. We therefore do not expect the box size to introduce a significant bias in the comparison of the two solvers.

[2] https://github.com/pigimonaco/Pinocchio

6.4.1 OpenGADGET3

OpenGADGET3 (Dolag et al in prep.) is a highly flexible and efficient code used for simulating the gravitational and hydrodynamic evolution of cosmic structures, ranging from galaxies to the large-scale structure of the Universe. The core of OpenGADGET3's gravitational dynamics for a collision-less fluid relies on two complementary schemes: a hierarchical tree algorithm (Barnes & Hut 1986) for short-range interactions and a Particle-Mesh (PM) method (Efstathiou et al. 1985) for long-range contributions. This combination results in a TreePM (Xu 1995) approach, where the PM method significantly reduces the computational complexity of long-range gravitational interactions from $O(N^2)$ to $O(N\log N)$, while the tree algorithm efficiently resolves local gravitational dynamics. These optimizations allow OpenGADGET3 to accurately and efficiently simulate large cosmological volumes (Springel 2005). Although OpenGADGET3 includes Smoothed Particle Hydrodynamics (SPH) for modeling complex gas processes, this work focuses exclusively on DM dynamics, because the void scales investigated in this work are largely unaffected by these more complex processes (Schuster et al. 2024; Lehman et al. 2025). The void radii analyzed lie in the range of several tens of h^{-1} Mpc, up to approximately 100 h^{-1} Mpc, which are more than an order of magnitude larger than typical halo or cluster virial radii ($\approx 1\ h^{-1}$ Mpc). Baryonic processes such as cooling, star formation, and feedback primarily affect the internal structure of halos and the density field on megaparsec and sub-megaparsec scales, while their impact on the large-scale distribution of matter is comparatively small. We therefore do not expect hydrodynamical effects to significantly alter the void statistics considered in this work.

In this work, the ICs are generated using PINOCCHIO and provided as input to OpenGADGET3. The particle distribution then evolves under the influence of gravity, as described above. The tree structure is updated periodically to track the movements of the particles throughout the simulation volume. At each time step, particle positions and velocities are updated based on the gravitational forces using a leapfrog integration scheme (Duncan et al. 1998).

The final halo catalog in OpenGADGET3 is generated on the fly using the SUBFIND algorithm (Springel et al. 2001; Dolag et al. 2009). SUBFIND operates in two steps: first, it uses a Friends-of-Friends (FoF) algorithm to identify parent haloes. Second, SUBFIND detects subhalos by estimating densities via SPH, locating overdense regions, and applying a gravitational unbinding procedure to retain only self-bound structures.

6.4.2 Halo Mass Function: OpenGADGET3 vs PINOCCHIO

Before analyzing voids, it is essential to first examine the halo populations, as the halos serve as tracers for void identification. Ensuring consistency in the halo catalogs between the two simulations is therefore crucial for a fair comparison of the resulting void statistics.

First, we perform some preliminary cleaning of the halo catalogs as output by the two simulation codes. This involves addressing discrepancies in the halo definitions between OpenGADGET3 and PINOCCHIO. This cleaning process consists of two main steps.

- FoF Mass Consistency: Since PINOCCHIO has been calibrated to match the Watson model (Watson et al. 2013), which is based on the FoF halo mass definition, we ensure consistency by also using FoF masses for OpenGADGET3 halos. However, the FoF algorithm is known for overestimating halo masses, particularly for halos with a low particle count. To mitigate this bias, we apply the correction proposed by (Warren et al. 2006), which adjusts the particle count as follow:

$$N_{\text{corrected}} = N(1 - N^{-0.6}), \tag{6.11}$$

  where N is the number of particles in the halo. This correction is applied to the OpenGADGET3 halo masses in our analysis.

- Number Density Matching: Although PINOCCHIO is calibrated to match the FoF HMF, its accuracy is limited by systematic differences between the two codes, particularly at the low-mass end, and motivates the need for further calibration.

  Given that void statistics depend on the properties of the underlying tracer distribution used to define the voids, to ensure a fair comparison between OpenGADGET3 and PINOCCHIO we also match the number density of halos in the two halo catalogs.

  To achieve this, we first apply a mass threshold cut of $10^{13} M_{\odot}/h$ to the corrected OpenGADGET3 masses, and count the number of halos above this threshold. We then select the same number of halos from the PINOCCHIO catalog, sorted in decreasing mass, to construct the corresponding PINOCCHIO catalog. This number density matching serves as a halo catalog calibration, analogous to the procedure used in Fumagalli et al. (2021), because both simulations in our analysis share the same ICs.

  This calibration is necessary because the HMF is sensitive to differences in the methods used to generate halo catalogs. These include differences in simulation codes, halo definitions, and numerical resolution (Castro et al. 2023). Therefore, calibrating the HMF by accounting for these systematic differences between codes

is crucial for fair comparisons and the robustness of our results. This choice ensures that the comparison is performed between halo populations with similar mass distributions and effective bias. Random subsampling to match number density would instead select a heterogeneous population spanning a broader mass range, potentially altering the tracer bias and introducing additional stochastic differences unrelated to the gravitational solver itself. Since PINOCCHIO is known to achieve a near one-to-one correspondence with N-body halos at the high-mass end, while increased scatter appears toward lower masses, selecting halos by mass ranking minimizes noise from poorly matched low-mass objects. This approach therefore isolates differences arising from the underlying structure formation model rather than from variations in tracer selection.

The halo mass cut of $10^{13} M_\odot/h$ provides a "golden sample" of voids linked to robustly identified halos, well suited for studies involving luminous red galaxies (LRGs). Extending the framework to lower-mass cut relevant for emission line galaxies (ELGs) will require additional work due to more uncertain halo identification, stronger scale-dependent galaxy-bias, and enhanced non-linear effects. Nonetheless, the stability of PINOCCHIO at higher mass resolution has been tested for other probes, with galaxy clustering accurately reproduced down to $\sim 1.5 \cdot 10^{11} M_\odot/h$ (Monaco et al. 2025), so we do not expect major changes in the void statistics when moving to lower mass cut.

Lowering the mass threshold increases the tracer number density and reduces shot noise, enabling the identification of smaller voids and potentially modifying the normalization and small-radius tail of the void size function. Differences in tracer bias may also affect void density profiles, particularly in redshift-space analyses where scale-dependent bias becomes relevant. These effects are expected to be most pronounced for small voids near the resolution limit, while large voids (tens of h^{-1} Mpc) primarily probe the large-scale matter distribution and are therefore less sensitive to tracer-specific details. On this basis, we do not expect major qualitative changes in the large-scale void statistics when moving to lower mass cuts, although modest quantitative shifts in the small-void regime may arise depending on tracer selection and survey characteristics

These steps ensure that the halo catalogs from the two simulations are statistically comparable, minimizing the impact of halo number density differences on the resulting void statistics. Because the void finder relies on the spatial distribution of tracers and not their masses, matching the number density, rather than applying identical mass thresholds, provides a more meaningful basis for comparison.

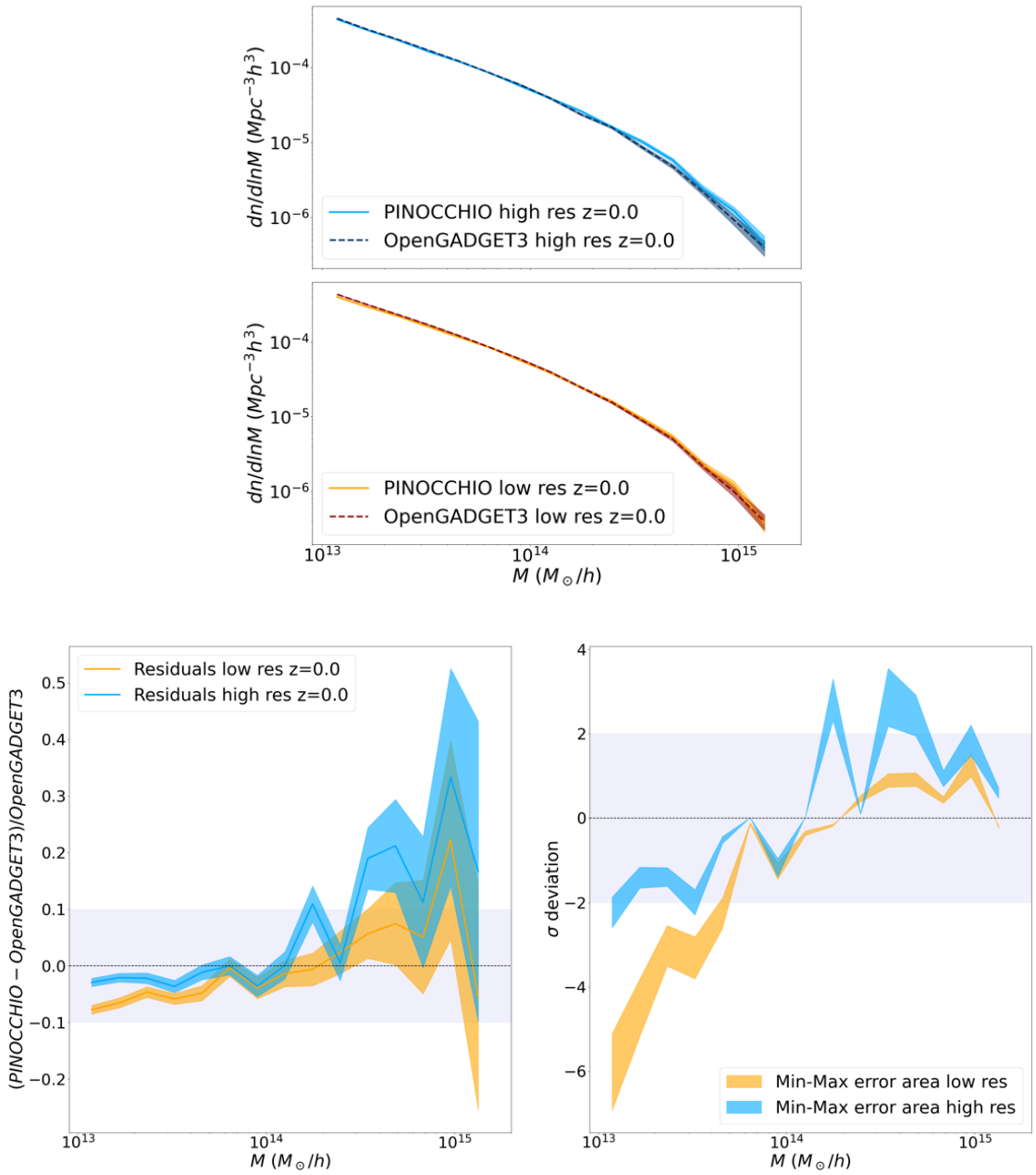


Figure 6.1: Comparison of the HMFs (upper panel) at $z = 0.0$ for low- and high- resolution simulations. Jackknife errors are shown as filled regions. In the bottom left panel, the shaded band indicates a ±10% range around the `OpenGADGET3` result, offering a reference for the level of agreement with `PINOCCHIO`. The filled regions indicates the Jackknife errors around the residual curves. The bottom right panel displays the relative difference between the two HMFs, normalized by the maximum and minimum statistical uncertainties, as described in Eqs. (6.13) and (6.12), and expressed in units of σ. The shaded region indicates the $\pm 2\sigma$ range, highlighting the level of statistical consistency between the two HMFs. The filled region further illustrates the variation range between these two error estimates.

In Figure 6.1, we show the HMFs measured from OpenGADGET3 and PINOCCHIO after the above corrections and criteria have been applied. The figure includes three panels. The upper panel displays the HMFs for both low- and high-resolution simulations ($N_p = 512^3$ and $N_p = 1024^3$, respectively), with their respective Jackknife errors shown as shaded regions. OpenGADGET3 results are shown using dashed lines, while PINOCCHIO is represented by solid lines; different colors are used to distinguish the two resolutions. The bottom left panel shows the difference between the HMFs, normalized by the OpenGADGET3 values. The bottom right panel presents the difference between the HMFs, normalized by the statistical uncertainties using two estimators:

$$\mathrm{Max}_{\mathrm{err}} = \frac{\mathrm{HMF}_{\texttt{PINOCCHIO}} - \mathrm{HMF}_{\texttt{OpenGADGET3}}}{\sigma_{\texttt{OpenGADGET3}}}, \tag{6.12}$$

$$\mathrm{Min}_{\mathrm{err}} = \frac{\mathrm{HMF}_{\texttt{PINOCCHIO}} - \mathrm{HMF}_{\texttt{OpenGADGET3}}}{\sqrt{\sigma^2_{\texttt{PINOCCHIO}} + \sigma^2_{\texttt{OpenGADGET3}}}}, \tag{6.13}$$

where $\sigma_{\texttt{PINOCCHIO}}$ and $\sigma_{\texttt{OpenGADGET3}}$ are the respective Jackknife errors. Since the two simulations share the same ICs, the statistical uncertainties are positively correlated. Ideally, the significance of their differences should be estimated by accounting for this covariance. In the absence of a direct estimate of the covariance, we adopt two approximations that capture the range of plausible significance values. The $\mathrm{Max}_{\mathrm{err}}$ estimator accounts only for the statistical uncertainty of the OpenGADGET3 , effectively assuming that the PINOCCHIO prediction is exact. This results in an upper-limit of the significance. In contrast, the $\mathrm{Min}_{\mathrm{err}}$ estimator assumes uncorrelated errors, thereby overestimating the total variance and yielding a lower-limit significance. Together, these two estimators provide a conservative estimate of the range within which the true significance is expected to lie.

The comparisons in Figure 6.1 show the level of agreement between the OpenGADGET3 and PINOCCHIO HMFs at $z = 0.0$. For both low- and high-resolution simulations, the bottom left panel demonstrates that PINOCCHIO remains well within the $\pm 10\%$ range of OpenGADGET3 for most mass scales, consistent with findings in Munari et al. (2017).

The bottom right panel shows that most values fall within $\pm 2\sigma$, indicating overall consistency with statistical expectations. However, the clear mass-dependent deviations, reaching more than -3σ at the low-mass end, point to potential systematic discrepancies. These discrepancies are likely driven by limitations in the number density matching algorithm, which assumes a one-to-one correspondence between structures in PINOCCHIO and OpenGADGET3. This assumption becomes increasingly prone to systematic effects at lower masses, where the identification of halos is inherently noisier. Nonetheless, the agreement between the two methods remains within 10%, suggesting that we are limited

more by systematics than by statistical uncertainties in this regime.

Figure 6.2 repeats the measurements defined in Eq. (6.12) for multiple redshifts. The figure consists of two panels: the top panel corresponds to measurements from the low-resolution simulations ($N_p = 512^3$), while the bottom panel shows the results from the high-resolution case ($N_p = 1024^3$). The left panels results confirm that the agreement between PINOCCHIO and OpenGADGET3 is consistent across a broad range of redshifts and

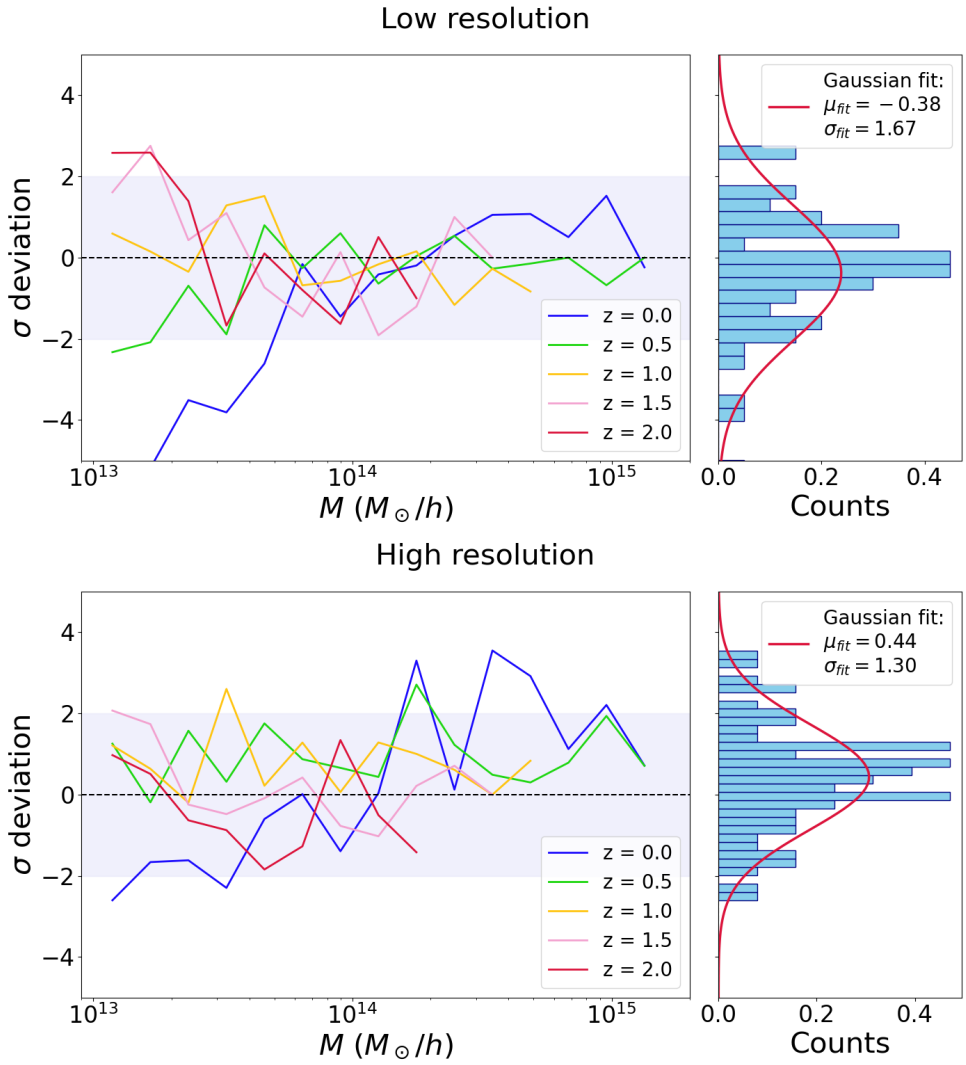


Figure 6.2: Left panels: relative difference between the two HMFs at different redshifts $z = (0.0, 0.5, 1.0, 1.5, 2.0)$, with low-resolution results shown on the top and high-resolution results on the bottom. Each line represents the difference between PINOCCHIO and OpenGADGET3 normalized by the Jackknife errors from the OpenGADGET3 estimates as defined in Eq. (6.12). The shaded area indicates the $\pm 2\sigma$ range, highlighting the region of statistical agreement between the two methods. The horizontal dashed line at $\sigma = 0$ serves as a reference for perfect agreement. Right panels: overall distribution of the measurements in the left panels, aggregated over all redshifts and size bins, with overlaid Gaussian fits.

halo masses. The $\pm 2\sigma$ shaded region illustrates that the two methods remain statistically consistent across most mass scales and redshift intervals. However, particularly in the upper panel, deviations reaching up to $\pm 3\sigma$ in low mass bins, are observed, similar to the trends seen in the bottom right panel of Figure 6.1. These larger deviations arise not from large absolute discrepancies, but from the small statistical uncertainties at these mass scales. As a result, the $\sigma_{\texttt{OpenGADGET3}}$-normalized relative difference (Eq. (6.12)) become more sensitive to even modest mismatches. This regime is therefore limited more by systematic effects than by statistical noise.

The right panels provide a complementary, global diagnostic through the distribution of all the deviation values aggregated across mass bins and redshifts. In an ideal case, perfect agreement between the methods would yield a Gaussian with mean $\mu_{fit} = 0$ and width $\sigma_{fit} = 1$, indicating that the residual scatter is entirely due to statistical uncertainties. In both resolution cases, the fitted distributions show $|\mu_{fit}| - \sigma_{fit} < 0$, implying that the residuals do not present a significant bias. Widths broader than unity ($\sigma_{\text{fit}} > 1$), suggest that the observed scatter slightly exceeds what is expected from purely statistical fluctuations. This excess variance likely reflects localized systematics, such as those observed in the low-mass bins, rather than a global mismatch. This supports the conclusion that, despite some small deviations, the overall statistical consistency between PINOCCHIO and OpenGADGET3 remains robust.

As voids are underdense regions, they are less affected by non-linear gravitational effects, and they are particularly well-suited for semi-analytic methods. Given this context, the same level of agreement observed here is expected for the subsequent void statistics, which we investigate in the next section.

6.5 Void statistics: PINOCCHIO vs OpenGADGET3

The following sections present summary statistics of cosmic voids, identified with VIDE (Section 6.2) and applied to the halo catalogs of OpenGADGET3 and PINOCCHIO. The analysis presented here follows the cleaning procedures described in Section 6.4.2.

6.5.1 Void size function

The number of voids identified for each redshift z is reported in Table 6.1, showing comparable counts between the two codes in both simulation resolutions. Since the same mass cut is applied at both resolutions, the number of halos remains comparable, leading VIDE to identify a similar number of voids in both cases.

z	PINOCCHIO		OpenGADGET3	
	Low Res	High Res	Low Res	High Res
0.0	597	602	583	591
0.5	468	480	486	516
1.0	344	340	326	361
1.5	211	205	209	216
2.0	107	108	106	113

Table 6.1: Number of voids identified by VIDE. The first column lists the redshifts z, while the second and third columns shows the number of voids identified for both simulation resolutions and codes.

Figure 6.3 presents the same set of measurements as in Figure 6.1, but for the VSFs derived from OpenGADGET3 and PINOCCHIO. The upper panel show that the two codes agree well for both high and low resolution at z = 0. Both PINOCCHIO and OpenGADGET3 yield VSFs that have qualitatively the same shape. The two codes agree well around the peak of the distributions, with larger discrepancies at the low- and high-size tails. This is verified by the bottom left panel, which demonstrate that PINOCCHIO remains within the $\pm 10\%$ range of OpenGADGET3 for $25\text{Mpc}/h < R_{\text{eff}} < 55\text{Mpc}/h$ (corresponding to $\sim 73\%$ of the total number of voids at both resolutions), consistent with the HMFs comparisons discussed in Section 6.4.2. As in the HMF comparison, the bottom right panel shows that most values fall within $\pm 2\sigma$, suggesting that the observed deviations are consistent with statistical fluctuations (even though 20–80% discrepancies may appear large), with no evidence of a systematic bias across the R_{eff} range.

Even though the halo catalogs are matched in number density, as described in Section 6.4.2, the clustering of halos in PINOCCHIO has less power at small scales compared to OpenGADGET3. Since VIDE estimates the density field using a Voronoi tessellation, differences in tracer clustering can influence void identification and subsequent void properties. In particular, the weaker clustering of PINOCCHIO halos leads to a coarser density field, which alters how voids are detected and classified. This reduced small-scale power directly affects the VSF. In PINOCCHIO, weaker nonlinear clustering suppresses halo collapse and merging relative to OpenGADGET3, allowing more small voids in the void-in-cloud regime to survive rather than being erased by environmental collapse. For larger voids, the lack of small scale overdensities in PINOCCHIO can reduce the fragmentation of large underdense regions, potentially leading to an an excess of large voids relative to OpenGADGET3. At lower resolution, however, the under-resolved density field can lead to spurious overdense bridges that artificially split or isolate underdense regions, suppressing the formation of large voids through void-in-void merging. The improved agreement between the codes for the largest voids in low-resolution simulations likely reflects a

regime where both are similarly limited in resolving small-scale structure.

Additionally, differences in resolution and tracer populations further contribute to the observed discrepancies. At high resolution, PINOCCHIO shows an excess of large voids relative to OpenGADGET3, possibly due to its improved ability to resolve intermediate-mass halos and thereby alter the reconstructed density field. As shown in Section 6.6, the VSFs in PINOCCHIO vary more strongly with resolution than in OpenGADGET3, which remains largely stable. This suggests a stronger resolution dependence in PINOCCHIO, although the statistical significance of this trend remains limited.

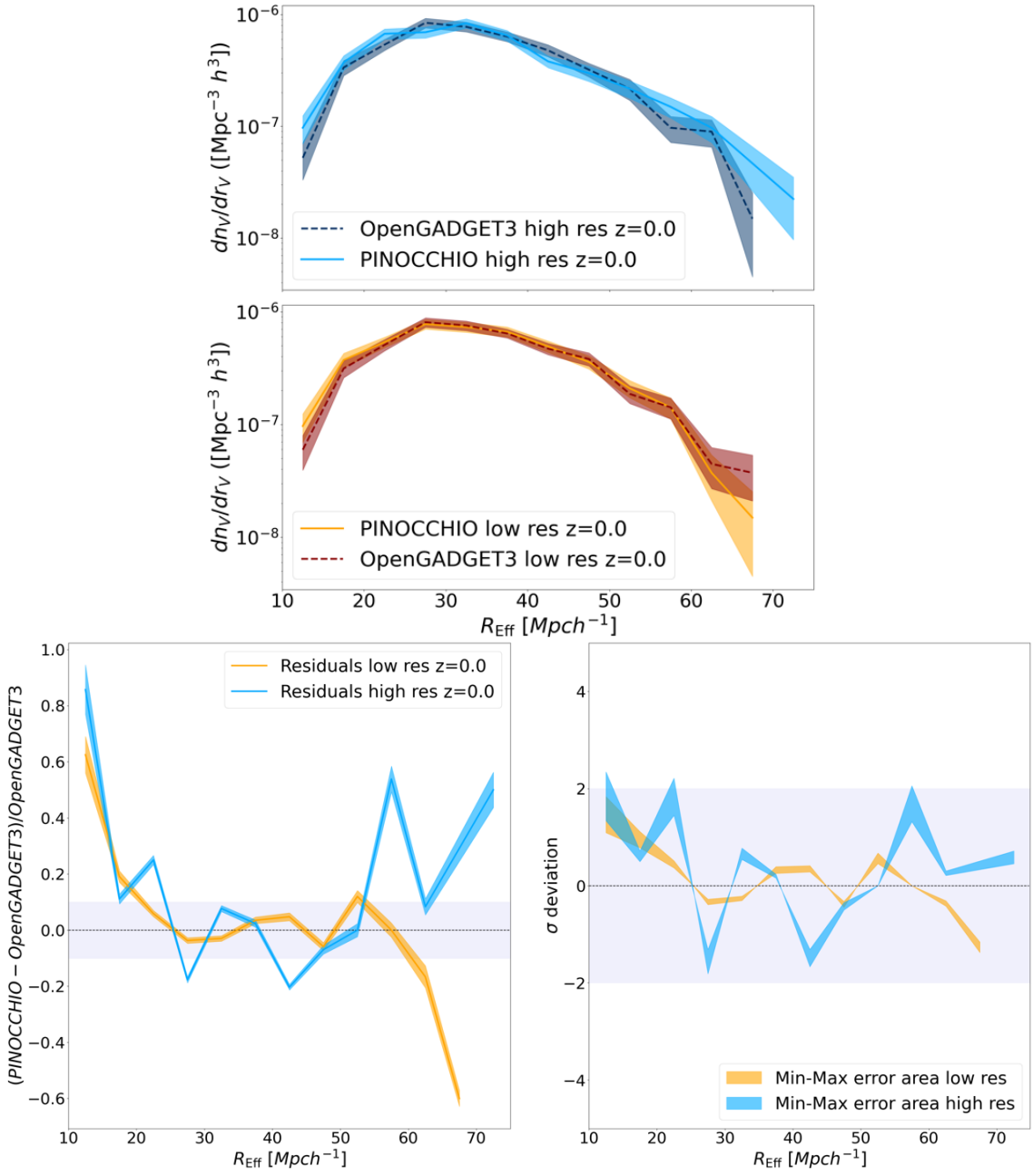


Figure 6.3: Same as Figure 6.1, but for the VSFs.

Figure 6.4 repeats the measurements from Figure 6.2 but for the VSFs at multiple resdshifts. These results confirm the robustness of the PINOCCHIO method compared to OpenGADGET3 across cosmic time. For both resolutions, the agreement remains well within the $\pm 2\sigma$ shaded region for most void size scales, demonstrating reliable performance across a wide range of epochs. As for the HMF, the right panels show that in both resolution cases, the fitted distributions show $|\mu_{fit}| - \sigma_{fit} < 0$, implying that residuals are symmetrically distributed around zero and no significant bias is present, when marginalising over redshift. This indicates that the observed deviations are primarily statistical in nature and could potentially be reduced with improved measurements, such as larger sample size by increasing the resolution and/or box size.

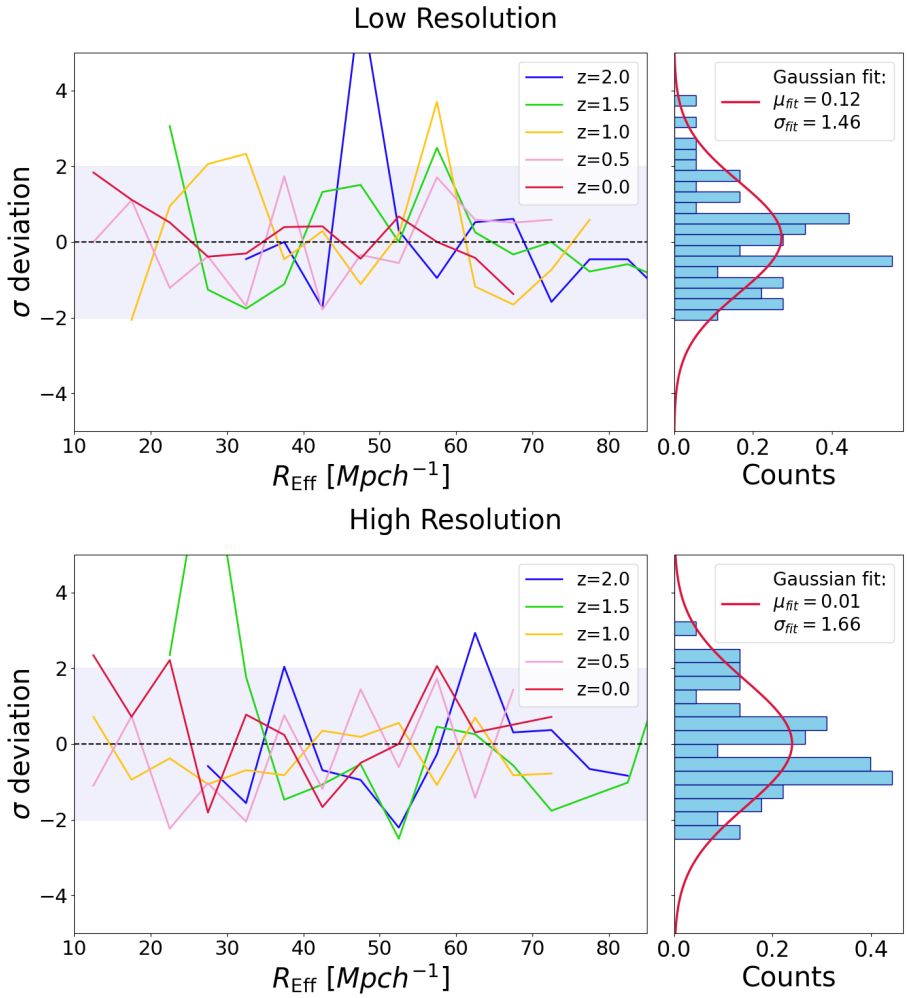


Figure 6.4: Same as Figure 6.2, but for the VSFs as a function of R_{Eff}.

6.5.2 Void ellipticity function

Figure 6.5 presents the same set of measurements as in Figure 6.1, but for the VEFs. The upper panel highlight a reasonable agreement between the two methods. For both low- and high-resolution simulations, the four curves, along with their respective error bars, exhibit substantial overlap, with a stronger peak in the ellipticity distribution given by PINOCCHIO in the high resolution case. The bottom left panels indicate that PINOCCHIO generally follows the OpenGADGET3 ellipticity distribution within the ±10% deviation range over the

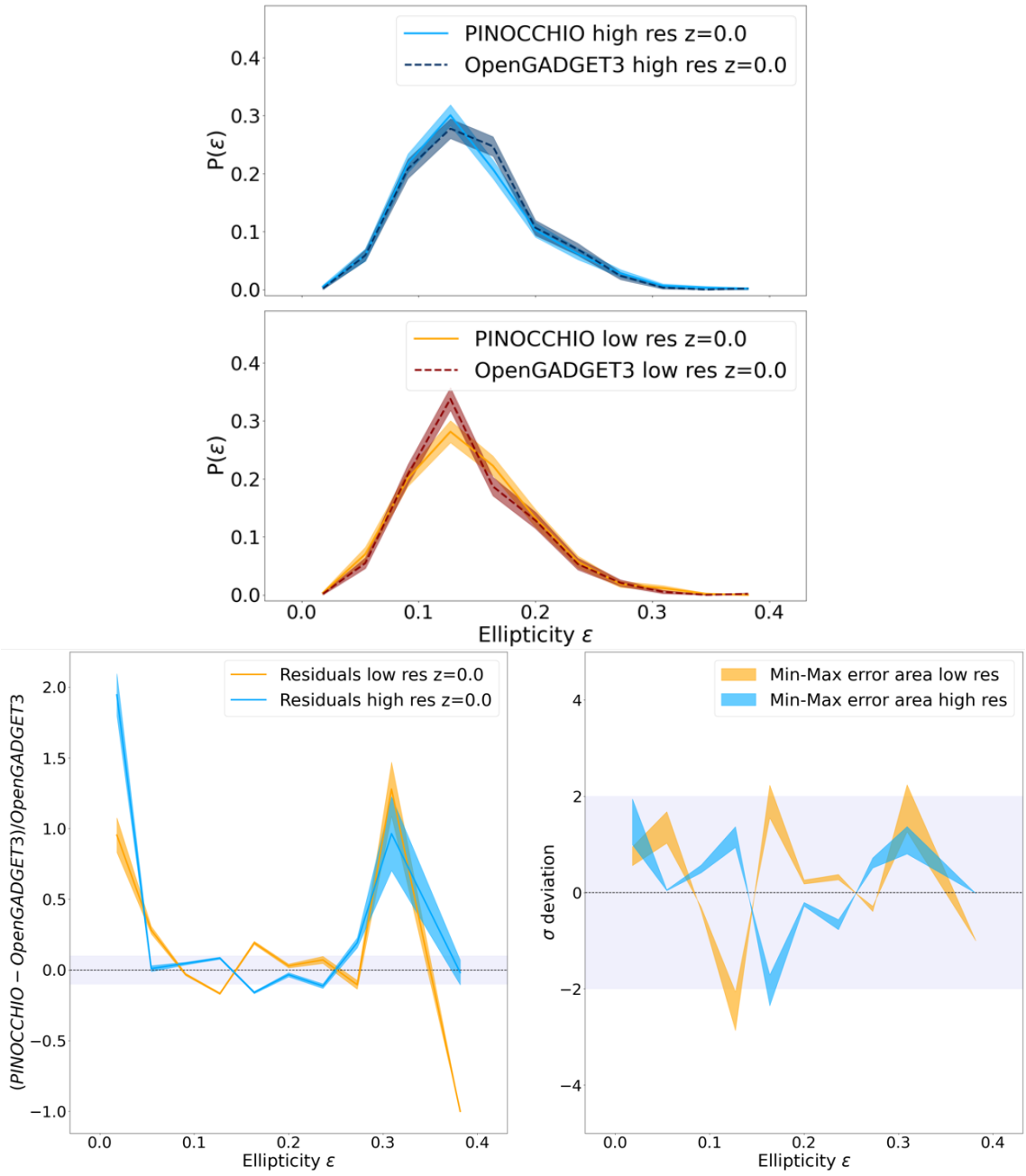


Figure 6.5: Same as Figure 6.1, but for the VEFs.

$0.05 < \epsilon < 0.25$, though residuals show noticeable noise, preventing a precise one-to-one correspondence across all bins. The deviations become larger at the extremes of the ellipticity range, which can be attributed to the small number of voids with such shapes and the resulting shot noise. Values of ($\epsilon > 0.4$) are excluded, as void counts in these bins are insufficient to draw statistically meaningful conclusions. As in the VSFs case, the bottom right panel show that most differences fall within $\pm 2\sigma$, consistent with statistical fluctuations and show no clear sign of systematic bias across the ϵ range. Resolution effects appear modest, with broadly similar distributions across the two resolutions, although the higher-resolution case shows slightly better agreement within the $\pm 2\sigma$ range, likely due to the higher resolution tracer sample giving a more robust sampling of the of the underlying density field.

Figure 6.6 extends the comparison of VEFs to multiple redshifts, following the same methodology applied in Figure 6.2. The results confirm that the ellipticity distributions from PINOCCHIO and OpenGADGET3 remain statistically consistent as a function of redshift.

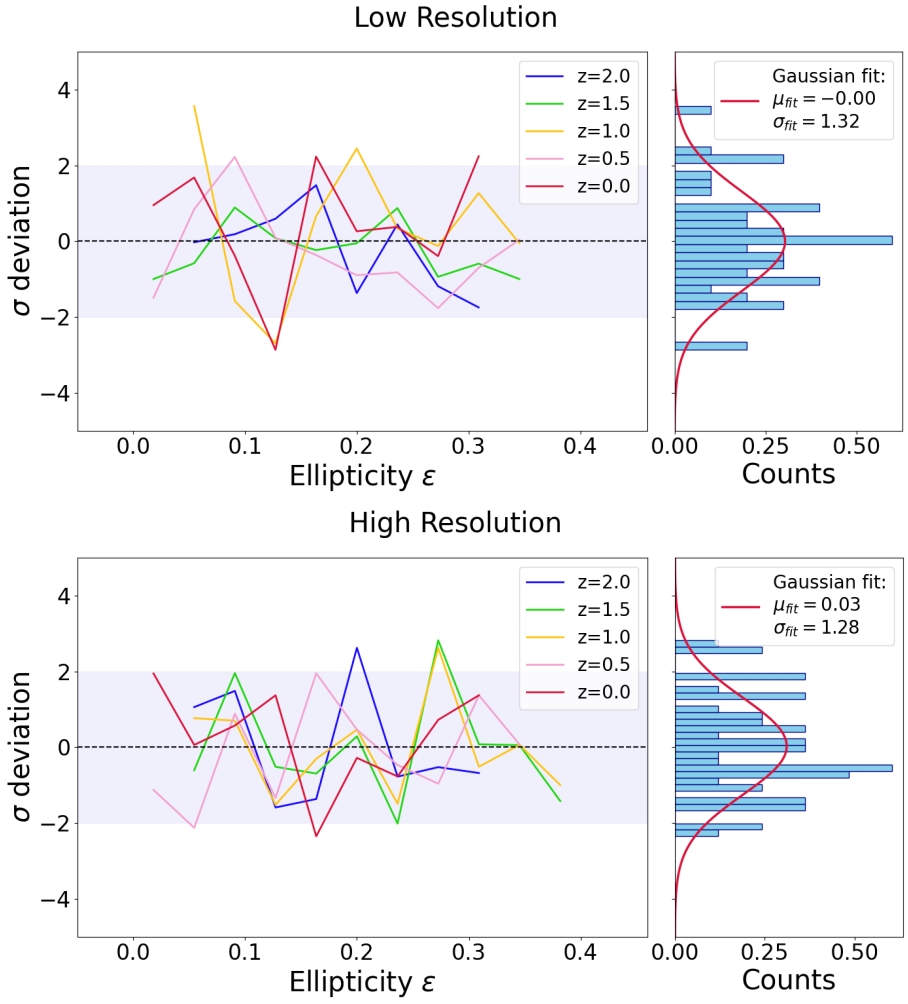


Figure 6.6: Same as Figure 6.2, but for the VEFs as a function of ϵ.

As observed for the VSFs, the differences between the two methods lie within the $\pm 2\sigma$ range for most ellipticity values at all redshifts, suggesting no significant systematics or biases introduced by the PINOCCHIO method. The Gaussian fits in the right panels further support this, showing a slightly better agreement relative to the VSF case, with $\mu_{\rm fit} \approx 0$ and $\sigma_{\rm fit} \approx 1.3$ for both resolutions.

Interestingly, the evolution of the VEFs σ deviations with redshift reveals that, even as voids tend to become slightly more elliptical at earlier epochs due to stronger tidal interactions, PINOCCHIO effectively captures this trend in alignment with OpenGADGET3. This consistency demonstrates the reliability of PINOCCHIO in reproducing the structural and dynamical properties of voids across cosmic time, despite the differences in the underlying halo distributions.

6.5.3 Core density function

Figure 6.7 presents the same set of measurements as in Figure 6.1, but for the CDFs derived from OpenGADGET3 and PINOCCHIO. The comparison in the upper panel reveals a generally good agreement at low core densities $\hat{n}_C$, where both simulations exhibit similar void properties. Since the core densities $\hat{n}_C$, as defined in Eq. (6.5), corresponds to the region of lowest density within a void, the observed behavior suggests that PINOCCHIO performs more accurately in underdense regions compared to overdense regions. This trend is confirmed in the bottom left panel: PINOCCHIO generally follows the OpenGADGET3 core density distribution within the $\pm 10\%$ deviation range over the interval $\sim 0.05 < \hat{n}_C < 0.35 - 0.4$. The distribution in the bottom right panel reinforces the trend seen in previous statistics: the majority of values remain within $\pm 2\sigma$, indicating that differences between PINOCCHIO and OpenGADGET3 are largely driven by statistical fluctuations. Notably, across the full $\hat{n}_C$ range, there is no clear evidence of a systematic bias, further validating the consistency of PINOCCHIO in capturing the low-density interiors of voids.

At higher core densities, a systematic overestimation of void numbers is observed in PINOCCHIO. Unlike the other summary statistics, the CDF appears more challenging for PINOCCHIO to reproduce accurately, though the overall agreement remains within 2σ. We argue that the Fragmentation algorithm in PINOCCHIO (Section 2), may boost the relative number of low-mass halos, as can also be seen from the HMF comparison in Figure 6.1, which in turn enhances the abundance of voids with higher core densities. This interpretation is further supported by the resolution dependence of the discrepancy: better agreement is seen at higher resolution, where Fragmentation is expected to operate more effectively. These trends are illustrated in Section 6.6, where the resolution dependence of the CDFs is explored in more detail.

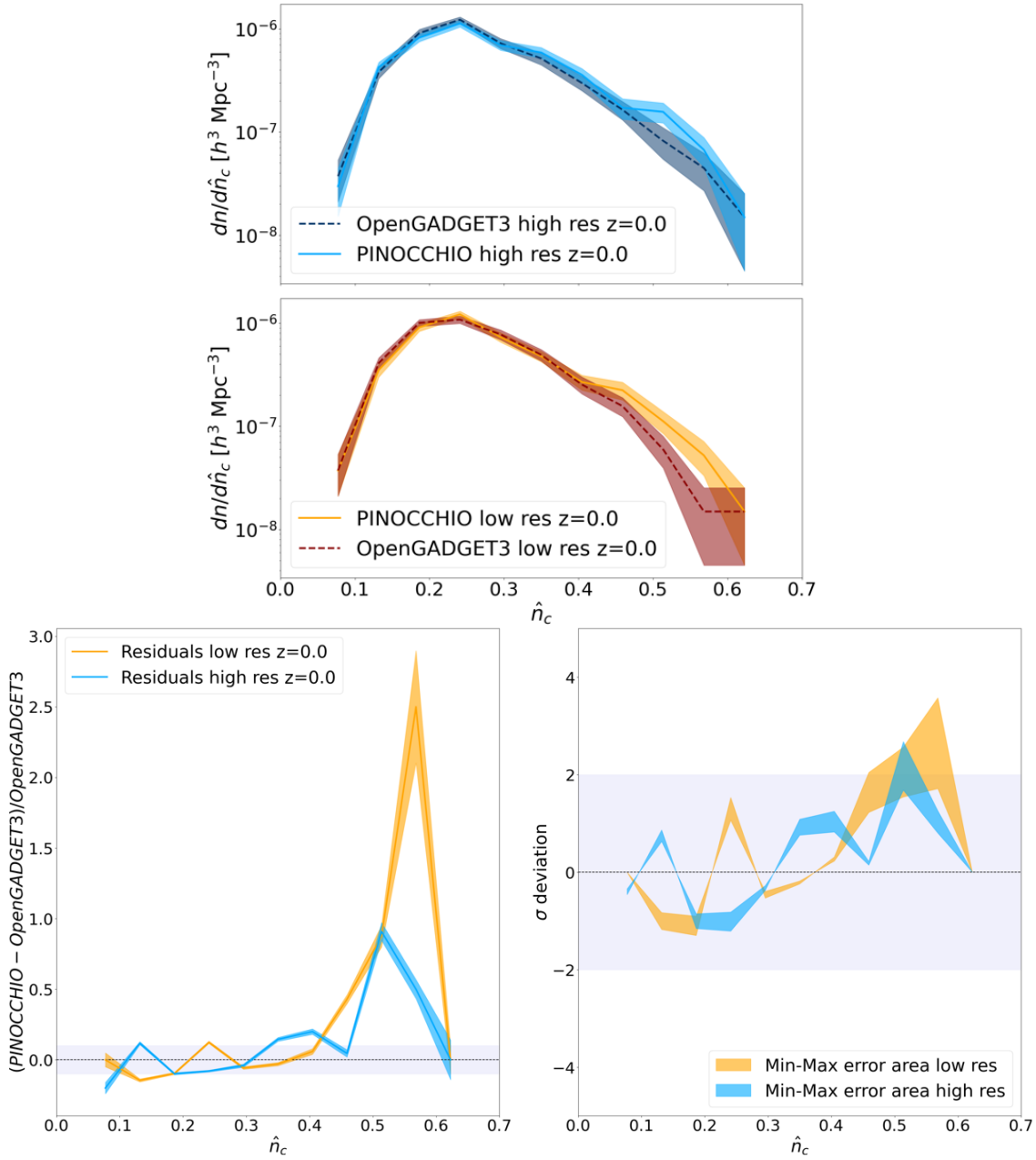


Figure 6.7: Same as Figure 6.1, but for the CDFs.

Figure 6.8 shows the redshift evolution of the CDF comparison, applying the same approach as in Figure 6.2. Across all epochs, the agreement between PINOCCHIO and OpenGADGET3 remains stable, with most deviations falling within the $\pm 2\sigma$ range. The fitted Gaussian distributions in the right panels further suggest that these residuals are statistically distributed around zero, showing no evidence of redshift-dependent biases. This consistency supports the robustness of PINOCCHIO in tracing the evolution of the most underdense regions of the cosmic web. Since such regions are expected to feel the effects of accelerated cosmic expansion earlier in time, the agreement with OpenGADGET3 suggests

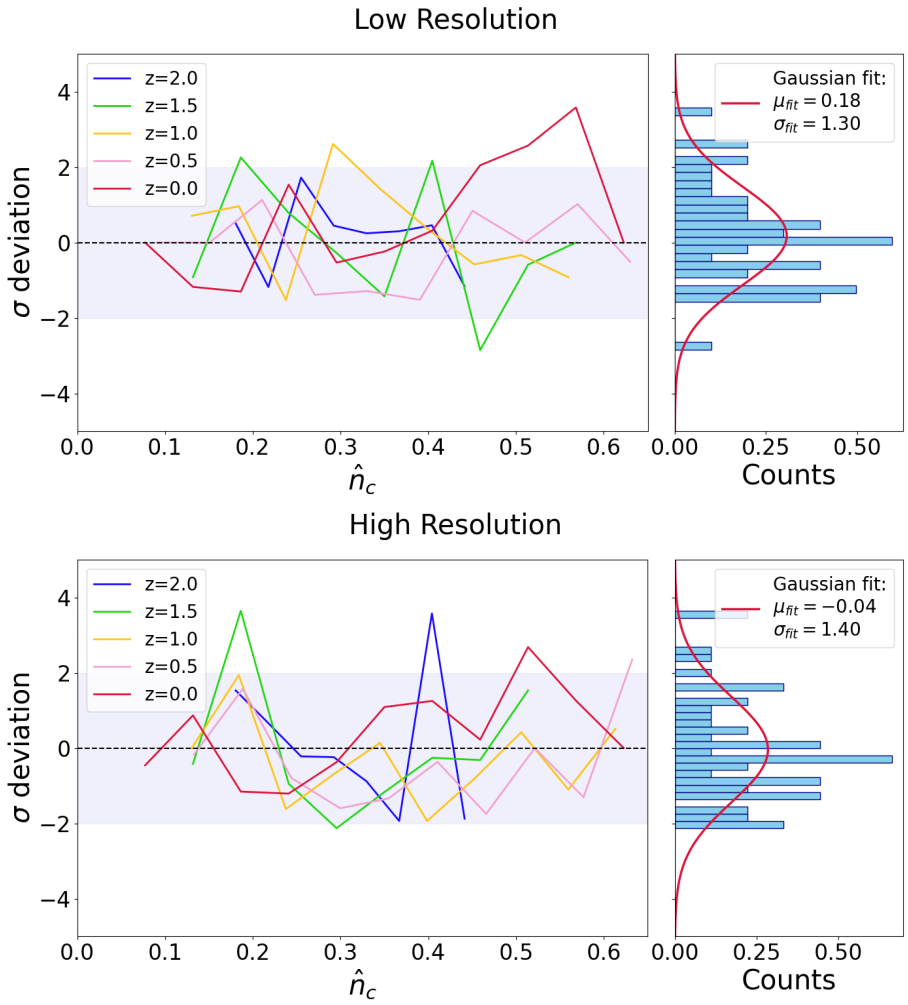


Figure 6.8: Same as Figure 6.2, but for the CDFs as a function of $\hat{n}_C$.

that PINOCCHIO effectively captures this behavior in line with full N-body dynamics.

6.5.4 Radial density profile

Figures 6.9 and 6.10 present the same type of measurement depicted in Figure 6.1, but for the RDPs derived from OpenGADGET3 and PINOCCHIO after applying the stacking procedure described in Section 6.3. The stacking bins are defined using the range of effective void radii $R_{\rm eff}$ from the PINOCCHIO simulation.

This range is then applied consistently to both simulations, ensuring that the void stacking and jackknife resampling are performed over a common reference scale. The average void radii for each bin are indicated in the legend of Figures 6.9 and 6.10, and the bin edges along with the number of voids per simulation are summarized in Table 6.2. The upper panels show that the RDPs, along with their respective error bars, exhibit good overlap across all three linear bins. Although the bin edges are defined based on the

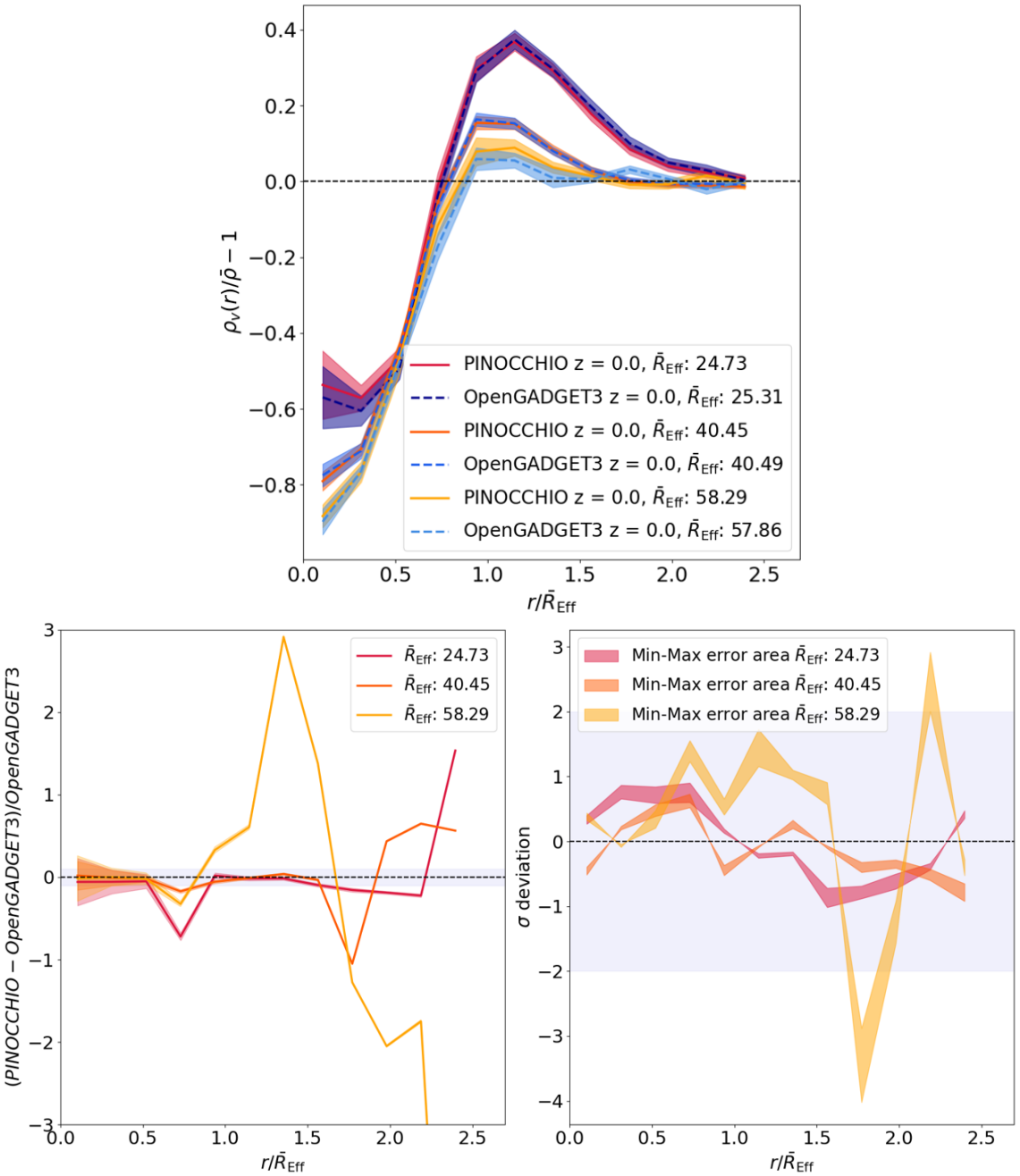


Figure 6.9: Same as Figure 6.1, but for the low resolution RDPs. The upper panel show the stacked radial density profiles, with the legend indicating the mean $R_{\rm eff}$ in each of the three linear bins used for stacking. For clarity, only the mean $\bar{R}_{\rm eff}$ from the PINOCCHIO are reported in the bottom panels.

PINOCCHIO void size range, the average $R_{\rm eff}$ values, indicated in the legends, are closely matched, suggesting that the void populations in OpenGADGET3 and PINOCCHIO are very similar. As expected, the stacked void profiles are deeply underdense at their centers, with the central density generally decreasing as the void size increases. The rise in central

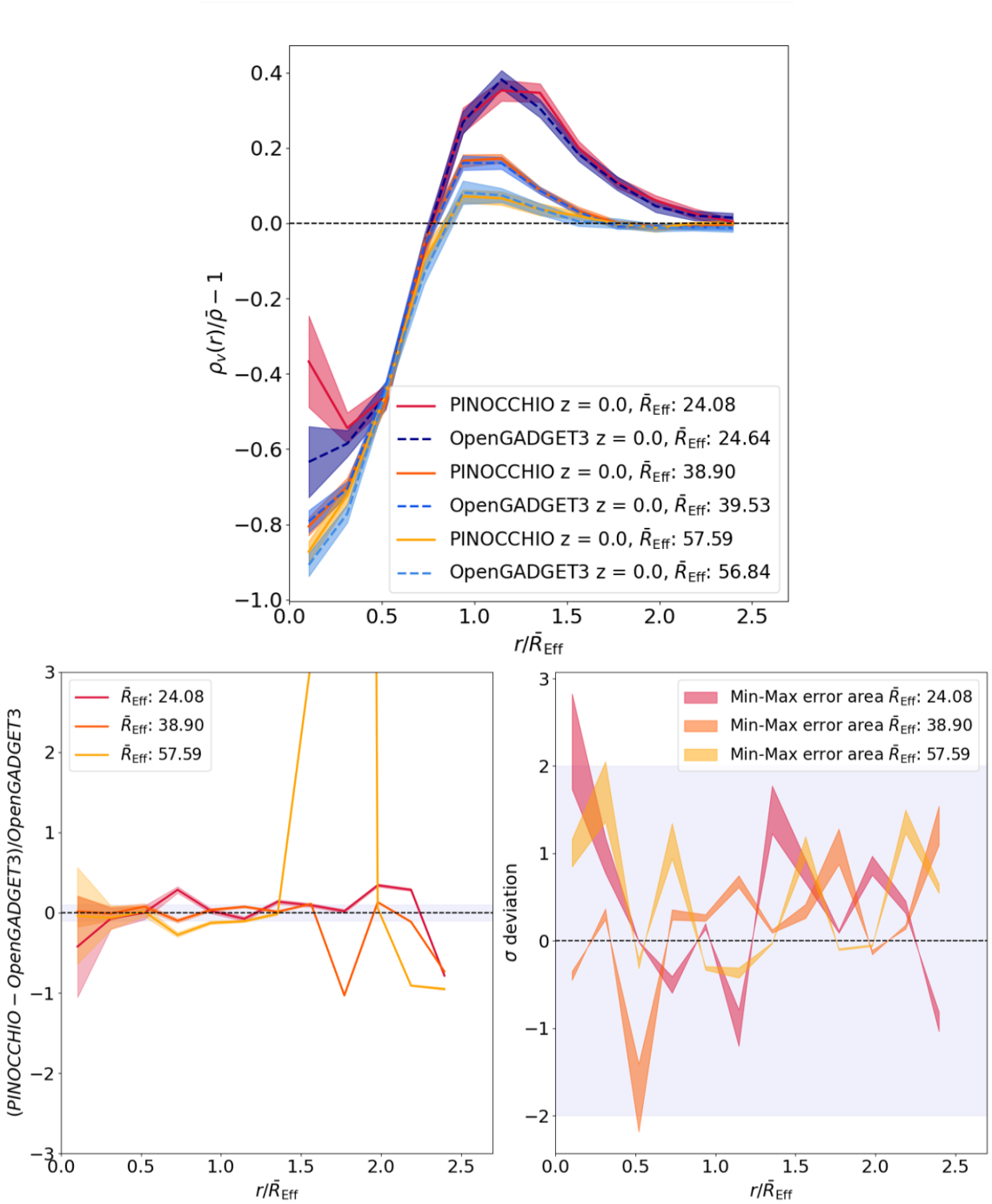


Figure 6.10: Same as Figure 6.9, but for the high resolution RDPs.

density for the smallest voids is due to the limited number of tracers at scales below the mean separation, which can bias the estimate from Eq. (6.6), particularly when a tracer falls within a small central shell (Schuster et al. 2023).

In general, all profiles show clearly defined compensation walls around $r = R_{\rm eff}$, a typical feature of voids identified via the watershed method used in VIDE. As expected, the profiles converge toward the mean background density at sufficiently large distances from

$R_{\rm eff}$ range [Mpc/h]		PINOCCHIO		OpenGADGET3	
Low res	High res	Low res	High res	Low res	High res
[12.08, 32.22]	[11.82, 31.47]	285	277	271	265
[32.22, 52.36]	[31.47, 51.13]	270	267	267	277
[52.36, 72.50]	[51.13, 70.78]	39	55	41	47

Table 6.2: Number of voids per radial bin. The first column lists the bin edges for both low- and high-resolution runs, defined from the PINOCCHIO $R_{\rm eff}$ distribution. The second and third columns report the number of voids from the PINOCCHIO and OpenGADGET3 simulations, respectively, at redshift z = 0.

the void center. The bottom left panels demonstrates that PINOCCHIO generally tracks the OpenGADGET3 profiles within the ±10% deviation range in the inner void regions, up to $r/R_{\rm eff} \sim 0.5$. The agreement is better for smaller voids, while the larger bins show increased noise due to lower void numbers as noted also for the VSF in Figure 6.3 . Additionally, the overall consistency between the two codes appears to improve with higher resolution (Figure 6.10), indicating that increased tracer density enhances the robustness of the RDP measurements. The results in the bottom right panels remain mostly within the $\pm 2\sigma$ interval, reinforcing the interpretation that the differences are dominated by statistical fluctuations rather than persistent biases across the $r/\bar{R}_{\rm eff}$ range.

Figure 6.11 extends the comparison of RDPs to multiple redshifts. It is organized in two rows and two columns: the top panels correspond to the low-resolution simulations ($N_p = 512^3$), while the bottom panels present results from the high-resolution case ($N_p = 1024^3$). The left panels replicate the same measurements of Figure 6.2, for the stacked RDPs for each of the three linear bins across all redshifts. The right panels display the overall distribution of these deviations, aggregated over all redshifts and size bins, with overlaid Gaussian fits. The results confirm that the RDPs from PINOCCHIO and OpenGADGET3 remain statistically consistent over cosmic time. Similar to the VSFs, VEFs, and CDFs, the deviations in the left panels fall within the $\pm 2\sigma$ range with no redshift trend.

The Gaussian fit in the right panels provide a complementary statistical check: the relative difference normalized by the OpenGADGET3 errors (Eq. (6.12)) are centered near zero and exhibit a spread comparable to, or slightly above, the expected statistical noise. This behavior reinforces the view that differences between the two methods arise from random variation rather than systematic bias.

The agreement improves with resolution, reflecting better convergence in the tracer distribution. These findings reinforce the reliability of PINOCCHIO in capturing the large-scale structure around voids. This is especially relevant for studies aiming to extract

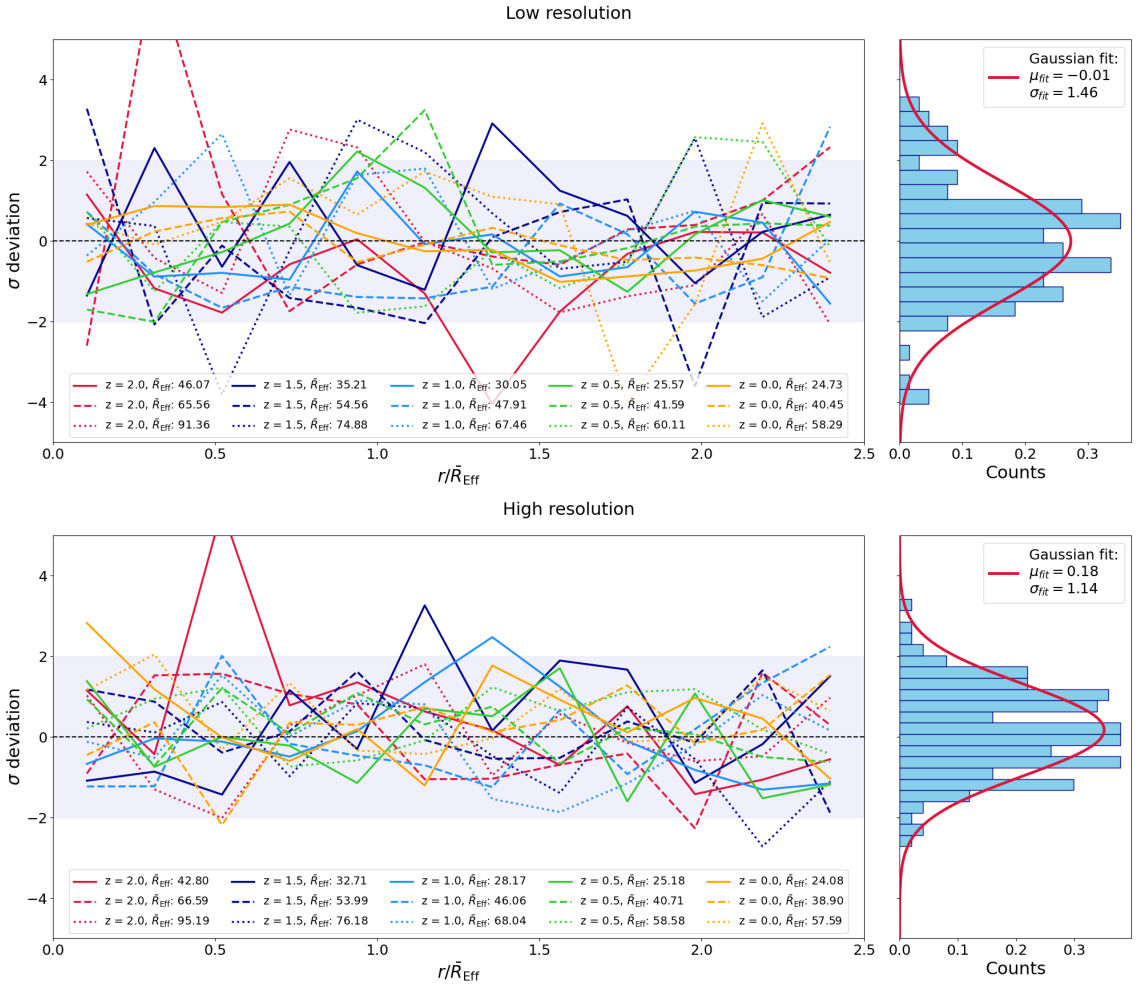


Figure 6.11: Left panels: same as Figure 6.2, but for the RDPs as a function of $r/\bar{R}_{\rm eff}$. The different stacking bins and redshifts are color-coded. Right panels: overall distribution of the measurements in the left panels, aggregated over all redshifts and size bins, with overlaid Gaussian fits.

cosmological information from void environments, including potential constraints on modified gravity or dark energy models through the shape and depth of the RDPs.

6.6 Resolution comparison

In this Section, we present a resolution comparison, i.e., PINOCCHIO vs. PINOCCHIO and OpenGADGET3 vs. OpenGADGET3, for two cosmic void summary statistics analyzed in the main text, where resolution effects appear to be more significant. In particular, Figure 6.12 shows the same measurement as in Figure 6.3, but now comparing different resolutions within each code. The variation in the VSF with resolution is more pronounced in PINOCCHIO than in OpenGADGET3, which remains largely stable. This suggests a

stronger resolution dependence in PINOCCHIO.

Similarly, Figure 6.13 shows the same measurement as in Figure 6.7, again comparing resolutions within each code. The resolution dependence of the discrepancy supports the interpretation above: agreement between the two codes improves at higher resolution, where the Fragmentation algorithm is expected to operate more effectively.

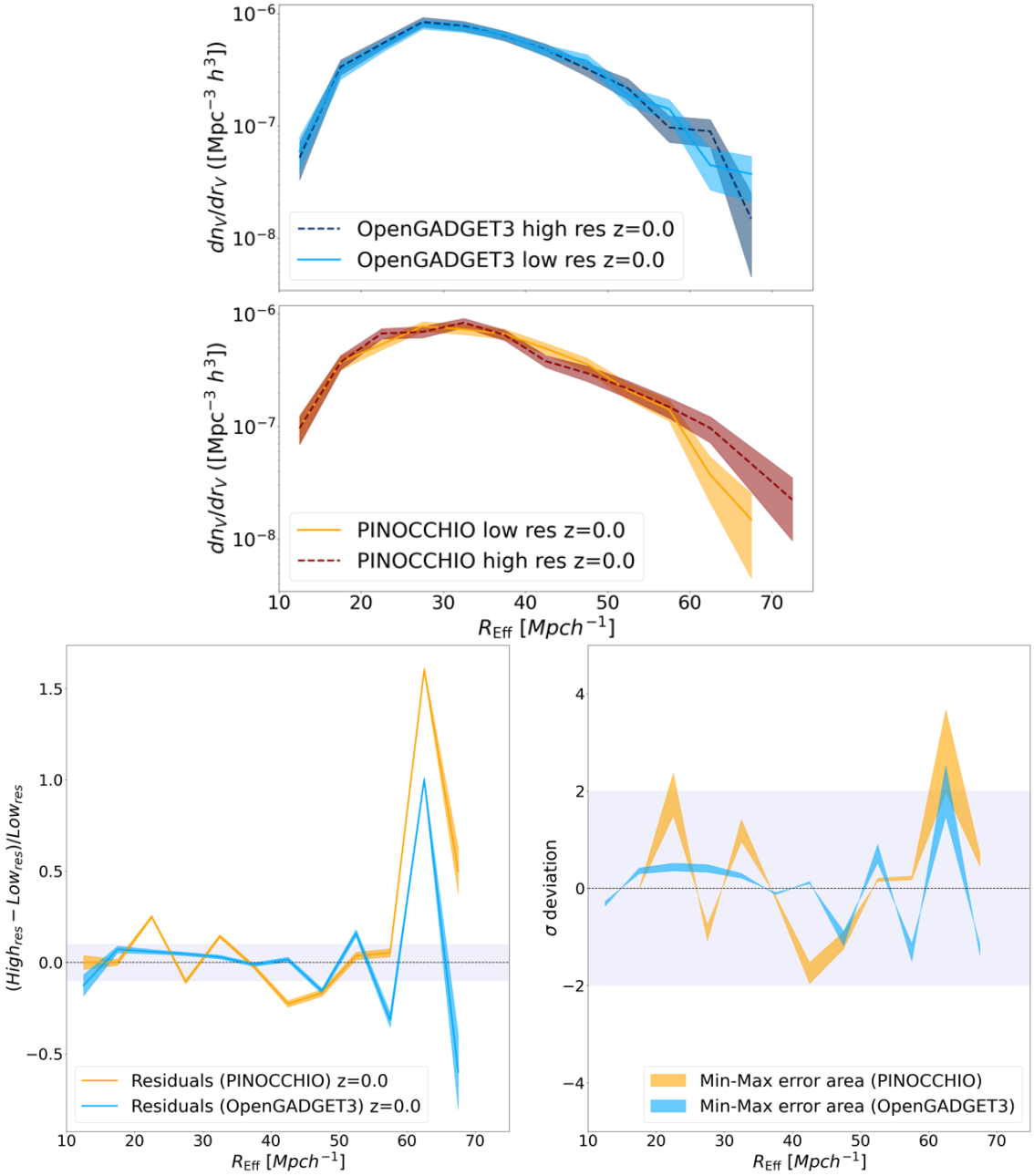


Figure 6.12: Same as Figure 6.3 for VSFs across different resolutions.

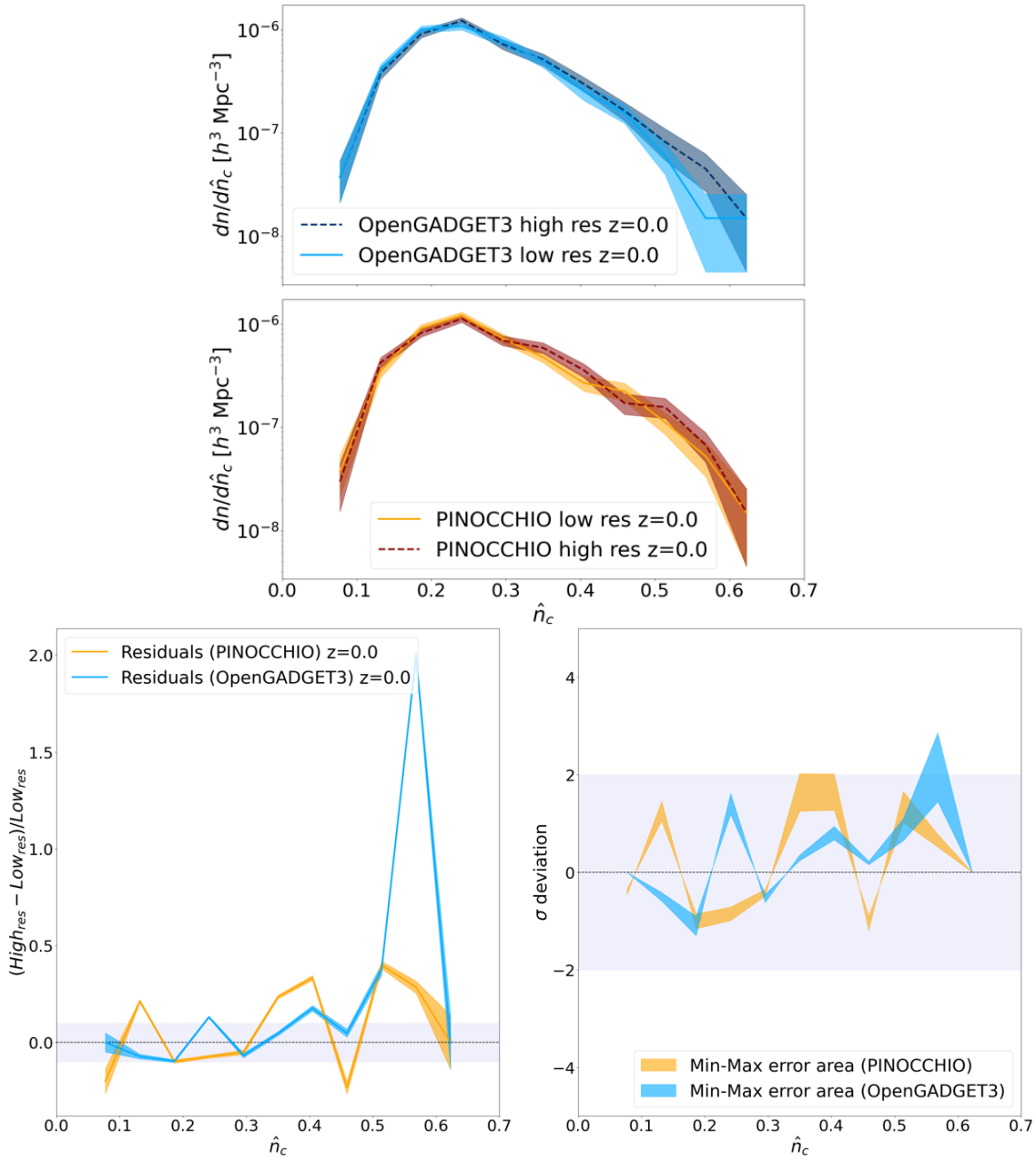


Figure 6.13: Same as Figure 6.7 for CDFs across different resolutions.

6.7 Large box simulation

Since running new pairs of simulations is computationally expensive due to the high cost of OpenGADGET3 runs, we make use of an already available set of larger-box simulations to test the robustness of our analysis with an increased sample of voids. As in the main analysis, the ICs for these simulations were generated with PINOCCHIO using 3LPT at $z = 50$. The simulations have a box size of $3870h^{-1}$Mpc and a particle resolution of 2048^3. The overall analysis presented here follows exactly the same procedure described in the main text, with the only differences being the higher halo mass cut of $10^{14}M_{\odot}$, required

due to the lower mass resolution compared to our smaller boxes, and the availability of only three redshift snapshots ($z = 0.0, 0.5, 1.0$).

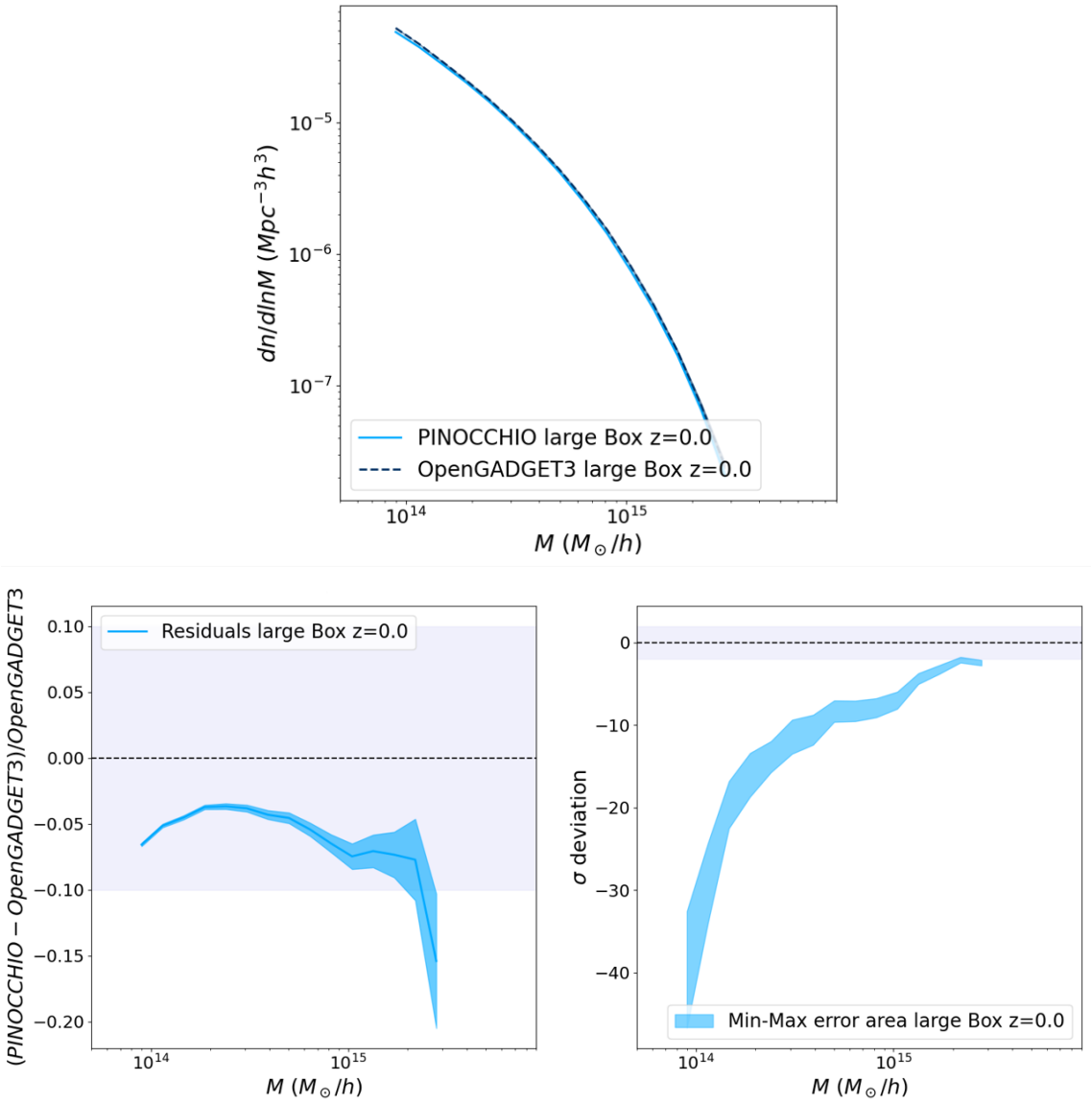


Figure 6.14: Same as Figure 6.1, but for the large box.

Figures 6.14 and 6.15 show, respectively, the same measurements as Figures 6.1 and 6.2, but for the larger box. These results confirm the clear mass dependence of the deviations already discussed in Section 6.4.2, with systematic discrepancies at the low-mass end where halo identification in PINOCCHIO is less reliable. Nonetheless, as in the main analysis, the agreement between the two methods remains within 10%, suggesting that we are limited by systematic effects in this regime. Despite the systematic deviations in the HMF, the void summary statistics remain consistent with the main analysis.

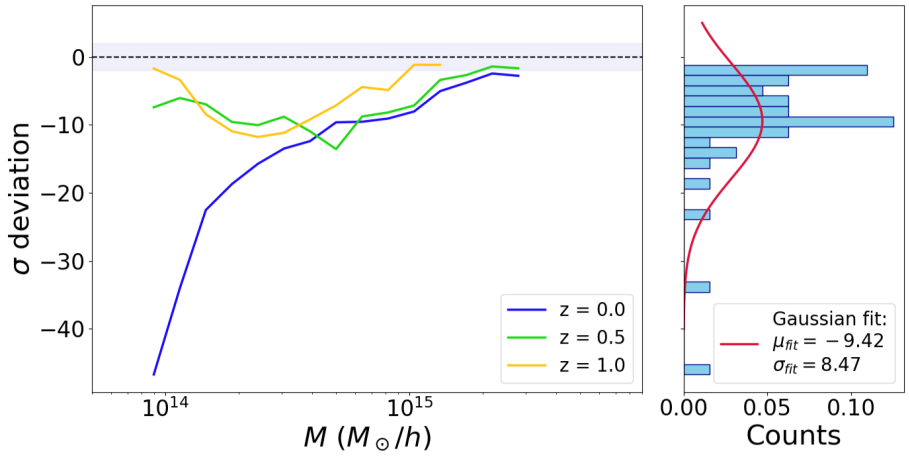


Figure 6.15: Same as Figure 6.2, but for the large box.

z	PINOCCHIO Large Box	OpenGADGET3 Large Box
0.0	22302	22506
0.5	10794	11131
1.0	3564	3617

Table 6.3: Number of voids identified by VIDE. The first column lists the redshifts z, while the second and third columns shows the number of voids identified for both simulation resolutions and codes.

The number of voids identified for each redshift z is reported in Table 6.3, showing comparable counts between the two codes, in agreement with the findings of the smaller-box runs. In addition, we have measured the VSF, VEFs, CDFs, and RDPs, together with the relative differences between PINOCCHIO and OpenGADGET3, normalized by the Jackknife errors from the OpenGADGET3 estimates at the three available redshifts.

The results for the VSF are shown in Figures 6.16 and 6.17, and confirm the same qualitative trends discussed in the main analysis. The same holds for the VEFs and RDPs, shown respectively in Figures 6.18, 6.19, 6.20 and 6.21.

The CDFs, presented in Figures 6.22 and 6.23, represent the only statistic where the agreement worsens, particularly at redshifts $z > 0$. However, this behavior is consistent with the discussion in the main text and in Section 6.6: the agreement in the CDFs improves with increasing resolution. Since the large-box runs have lower mass resolution than even our "low-resolution" simulations, the larger discrepancies found here are expected.

Overall, these results confirm the robustness of the analysis presented in Section 6.5, while also indicating that increasing the statistical sample of analyzed voids can significantly improve the precision of the measured summary statistics.

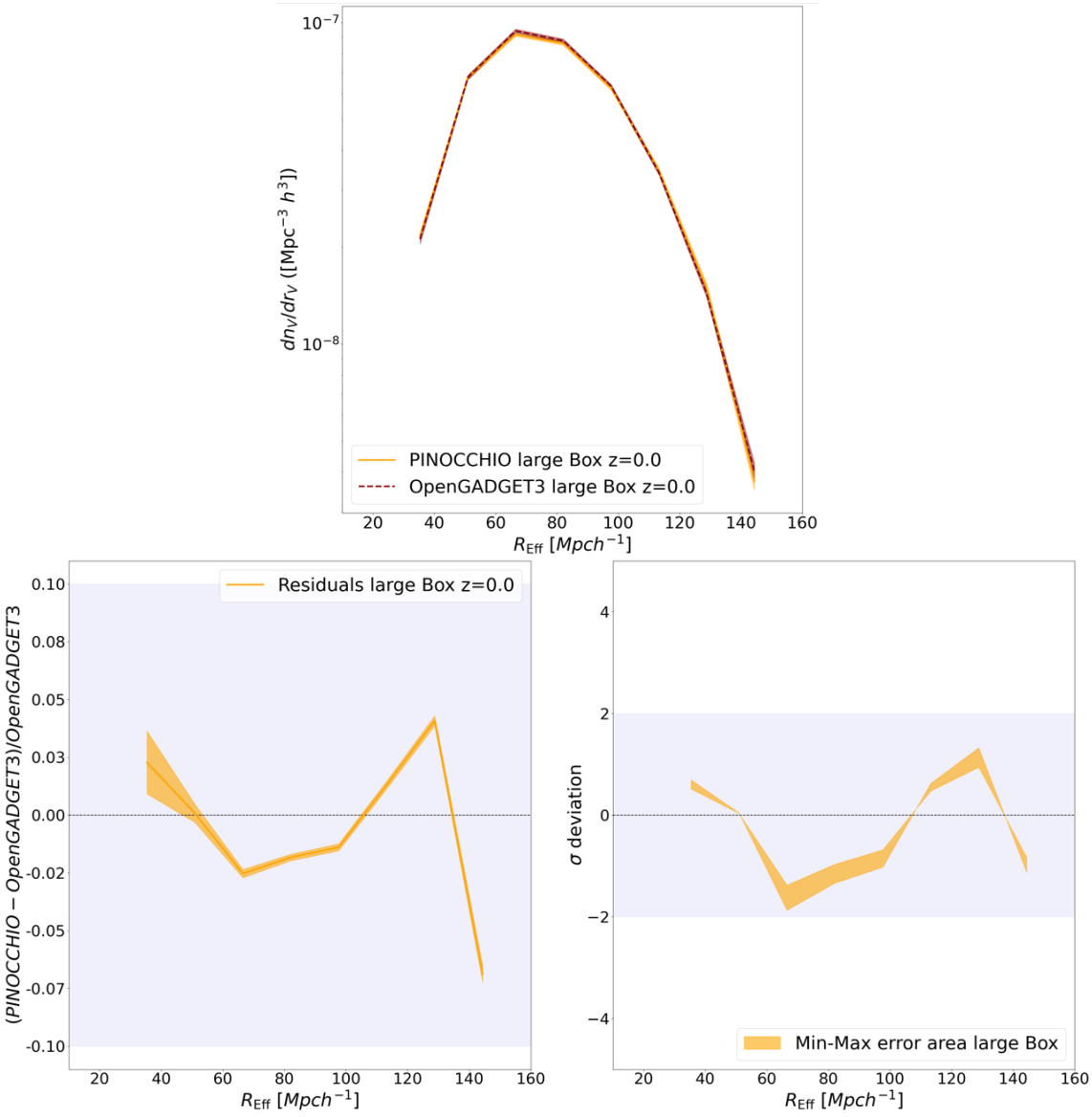


Figure 6.16: Same as Figure 6.3, but for the large box.

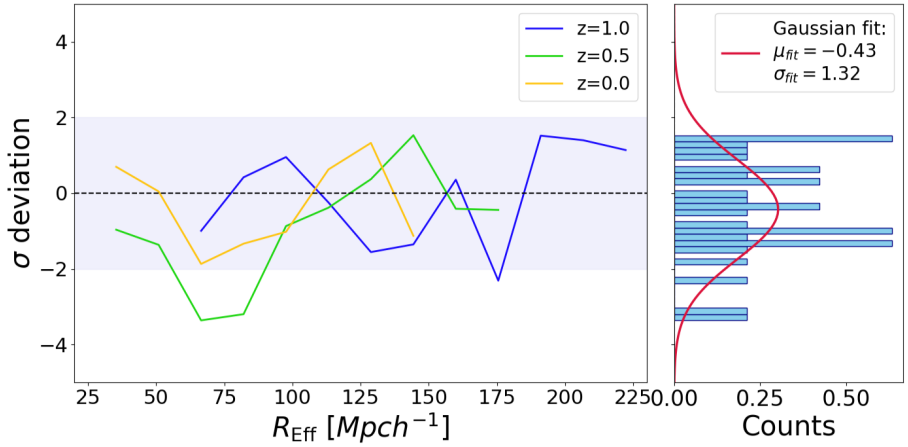


Figure 6.17: Same as Figure 6.4, but for the large box.

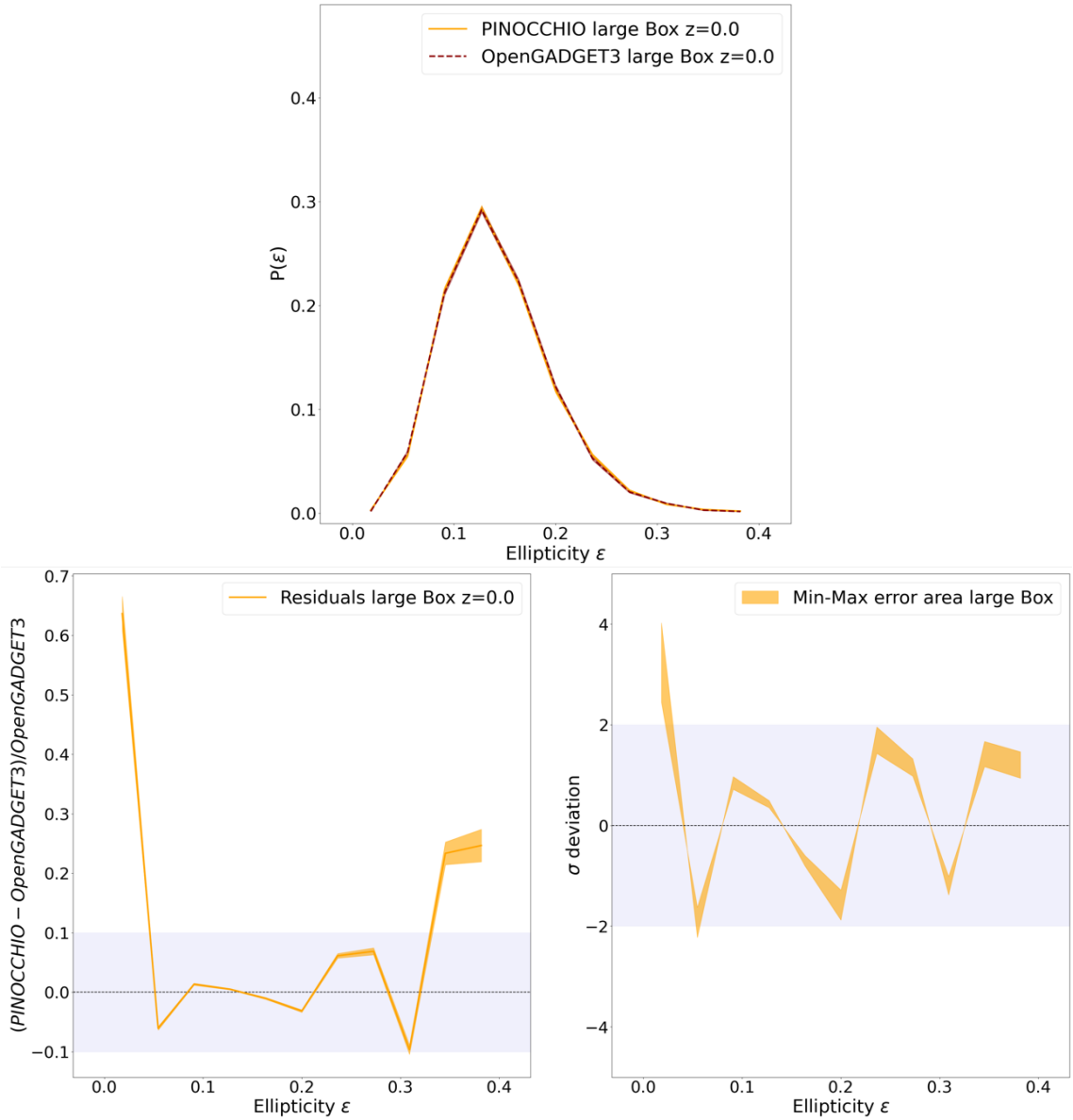


Figure 6.18: Same as Figure 6.5, but for the large box.

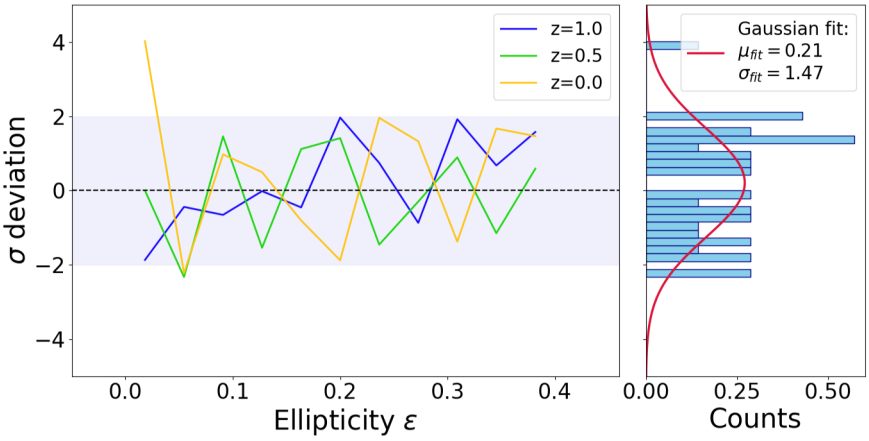


Figure 6.19: Same as Figure 6.6, but for the large box.

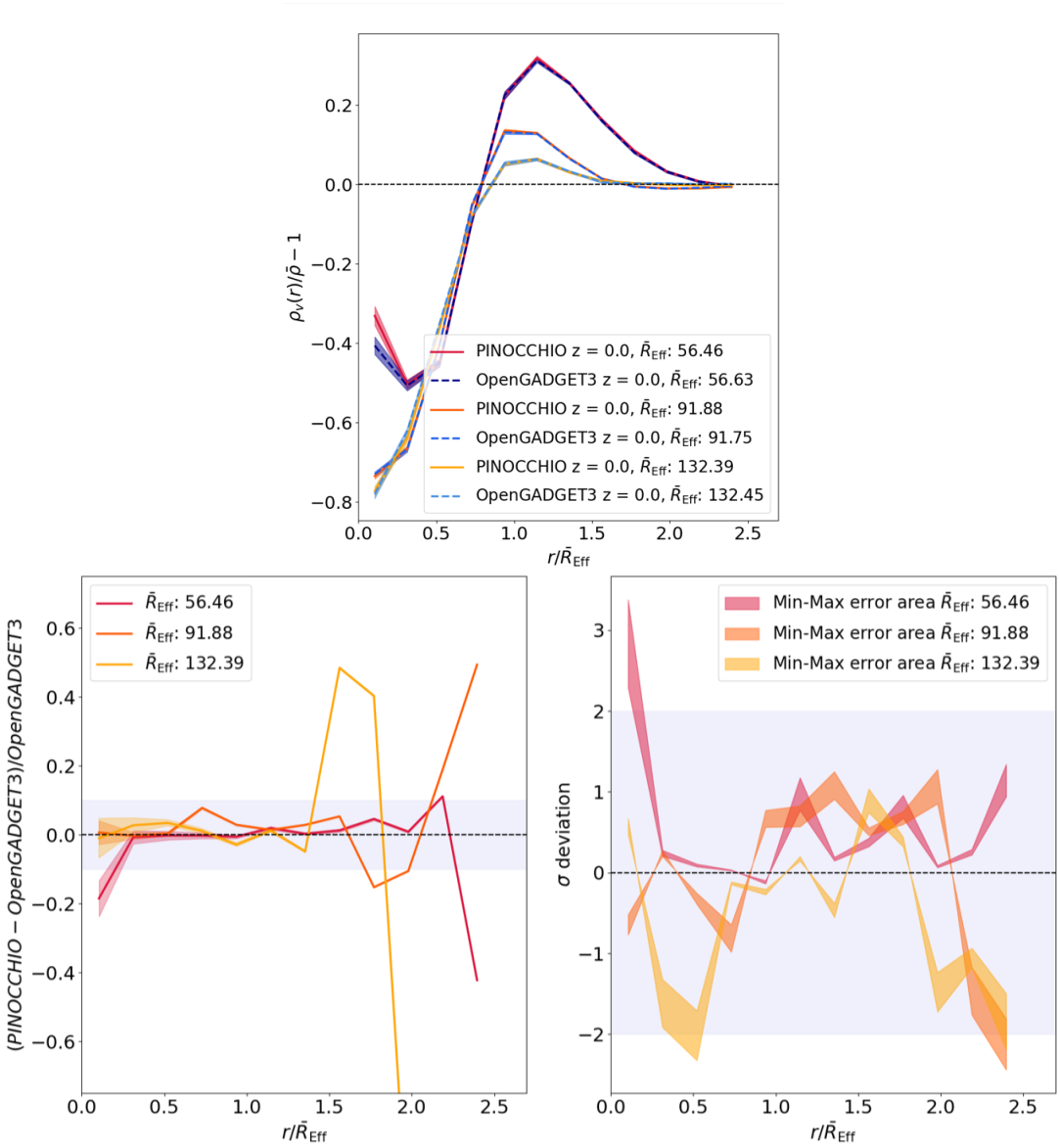


Figure 6.20: Same as Figure 6.9, but for the large box.

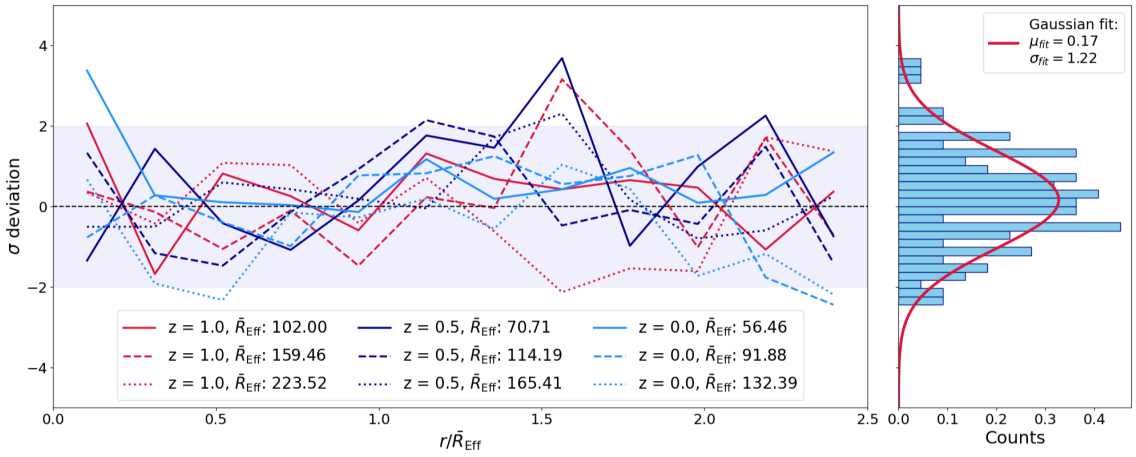


Figure 6.21: Same as Figure 6.11, but for the large box.

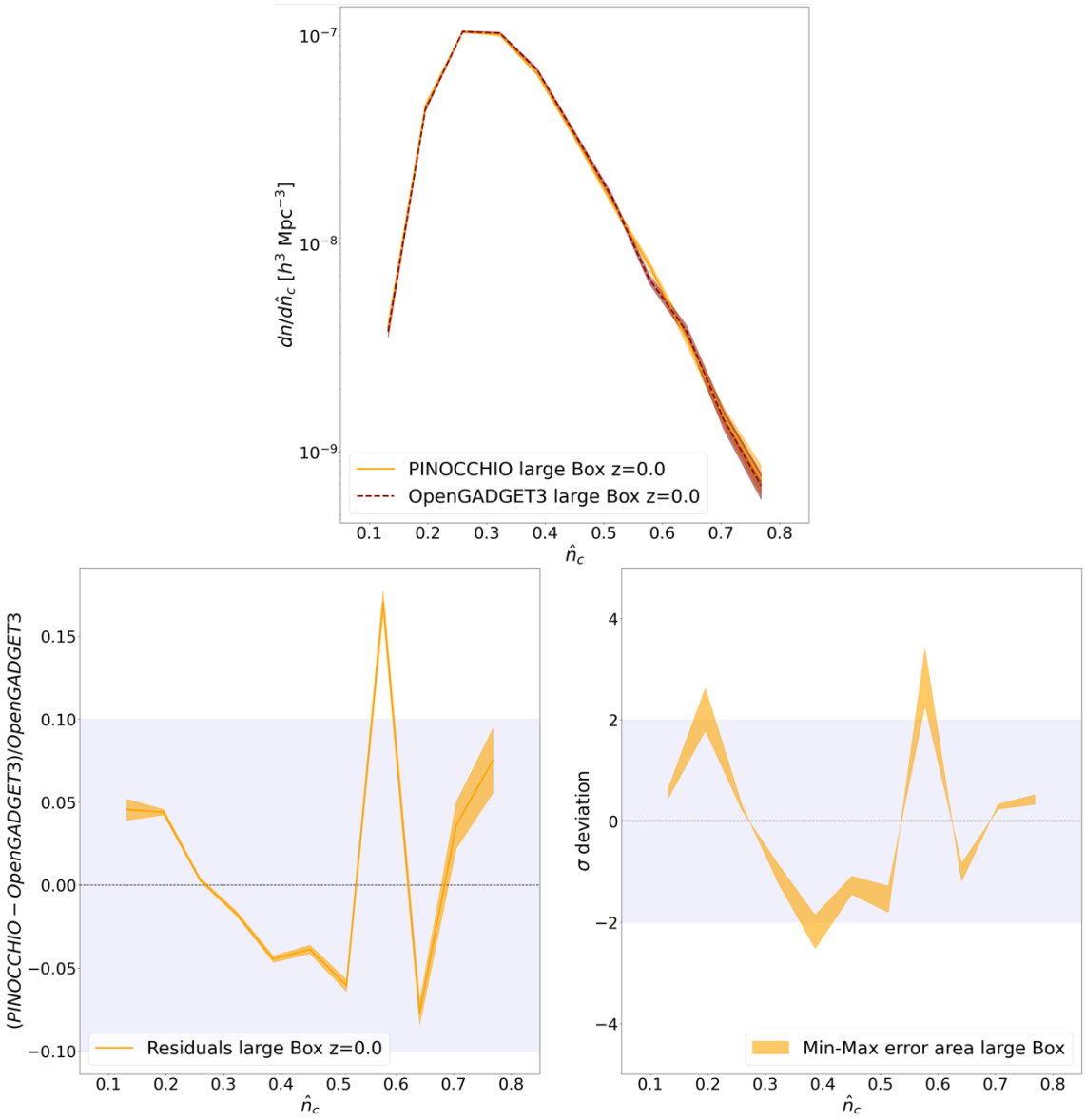


Figure 6.22: Same as Figure 6.7, but for the large box.

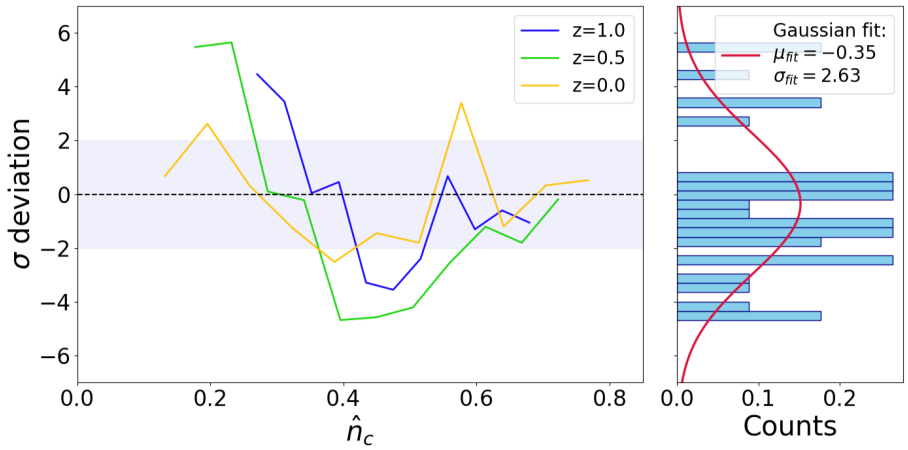


Figure 6.23: Same as Figure 6.8, but for the large box.

Chapter 7

Conclusions

In this thesis we developed and validated two complementary lines of research centered on the PINOCCHIO code: (i) the modernization, GPU porting, and energy-efficiency assessment of a key component of the code, aimed at enabling sustainable large-scale cosmological simulations for the exascale era; and (ii) the first comprehensive evaluation of the accuracy of PINOCCHIO in predicting cosmic-void statistics through a direct comparison with full N-body simulations. The first part addressed the computational challenges posed by next-generation cosmological surveys, focusing on performance portability, runtime reduction, and energy sustainability. The second part investigated the scientific accuracy of PINOCCHIO in underdense environments, assessing its suitability for void-based cosmology.

7.1 Conclusions on GPU porting and green performance of PINOCCHIO

In Chapter 3, we presented the porting and optimization of a specific segment within the PINOCCHIO cosmological simulation code to GPUs, using OpenMP target directives as a portable, directive-based programming model. Our effort focused specifically on the collapse time calculation, an embarrassingly parallel segment of the simulation pipeline ideally suited for GPU acceleration, and which in PINOCCHIO can be computed using either the *Classic* or the *Tabulated kernels*.

To ensure full GPU offloading and architectural portability, we developed custom GPU-native implementations of both cubic spline and bilinear interpolation routines, which constitute the computational core of the aforementioned *kernels*. These implementations successfully eliminated reliance on the GNU Scientific Library (GSL), which lacks GPU

support, and were written entirely with OpenMP pragma-based directives to ensure compatibility across both NVIDIA and AMD GPU architectures. Validation tests, using a standalone toy code, confirmed that the numerical results from the GPU-native interpolation routines maintain high fidelity, with residuals typically at the sub-percent level ($\sim 0.003\%$) compared to the original GSL-based CPU versions (Section 3.5.1 and Section 3.5.2). Performance benchmarks further demonstrated that our GPU implementations outperform their CPU counterparts once a sufficient number of evaluations points is reached, achieving up to $\sim 12\times$ speedup for both interpolation routines and showing significant scalability benefits in production-scale use cases.

Strong single-node scaling tests (see Section 3.4.3 for the single node resources configuration), of the offloaded *kernels* within the `PINOCCHIO` code, across both NVIDIA-LEONARDO and AMD (SETONIX) supercomputing platforms (Section 3.3) consistently showed GPU speedups exceeding $4\times$ relative to CPU-only implementations, with even larger gains observed for the *Tabulated kernel* due to the lack of OpenMP parallelization in the corresponding CPU baseline (Section 3.5.3). The consistent performance across the two architectures underlined the portability and generality of the OpenMP-based offloading approach.

A roofline analysis on the NVIDIA-LEONARDO platform, performed for the *Classic kernel* as the production-relevant case, revealed that the its GPU implementations achieves over 80% of the platform theoretical FP64 peak performance, operating in the compute-bound regime (Section 3.5.4). This confirmed not only the computational efficiency of the implementation but also the successful mitigation of thread divergence through a masked-based control strategy.

In a full production simulation setup, representative of typical large-scale simulation campaigns, the GPU-accelerated version of the *Classic kernel* led to a net reduction of approximately 100 seconds per run. When extrapolated to thousands of simulations, this would result in a saving of over 160000 Standard-h, representing a substantial gain in computational efficiency and resource optimization. At the same time, the scientific integrity of the *Classic kernel* was extensively validated: the distribution of collapsed particles as a function of redshift showed excellent agreement with the CPU reference, with residuals typically within $\pm 1\%$, and minor systematic deviations confined to specific redshift ranges . Furthermore, the resulting numerical HMF derived from the GPU run matched the CPU-derived counterpart well below the 1% level across the entire mass range (Section 3.5.5).

Beyond the computational savings, such reductions also implied a significant potential impact in terms of energy consumption. Building on the porting effort, in Chapter 4 we assessed also the energy efficiency of the GPU-accelerated version of `PINOCCHIO`,

with a dual focus on maximizing performance and minimizing energy consumption. To this end, we measured energy-to-solution alongside time-to-solution using a new parallel implementation of the PMT library (Section 4.2) across different HPC platforms equipped with AMD (SETONIX) and NVIDIA-KAROLINA GPUs (Section 4.1). Our primary goal was to address the growing need for computational sustainability in astrophysics, particularly in the context of producing large ensembles of mock halo catalogues for upcoming surveys such as *Euclid*.

On the AMD-based system, benchmarks (see Section 4.3.3 for the multi node resources configuration) showed an $\sim 8\times$ improvement in both time-to-solution and energy-to-solution relative to the CPU-only implementation (Sections 4.4.1 and 4.4.1), yielding an overall efficiency gain of $\sim 64\times$, as reflected in a substantially lower EDP (Figures 4.4 and 4.7). On the NVIDIA-based system, the improvements were more modest, with $\sim 2\times$ gains in both runtime and energy consumption, corresponding to a net efficiency increase of $\sim 4\times$ (Sections 4.4.2 and 4.4.2). The primary performance difference was mainly due to the disparity in the FP64 peak throughput between the AMD and NVIDIA architectures (Section 3.3). Specifically, the AMD GPU exhibits 47.9 TFlops peak FP64 performance (23.9 TFlops per GCD), significantly contrasting with the 9.7 TFlops demonstrated by the NVIDIA counterparts.

This gap was critically amplified by the unavoidable necessity of double-precision arithmetic in contemporary cosmological simulations, which leverages the $1:1$ single-to-double precision (FP32:FP64) ratio of the AMD accelerators, versus the $2:1$ ratio of the NVIDIA accelerators. The central objective of this study was to demonstrate cross-platform code portability across both device ecosystems. It is crucial to note, however, that code portability does not inherently ensure performance portability across different architectures, with the superior AMD performance constituting a key achievement of this investigation.

Nonetheless, any exhaustive performance analysis has to account account for the system-level architectural heterogeneity, including factors such as GPU/CPU count and the specific CPU-GPU interconnect topology, which impact overall efficiency and change the relative efficiency between one cluster and another one. The analysis of normalized `EDP`(Section 4.5) revealed which platform was best suited to run our *kernel* in CPU-only configurations and in GPUs, and we inferred that, depending on the configuration, one platform could be better than the other one and vice versa. Determining which platform is more efficient for a specific code is a non trivial task, and even for the same couple of architectures, it can differ depending on the specific configuration set up.

Taken together, these results confirmed that the GPU-accelerated version of `PINOCCHIO` was not only significantly faster than its CPU-based predecessor but also demonstrably

more energy-efficient.

Overall, GPU acceleration consistently reduced runtime and energy consumption, although the magnitude of the gains was system-dependent, thereby enhancing the sustainability of large-scale cosmological campaigns.

This achievement showcased a viable and effective pathway for modernizing legacy scientific codes to meet the demands of the exascale era. By leveraging the parallel processing power of GPUs, we enhanced the overall green productivity (GP) of cosmological simulations (Sections 4.4.1 and 4.4.2), enabling the generation of vast datasets required for next-generation surveys with a minimized environmental impact. This allowed us to tackle larger and more complex cosmological problems without a proportional increase in our energy footprint. The methodology and results presented here served as a compelling case study for the broader scientific community, highlighting the importance of adopting green-computing practices.

Determining the optimal execution environment for a given computational workload, particularly regarding the efficiency of GPU acceleration, is a non-trivial challenge. In this study, all performance benchmarks were conducted on GPU-equipped nodes, thereby incorporating the idle energy consumption of the accelerators into our analysis. This approach reflect a realistic scenario where GPUs, as integral components of the node, draw power regardless of their operational state. However, many supercomputing centers offer CPU-only partitions, presenting an alternative for workloads that may not benefit from GPU offloading. We propose that a systematic analysis of metrics such as the EDP and GP can serve as a valuable decision-making tool for HPC facility managers. By visualizing the trade-offs between performance and energy consumption, these metrics can guide the allocation of computational resources, especially in cases where the EDP values for CPU and GPU executions are comparable and the GP fails to exhibit strong scaling. Adopting such a quantitative approach to resource allocation can significantly mitigate energy waste, promoting a more sustainable and cost-effective operation of HPC infrastructure.

In conclusion, this study demonstrated the feasibility and effectiveness of accelerating scientific simulation kernels on modern GPU architectures using OpenMP directives. The combination of portability, computational efficiency, and scientific accuracy affirms OpenMP as a viable long-term strategy for modernizing legacy cosmological codes and adapting them to heterogeneous HPC environments.

Future work will involve extending these optimizations to other components of our simulation pipelines and exploring further algorithmic improvements to push the boundaries of sustainable computational cosmology.

7.2 Conclusions on tracing cosmic voids with PINOCCHIO

In Chapter 6, we assessed the accuracy of the PINOCCHIO code in reproducing void summary statistics by comparing it with the full N-body code OpenGADGET3. To ensure a fair comparison of the void properties, we used identical ICs and constructed halo catalogs matched in number density, thereby ensuring statistically comparable tracer distributions.

By applying the watershed void finder VIDE to the halo catalogs, we investigated four key void summary statistics: the void size function (VSF), void ellipticity function (VEF), core density function (CDF), and radial density profiles (RDP). Across all metrics, PINOCCHIO demonstrated very good agreement with OpenGADGET3 within statistical uncertainties, with no evidence of systematic biases. In particular:

- VSF: PINOCCHIO reproduces the void abundance as a function of size across redshift and resolution (Figures 6.3 and 6.4), with a level of agreement comparable to that achieved for HMF in overdense regions (Figures 6.1 and 6.2). Differences at large void sizes are attributed to resolution-dependent halo clustering effects.

- VEF: the ellipticity distributions show very good consistency across multiple redshifts, suggesting that PINOCCHIO captures the large-scale tidal influence on void shapes (Figures 6.5 and 6.6).

- CDF: PINOCCHIO performs particularly well at low core densities, where its LPT-based approach is expected to be most accurate (Figures 6.7 and 6.8). Deviations at higher core densities are mild and resolution-dependent, likely related to Fragmentation and small-halo statistics.

- RDP: the stacked radial density profiles agree well across all redshifts, reinforcing PINOCCHIO's reliability in reproducing the structure, environment, and evolution of voids (Figures 6.9, 6.10, and 6.11).

Across all these statistics, the differences between PINOCCHIO and OpenGADGET3 results remain mostly within $\pm 10\%$ variation range and below the 2σ deviation level. Because the two simulations share the same ICs, their statistical uncertainties are correlated; although a full covariance estimate is beyond scope, we bracketed the significance with conservative estimators that provide upper and lower limits on the true level of statistical disagreement. Additional tests on a larger box, with an order-of-magnitude more voids, confirmed consistent results (Section 6.7), showing that our conclusions are robust against sample variance and finite-box effects.

It is nevertheless important to interpret these results with care. Since the two simulations share identical ICs, any persistent offset between PINOCCHIO and OpenGADGET3 reflects a

modelling systematic rather than independent statistical scatter. Increasing the simulation volume reduces the statistical uncertainty of the measurements but does not remove such an intrinsic offset; rather, it allows it to be quantified more precisely. For this reason, the physically relevant quantity is the amplitude and scale dependence of the residual systematic difference, rather than its formal significance in units of the finite-volume statistical error (σ). The large-box test confirms this behavior: while measurement noise decreases with increasing volume, the relative level of discrepancy remains stable, supporting its interpretation as a solver-dependent systematic rather than a noise-driven fluctuation.

The observed 10% level of variation is consistent with previous validation studies of PINOCCHIO at the level of halo catalogs, where the HMF and large-scale clustering have been shown to agree with full N-body simulations at the few–10% level depending on scale and selection. Since cosmic voids are identified from the underlying halo distribution in this work, their statistical properties are directly linked to the accuracy with which the tracer field is reproduced. While void identification introduces additional non-linear processing through the watershed algorithm, the level of discrepancy measured here remains consistent with the established performance of the approximate gravity solver on large-scale structure statistics.

In the context of precision surveys such as Euclid and DESI, it is essential to distinguish between modelling of the mean observable and estimation of its covariance. Percent-level accuracy is typically required for forward modelling of the mean signal to avoid parameter biases, which generally necessitates calibrated or full N-body simulations. By contrast, covariance estimation primarily requires the generation of large ensembles of realizations that reproduce the large-scale clustering and the associated bin-to-bin correlations, so that sampling noise can be suppressed. In this respect, PINOCCHIO provides a computationally efficient framework suitable for uncertainty quantification, even if a residual systematic offset in the mean void observable must be calibrated separately.

More recently, hybrid approaches combining a limited number of expensive N-body simulations with a large ensemble of fast approximate realizations, such as control-variate techniques CARPool (Chartier et al. 2021), have demonstrated that the strong correlation between cheap and costly simulations can be exploited to significantly reduce statistical errors in mean predictions. In this framework, approximate solvers like PINOCCHIO can contribute not only to covariance estimation but also to improved modelling of the mean observable, while keeping the number of required N-body simulations manageable. The present results therefore support the use of PINOCCHIO-based catalogs both for covariance construction and as a component of hybrid variance-reduction strategies, while full N-body simulations remain necessary to anchor precision modelling.

Overall these results demonstrate that PINOCCHIO reproduces void summary statistics with good accuracy, particularly in the underdense regime where linear and quasi-linear dynamics dominate. This makes it a promising tool for cosmological applications involving voids, especially in the context of upcoming large-volume surveys such as Euclid and LSST, where computational efficiency and statistical robustness are crucial. A natural valuable next step would be to test the performance of PINOCCHIO in a lightcone geometry, incorporating observational systematics, to assess its direct applicability to survey data.

These results can also be interpreted as quantifying the bias that PINOCCHIO introduces in void statistics. Once characterized, such a bias can be incorporated into data analyses to recover unbiased constraints. Furthermore, a $\sim$10% uncertainty in the statistic is not necessarily limiting for parameter inference, since the steepness of both the HMF and VSF often makes such effects subdominant. The impact may also depend on tracer density, with small voids in dense surveys (e.g. DESI) being harder to model, while larger voids should remain robustly captured. Moreover such biases are not expected to vary significantly with cosmology within ΛCDM, and previous works (Munari et al. 2017) have shown that the calibration of PINOCCHIO is cosmology-independent. Applications to modified gravity would require re-calibration and are left for future work.

Since PINOCCHIO is already widely used for generating large ensembles of simulations to estimate and validate covariance matrices, a natural extension of this work would be to investigate whether PINOCCHIO can reliably reproduce the covariances of void statistics. The good agreement found here suggests this may be the case, as already demonstrated for halo statistics. Validating the precision of void covariance estimation would be an important step toward their use in high-precision cosmological inference from observational data, while taking full advantage of PINOCCHIO's computational efficiency.

More broadly, the accuracy of PINOCCHIO in reproducing void statistics, particularly in underdense regimes, makes it well suited for survey forecasts and theoretical modeling, including tests of modified gravity and dark energy. Future work will explore its performance in non-standard cosmologies and its direct application to cosmological inference from void statistics.

Last but not least, an important direction for future work concerns a direct, one-to-one comparison of individual voids identified in PINOCCHIO and N-body simulations. While the present study focuses on ensemble summary statistics, an object-level analysis could provide complementary insight into possible morphological differences that may be partially averaged out in statistical measures. Such an investigation would require the definition of a robust void-matching criterion, for example based on spatial overlap or shared tracer populations, and would build naturally upon existing halo-matching methodologies. Exploring this aspect would further clarify how solver-level differences

propagate into higher-order void properties and strengthen the validation of approximate methods for precision cosmology.

La borsa di dottorato è cofinanziata con risorse dell'Unione europea, NextGeneration EU - Piano Nazionale di Ripresa e Resilienza, Missione 4 – Componente 2 – Investimento 1.4 CUP C53C22000350006

Appendix A

ADP 2D: Deblending

This Appendix presents ongoing and exploratory developments related to the application of the Adaptive Density Peak (`ADP`) clustering algorithm to the problem of source deblending in astronomical imaging. While not directly part of the main `PINOCCHIO` simulation workflow, this work addresses a closely related computational and methodological challenge arising in the context of modern wide-field astronomical surveys, where accurate and scalable image segmentation is essential for reliable catalog production.

The method described here represents the two-dimensional counterpart of the new `Fragmentation` strategy introduced in Chapter 5 for the `PINOCCHIO` code, extending the same clustering-based principles to image-based source segmentation. This Appendix is organized as follows. We begin by introducing the scientific and computational challenges associated with source deblending in astronomical images. We then describe the `ADP` algorithm and the modifications introduced for image-based source segmentation, together with the high-performance computing considerations that motivate its distributed-memory extension. We subsequently present a modular validation pipeline based on controlled simulations and physically motivated ground-truth segmentation maps, along with experimental results obtained on simulated datasets, while initial tests on real *Euclid* data are currently ongoing.

While the current results are encouraging, a comprehensive characterization of performance and scientific impact is still in progress, and further validation is required to fully assess the applicability of the approach in production pipelines. The material collected here outlines a promising direction for future developments in scalable deblending methodologies and complements the main thesis by extending its high-performance computing perspective to imaging data analysis

A.1 Deblending in modern astronomical surveys

The accurate identification and separation of astronomical sources from imaging data, commonly referred to as source deblending, is a fundamental task in modern observational astronomy. With the advent of wide-field surveys and high-resolution instruments such as *Euclid* (Mellier et al. 2024), Large Synoptic Survey Telescope (LSST) (Ivezić et al. 2019) and James Webb Space Telescope (JWST) (McElwain et al. 2023), the density and level of detail of objects captured in a single image has dramatically increased, often resulting in overlapping or closely spaced sources. For instance, the public *Euclid* Quick Data Release (Q1) (Aussel et al. 2025) already contains over 30 million sources across just ~ 63 deg^2 of sky, and by the end of the mission the survey is expected to cover roughly 14000 deg^2. This blending of sources poses a significant challenge for downstream tasks such as photometry, morphological classification, and object cataloging (Bosch et al. 2018; Romelli et al. 2025).

To address this, various deblending algorithms have been developed, ranging from classical threshold-based techniques of image segmentation (`SourceXtractor++`; Bertin et al. 2020; Kümmel et al. 2022), to model-fitting methods (`SCARLET`; Melchior et al. 2018), to density-based approaches (`ASTErIsM`; Tramacere et al. 2016), and to topological methods based on persistent homology (`DRUID`; Shaw et al. 2025). Each approach trades accuracy, robustness, and computational cost differently: thresholding is fast but struggles in crowded fields; model fitting can be precise but is computationally heavy and sensitive to initial conditions; density and topological approaches are more agnostic to morphology but often require careful scale choices.

Outside astronomy, neural network-based methods nowadays represent the state of the art. Models such `U-Net` (Ronneberger et al. 2015) and its variants have been proven to handle successfully segmentation tasks with complex decision boundaries. However, their use has so far been constrained to relatively small image patches ($\approx 128\times128$ pixels). More complex models like `Mask R-CNN` (Burke et al. 2019) can overcome patch-size limitations, but at the expense of substantial computational cost in both training and inference, and without guaranteeing the transferability of model accuracy on data coming from different sources.

Despite these advances, two key challenges remain: (i) the absence of systematic benchmarks that stress-test methods under controlled simulations, for example by varying signal-to-noise ration (SNR) and source separation, and (ii) insufficient throughput to process the tens of millions of sources expected from surveys such as *Euclid* efficiently, where reducing deblending run-times from hours to minutes can free substantial resources for the downstream steps of the analysis pipeline.

In this Appendix, we present a novel adaptation of the Adaptive Density Peak (ADP) clustering algorithm (d'Errico et al. 2021) for image-based source segmentation in astronomy. Our method operates directly on pixel-level flux and requires no assumptions about source morphology. To ensure survey-scale efficiency and avoid deblending becoming a bottleneck in large data-processing pipelines, we also present as a future development a distributed-memory implementation of the algorithm called DADP, which will enable scalability to the data volumes of current and upcoming wide-field surveys.

To rigorously test ADP we have also developed a modular validation pipeline that combines controlled simulations of both point-like and extended sources, physically motivated ground-truth segmentation maps (Haigh et al. 2021), and a comprehensive set of evaluation metrics. These metrics capture not only pixel-level accuracy but also the tendency of ADP to over- or under-deblending for a given combination of simulation parameters. This framework enables reproducible stress-testing of ADP and highlights its strengths and limitations under conditions that are rarely explored in existing studies.

A.2 ADP: classic and deblending implementation

The deblending problem can be formally cast as an image segmentation task, where the goal is to identify regions of an astronomical image with significant luminosity above the background and group pixels around the local maxima of the luminosity field. This formulation aligns closely with the principles of density-based clustering, in which clusters are defined as regions surrounding local density peaks within a point cloud.

Several astronomical applications have already leveraged this connection. For example ASTErIsM (Tramacere et al. 2016) employs two well-known density-based techniques, namely DBSCAN (Ester et al. 1996) and DENCLUE (Hinneburg & Keim 1998; Hinneburg & Gabriel 2007), to detect and deblend sources on astronomical images (Romelli et al. 2025). In the realm of density-based clustering methods, ADP stands out as one of the most recent developments, combining the density-based framework with hierarchical structures detection. These features make it a compelling candidate for addressing the deblending problem.

In its original formulation, the ADP algorithm requires as input a dataset $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ where $\mathbf{x_i} \in \mathbf{R}^n$. For each data point, a pre-calculated estimate of the log-density value $\log(\rho_i)$ and its associated error ε_i are required, together with the list of its *k-nearest* neighbors. The algorithm then operates in three main steps:

1. **Cluster Center Identification**: This step identifies cluster centers by searching for local maxima of the density field. Once the centers are determined, each data point

is assigned to the cluster of its nearest neighbor with higher density.

2. **Saddle Point Identification**: This step identifies the saddle points in the density field, which serve as border points between clusters. A saddle point, call it i, between clusters c and c' is the data point that has the following properties: a) has a point j in its k-nearest neighborhood belonging to cluster c' and i is the nearest neighbor of j belonging to c, and b) i is the point with higher density belonging to c or c' for which property a holds.

3. **Density Peak Validation and Hierarchical Merging**: In this final step, the algorithm validates potential clusters. The density at the center of each cluster ($\log(\rho_c)$) is compared with the density at the border with each neighboring cluster ($\log(\rho_{cc'})$). The two are merged if the condition:

$$\log(\rho_c) - \log(\rho_{cc'}) < Z(\varepsilon_c + \varepsilon_{cc'})$$

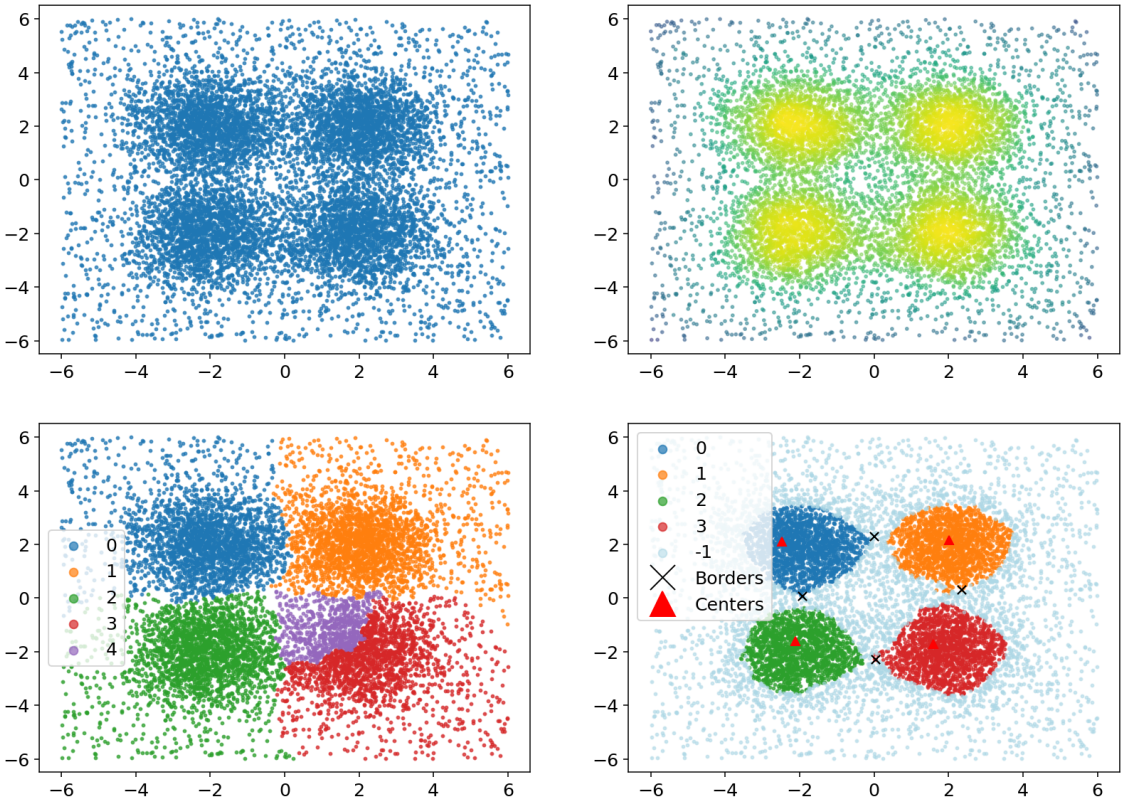


Figure A.1: Example of `ADP` applied to a synthetic dataset. The upper left panel shows a mixture of four Gaussians distributions with additional uniform noise. The upper right panel presents the corresponding density map of the dataset. The lower left panel depicts the initial clustering before the density peak merging validation step. The lower right panel displays the final clusters obtained after the merging procedure.

is met. This comparison helps to determine whether a cluster is a true density peak or merely the result of a statistical fluctuation. The value of Z controls the merging process; a larger Z value leads to more aggressive merging. The complete procedure is described in detail in d'Errico et al. (2021).

Figure A.1 reports a pictorial representation of the ADP procedure. To adapt `ADP` for the deblending use case, two primary modifications were introduced: i) individual data points are replaced with pixels from the image, so that the *k-nearest* neighbors correspond now to neighboring pixels within a specified radius; ii) density values are computed as the average luminosity of these neighboring pixels, with the associated error estimated as the standard deviation.

A.3 HPC for deblending of large images

The search for a suitable candidate for deblending has been accompanied by a significant effort to develop an efficient implementation optimized for High Performance Computing (HPC) infrastructures. The current implementation is designed as a compact C program that executes the `ADP` algorithm. It relies on the OpenMP (Dagum & Menon 1998) library to leverage CPU parallelism whenever feasible. The code is compiled into a shared object that can be used either from an executable or encapsulated into a Python module that binds to the compiled code. The latter option allows for both ease of use and performance.

Although `ADP` proves effective for clustering, its computational characteristics presents a mixed computational profile for parallelization. Parallel processing can be exploited in the preliminary phase, both in the initial conversion of luminosity values into density estimates manageable by the algorithm, and the identification of candidate cluster centers via the first heuristic step, where each pixel of the image can be processed independently. The same holds for the identification of saddle points in the second step of ADP. However, the overall performance is ultimately constrained by inherently serial components. Specifically, the validation of peaks (or cluster centers) in the third procedural step requires sequential execution. Since the validation of any given peak often depends on the finalized status or metrics of previously validated peaks, this dependency creates a bottleneck, limiting the extent to which `ADP` can benefit from parallel computing architectures. Despite this, the processing in this step involves a number of elements that is significantly smaller than the total number of points (or pixels, in the context of images); therefore, this potential bottleneck is manageable.

The current codebase for image segmentation is optimized for execution on CPUs within shared-memory environments through a patch-level parallelization strategy. In the standard

Euclid analysis pipeline, source detection (Romelli et al. 2025) is first performed using `SourceXtractor++`, which identifies a set of independent image patches corresponding to individual detected objects or small groups of overlapping sources above a given background threshold. These patches can be processed independently and therefore provide a natural level of parallelism for the deblending stage.

In this workflow, each detected patch is assigned to a single CPU core, and multiple independent instances of the `ADP` are executed concurrently, one per patch. This embarrassingly parallel strategy enables efficient utilization of multi-core shared-memory systems and significantly reduces the overall time-to-solution for large images. We are currently developing an adaptive memory-management strategy to further improve robustness. While in most cases a single core can comfortably handle the memory requirements of an individual patch, a small fraction of highly crowded or extended regions may exceed the available per-core memory. Ongoing work focuses on dynamically detecting such cases and adapting the execution strategy accordingly to ensure both correctness and scalability.

The computational efficiency of this `ADP` implementation has been tested on selected images from the *Euclid* Q1 public release, with sizes up to 19200×19200 pixels. In these test cases, the implementation achieved an approximately tenfold speedup compared to the current standard, the `ASTErIsM` deblender, completing the task in about 3 minutes compared to ASTErIsM's ~50 minutes. As an illustrative example, Figure A.2 shows a 500×500 cutout from the *Euclid* Q1 data release alongside the corresponding segmentation maps produced by `ADP` and `ASTErIsM`, while Figure A.3 shows a larger cutout of the same region. In the latter case, only the image cutout and the `ADP` segmentation map are shown. While a comprehensive quantitative validation is still ongoing, these examples provides a visual impression of the segmentation output.

To enhance performance and versatility, future development should focus on two key areas. First, leveraging the OpenMP library offers a direct path to exploring performance offloading to specialized hardware accelerators, in particular GPUs. With GPU support introduced in version 4.0, OpenMP (Martineau et al. 2016) provides a directive-based programming model that facilitates computation offloading while maintaining cross-platform portability and preserving code readability (Lepinzan et al. 2025). Second, extending the code capability to distributed-memory environments. This transition would enable the processing of much larger images, with significant memory footprints, by harnessing the collective computational power of multiple nodes.

Following this latter direction, a distributed-memory implementation of the `ADP`, referred to as `DADP`, has already been developed for its original clustering formulation as in d'Errico et al. (2021). This extension specifically targets tabular datasets, such as those arising in cosmological simulations, where `ADP` serves as a substructure finder. The primary

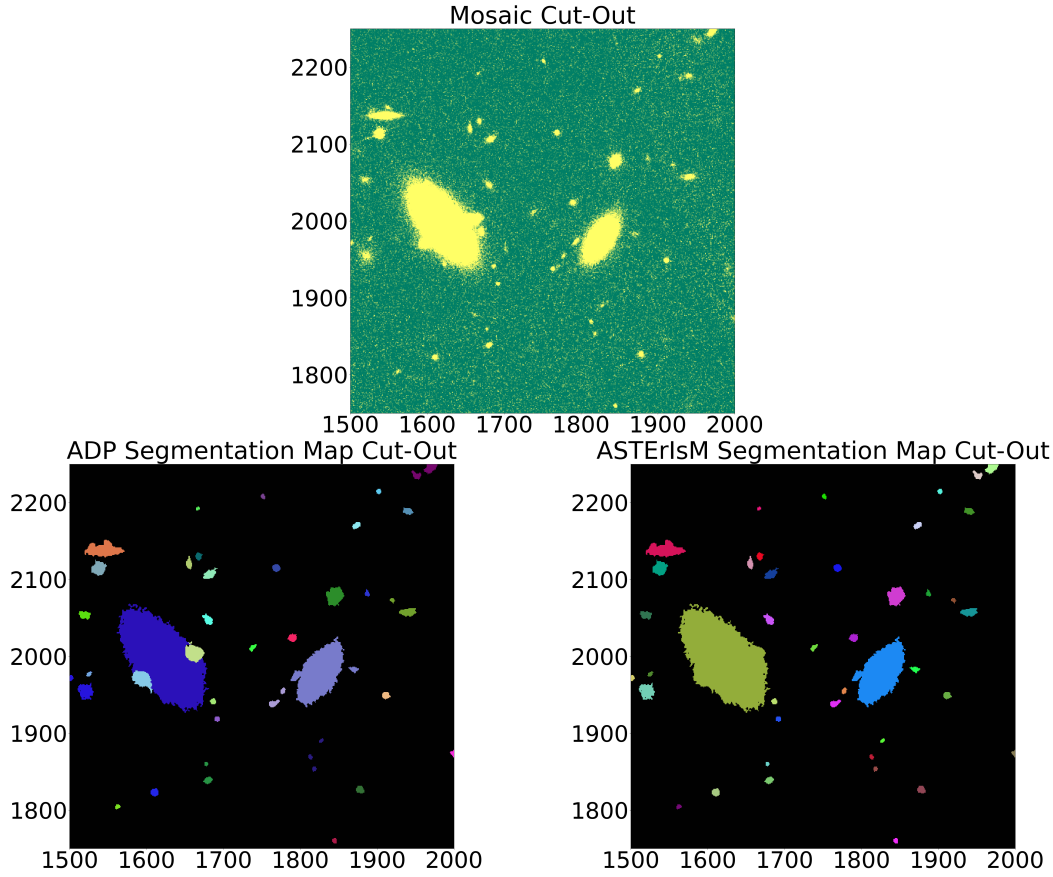


Figure A.2: Comparison of image segmentation results. The upper panel shows a 500×500 cutout from the *Euclid* Q1 data release. The lower-left panel presents the segmentation map of the same region obtained with `ADP`, while the lower-right panel shows the result produced by `ASTErIsM`. In both maps, the identified sources are color-coded. Although this is only a single example, the comparison highlights the broadly similar segmentation performance achieved by the two codes.

objective of this effort was to enable the algorithm to process datasets containing billions of data points.

To achieve scalable performance, `DADP` employs a distributed-memory implementation of a kd-tree for both domain decomposition and the *k-nearest* neighbors search. Within the `ADP` core procedure, efficiency is further enhanced through One-Sided Remote Memory Access (RMA) provided by the MPI framework (Message Passing Interface Forum 2023). This feature enables data points to directly exchange information, such as density values and cluster assignments. Unlike the traditional Send/Receive scheme, one-sided access is more suitable in this context, as information from remote tasks is mostly accessed in read mode and does not require the simultaneous coordination of sender and receiver processes. A detailed paper on this distributed architectural approach is currently in preparation (Tomba et al. in prep.).

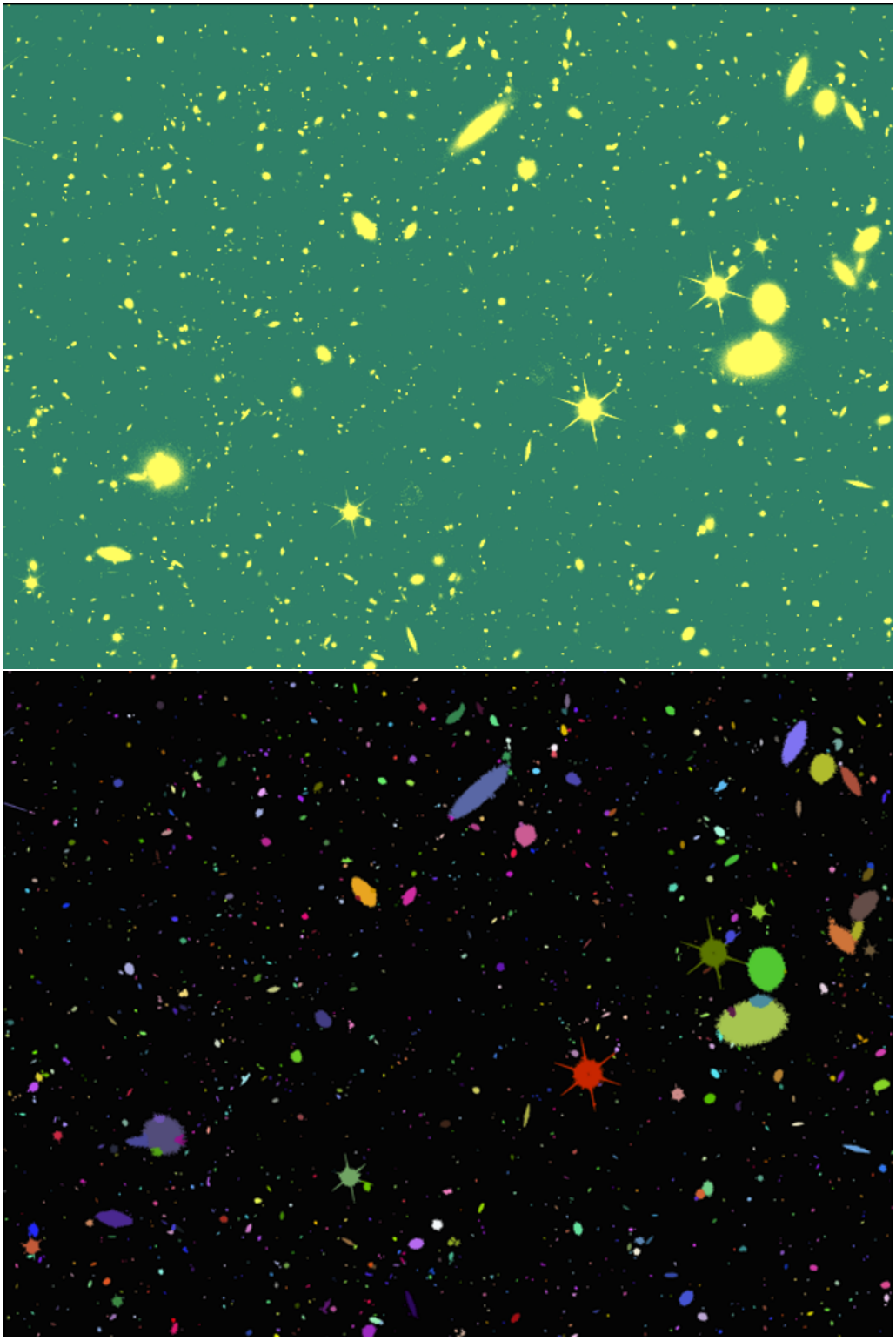


Figure A.3: Similar to Figure A.2, but showing a larger area. The upper panel displays the image cutout, while the lower panel shows the corresponding `ADP` segmentation map.

Looking ahead, the deblending application represents a natural next key target for performance scaling, with future work focusing on the development of a distributed-

memory version of the `ADP` deblender. This effort will directly leverage optimizations and other solutions that have been developed for the case of tabular datasets in distributed memory environments.

A.4 Validation of deblending alghorithm

A viable approach for validating a deblending procedure would typically involve comparing the resulting segmentation with an established *ground truth* reference. However, a critical challenge arises in the context of astronomical images: a definitive ground truth never exists. Across different astronomical surveys, diverse methodologies have been employed to generate segmentation maps, each introducing its own assumptions and biases. As a result, direct comparisons with these maps may lead to misleading conclusions and do not guarantee a fair or objective evaluation of deblending performance.

To address this, we developed a robust and modular validation pipeline tailored for astronomical deblending. This pipeline includes:

- A fully controllable simulation engine, generating synthetic astronomical images with both point-like (stars) and extended sources (galaxies). Sources are modeled using parametric profiles: 2D Gaussian for stars and Sérsic profiles for galaxies.
- Flexible control over simulation parameters such as flux ratio, source separation, and background level (with additional morphological parameters optionally available for testing).
- Ground truth segmentation maps constructed using an importance-based pixel assignment strategy following Haigh et al. (2021).

Each source is simulated independently, and the final image is constructed by summing its individual contributions together with a random Poisson background. For every image, a ground-truth segmentation map is generated by assigning pixels according to a combined measure of pixel significance to the source and source contribution to the pixel. This approach ensures that overlapping and blended sources are appropriately labeled, preserving fainter components while preventing domination by brighter neighbors.

A second central challenge in deblending validation lies in the choice of evaluation metrics. Rather than relying on label-sensitive metrics like the pixel-wise confusion matrix or Intersection over Union (IoU), which may vary significantly due to label permutations, we adopt *pairwise precision* and *pairwise recall*, two clustering-based metrics that provide a more robust characterization of deblending performance. These metrics are:

- Label-invariant, meaning they do not depend on how individual segment labels are assigned.
- Insensitive to over/under-labeling, offering a clearer measure of over- and under-deblending.

Formally, let:

- True Positives (TP): pairs of pixels that belong to the same source in both the prediction and the ground truth.
- False Positives (FP): pairs grouped together in the prediction but not in the ground truth.
- False Negatives (FN): pairs grouped together in the ground truth but missed in the prediction.

Then:

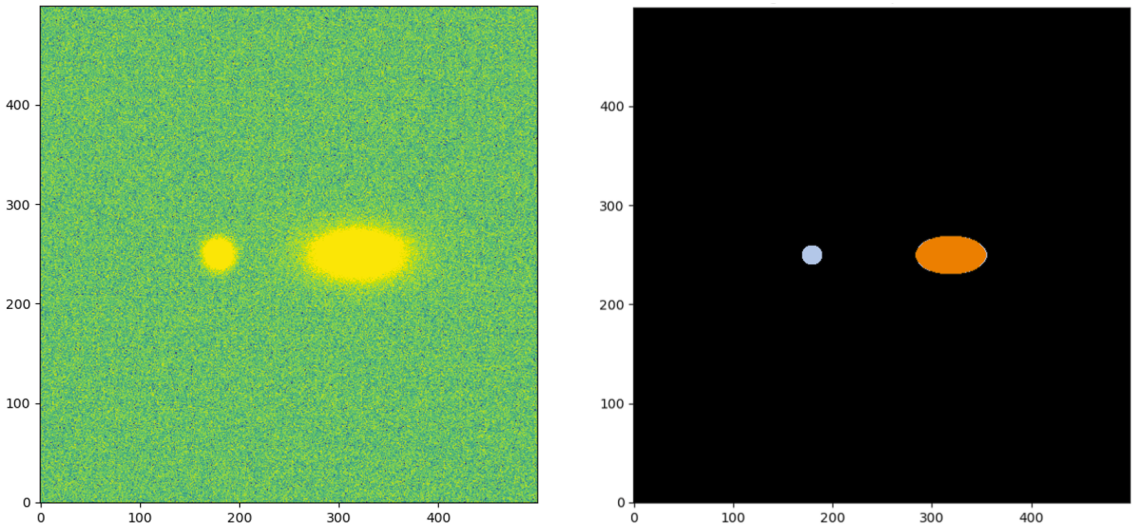
- Pairwise Precision = TP / (TP + FP)
- Pairwise Recall = TP / (TP + FN)

Together, these metrics provide a robust, high-level understanding of the deblending behavior: low precision (or equivalently, high 1 - precision) indicates under-deblending, where distinct sources are erroneously merged, while low recall (high 1 - recall) indicates over-deblending, where a single source is incorrectly split into multiple components. Beyond strengthening the evaluation of `ADP`, this methodology lays the foundation for a more general benchmarking framework for astronomical deblending. It highlights the importance of standardized datasets and quantitative, label-invariant metrics to complement, and ultimately move beyond, qualitative visual inspection.

The validation pipeline is exercised on a progressively more complex set of test scenarios, including:

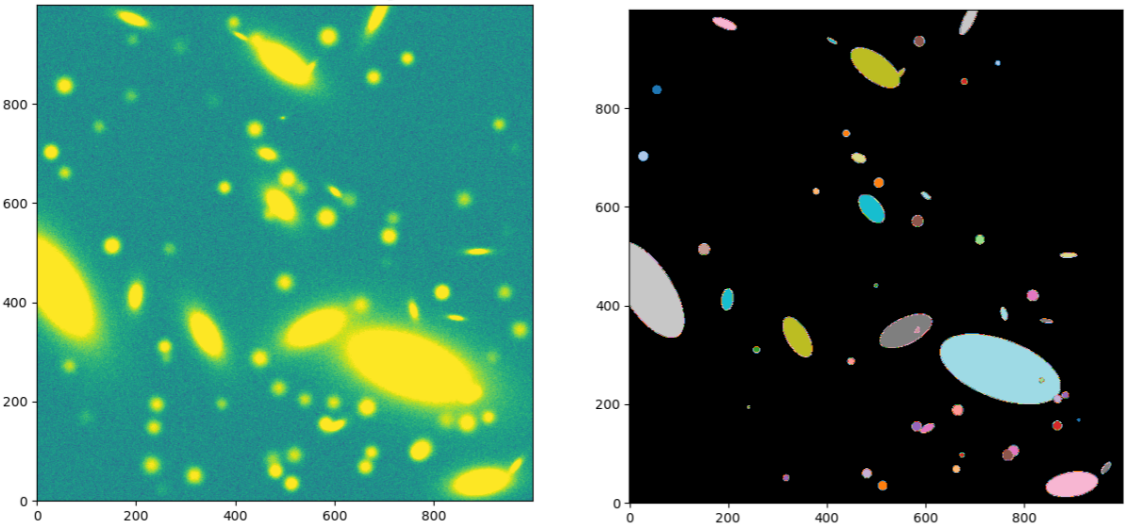
- Two point-like sources, varying both flux ratio and separation.
- Two extended sources (galaxies), varying flux ratio and separation.
- A mixed pair consisting of one point-like source and one galaxy, varying flux ratio and separation.
- A crowded field containing approximately 100 sources.

This set of controlled experiments enables us to probe the behavior of the algorithm under both idealized conditions and more realistic, crowded environments. Examples of simulated images together with their corresponding importance-based ground-truth segmentation maps are shown in Figures A.4 and A.5. These illustrate the cases of a star–galaxy pair and a crowded field, respectively. Similar behavior is observed for the star–star and galaxy–galaxy pair configurations.



(a) Simulated image **(b)** Importance-based segmentation map

Figure A.4: Simulated star–galaxy pair and corresponding importance-based segmentation result.



(a) Simulated image **(b)** Importance-based segmentation map

Figure A.5: Simulated crowded field and corresponding importance-based segmentation result.

Figures A.6, A.7, A.8, and A.9 report the four complementary diagnostics used to characterize the crowded-field test case. Specifically, Figures A.6 and A.8 show the *pairwise recall* and *precision* for stars, while Figures A.7 and A.9 report the corresponding metrics for galaxies.

In all panels, the heat maps display the two metrics as a function of the mean source separation (in pixels) and of the mean signal-to-noise ratio (SNR), for stars and galaxies, respectively. For both source populations, the two metrics are typically found to be around $\sim 0.85 - 0.9$, corresponding to an overall level of over- and under-deblending of approximately 10-15%. These results indicate a promising deblending performance under crowded-field conditions. The present tests were conducted using simulation parameters drawn from the mean properties of the *Euclid* Q1 source catalog only. Ongoing work is extending this validation to a broader region of parameter space in order to more comprehensively characterize the behavior and limitations of the algorithm.

Once the performance limits on simulated images have been fully characterized, we will proceed to stress-test the algorithm on real *Euclid* imaging data, by assessing the impact of the deblending on downstream photometric measurements, which are directly sensitive to deblending efficiency.

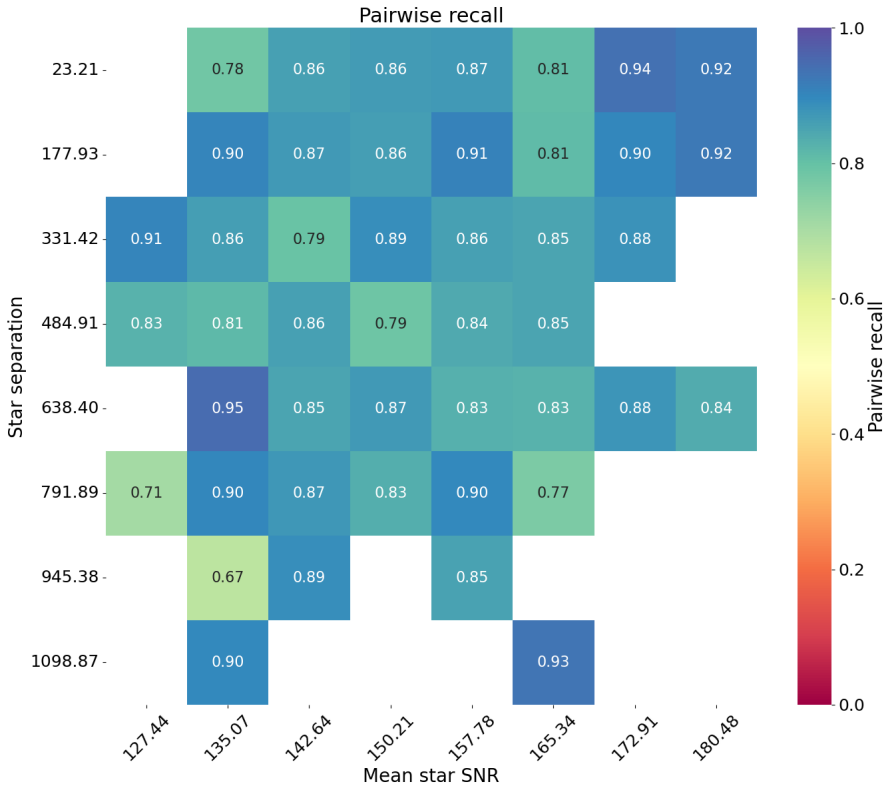


Figure A.6: Pairwise recall for stars in the crowded-field test case, shown as a function of mean source separation (in pixels) and mean SNR.

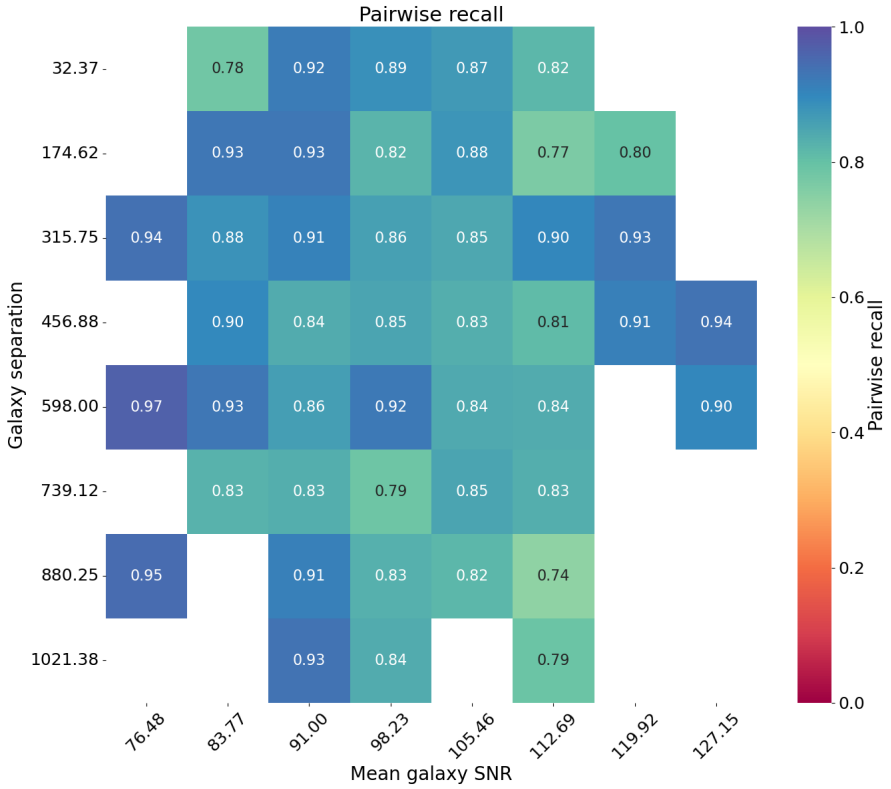


Figure A.7: Same as FigureA.6 for galaxies.

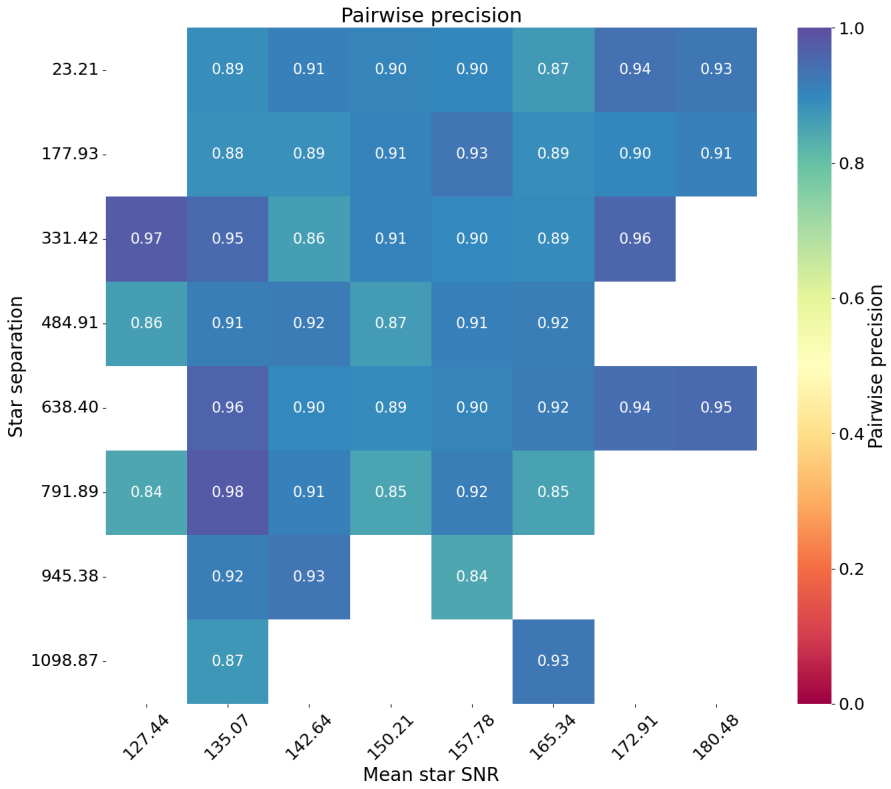


Figure A.8: Pairwise precision for stars in the crowded-field test case, shown as a function of mean source separation (in pixels) and mean SNR.

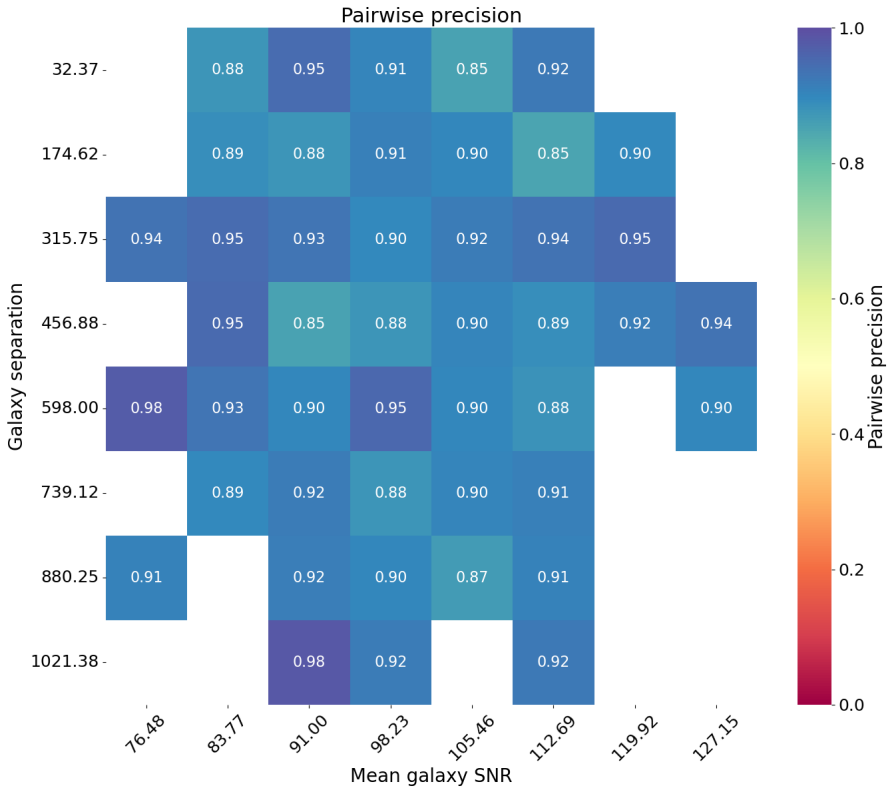


Figure A.9: Same as FigureA.8 for galaxies.

Appendix B

Technical terminology and GPU computing concepts

This Appendix provides a structured overview of the hardware architectures, programming models, and performance concepts used throughout this thesis. It is intended as a reference for readers without a background in high-performance heterogeneous computing.

Central Processing Unit (CPU)

A Central Processing Unit (CPU) is a general-purpose processor optimized for low-latency execution of complex control logic. CPUs feature a limited number of sophisticated cores, deep cache hierarchies, and aggressive branch prediction mechanisms.

Graphics Processing Unit (GPU)

A Graphics Processing Unit (GPU) is a highly parallel processor optimized for throughput-oriented workloads. Modern GPUs contain thousands of lightweight cores organized into Streaming Multiprocessors (SMs) or Compute Units (CUs).

Heterogeneous nodes

A heterogeneous node integrates CPUs and accelerators, typically GPUs, allowing workloads to be distributed according to architectural strengths.

Offloading

Offloading refers to executing selected computational kernels on accelerator devices while maintaining the main control flow on the CPU.

Execution model

Threads are organized into warps (NVIDIA) or wavefronts (AMD), executing in lockstep.

Thread divergence

Thread divergence arises when threads within a warp follow different control paths, leading to serialized execution.

Memory hierarchy

GPUs expose multiple memory spaces including global memory, shared memory, registers, and caches.

Roofline performance model

The Roofline model relates achievable floating-point throughput to memory bandwidth and is used to identify performance bottlenecks.

Performance portability

Performance portability is the ability of a code to achieve high performance across different hardware architectures.

Energy-to-Solution (E2S)

Energy-to-Solution measures total energy consumed to complete a computation.

Time-to-Solution (T2S)

Time-to-Solution measures total execution time.

Streaming Multiprocessor (SM) / Compute Unit (CU)

SMs (NVIDIA) and CUs (AMD) are the fundamental execution blocks of GPUs.

FP64, Tensor, and Matrix Cores

FP64 cores perform double-precision arithmetic. Tensor or Matrix cores accelerate dense matrix operations.

High Bandwidth Memory (HBM)

HBM is a stacked memory technology providing very high bandwidth at reduced power consumption.

Cache hierarchy

GPUs include L1 caches local to SMs/CUs and a shared L2 cache.

Host-to-Device interconnect

Interconnect technologies such as PCIe, Infinity Fabric, and NVLink connect CPUs and GPUs.

Non-Uniform Memory Access (NUMA)

NUMA architectures exhibit variable memory latency depending on physical locality.

Device-to-Device interconnect

Direct GPU–GPU communication fabrics improve bandwidth and reduce latency.

Unified and Coherent memory

Unified memory systems provide coherent CPU–GPU memory spaces.

Compute density

Compute density refers to floating-point throughput per unit area or power.

Memory bandwidth

Memory bandwidth determines data movement rates between cores and memory.

Host–Device transfer overhead

Transfer overhead describes latency and bandwidth limitations for CPU–GPU data movement.

Intra-Node Interconnect Topology

Internal connection layouts influence scalability and contention.

Toolchain maturity

Toolchain maturity reflects compiler and runtime optimization quality.

Strong and Weak Scaling

Strong scaling keeps problem size fixed; weak scaling increases problem size with resources.

Embarrassingly Parallel Workloads

Workloads with minimal inter-thread communication.

Bibliography

Abbott, T. M. C. et al. 2018, Phys. Rev. D, 98, 043526

Adame, A. G. et al. 2025, JCAP, 07, 028

Adamek, J. et al. 2023, JCAP, 06, 035

Ade, P. A. R. et al. 2016, A&A, 594, A13

Alcock, C. & Paczynski, B. 1979, Nature, 281, 358

Amendola, L. et al. 2018, Living Rev. Rel., 21, 2

Aussel, H. et al. 2025 [[arXiv]2503.15302]

Bacon, D. J. et al. 2020, Publ. Astron. Soc. Austral., 37, e007

Barnes, J. & Hut, P. 1986, Nature, 324, 446

Behroozi, P. S., Wechsler, R. H., & Wu, H.-Y. 2013, , 762, 109

Bernardeau, F., Colombi, S., Gaztanaga, E., & Scoccimarro, R. 2002, Phys. Rept., 367, 1

Bertin, E., Schefer, M., Apostolakos, N., et al. 2020, in ASP Conf. Ser., Vol. 527, Astronomical Data Analysis Software and Systems XXIX, ed. R. Pizzo, E. R. Deul, J. D. Mol, J. de Plaa, & H. Verkouter, 461

Blumenthal, G. R., da Costa, L. N., Goldwirth, D. S., Lecar, M., & Piran, T. 1992, , 388, 234

Bock, J. J. et al. 2025 [[arXiv]2511.02985]

Bond, J. R., Cole, S., Efstathiou, G., & Kaiser, N. 1991, ApJ, 379, 440

Bond, J. R. & Myers, S. T. 1996, , 103, 1

Borgani, S. & Kravtsov, A. 2011, Advanced Science Letters, 4, 204

Bos, E. G. P., van de Weygaert, R., Dolag, K., & Pettorino, V. 2012, MNRAS, 426, 440

Bosch, J., Armstrong, R., Bickerton, S., & Furusawa, e. a. 2018, , 70, S5

Buchert, T. 1992, Monthly Notices of the Royal Astronomical Society, 254, 729

Buchert, T. 1995, in International School of Physics, 'Enrico Fermi', Course 132: Dark Matter in the Universe, 543–564

Buchert, T. & Ehlers, J. 1993, Monthly Notices of the Royal Astronomical Society, 264, 375

Burke, C. J., Aleo, P. D., Chen, Y.-C., et al. 2019, Monthly Notices of the Royal Astronomical Society, 490, 3952

Cammelli, V., Monaco, P., Tan, J. C., et al. 2025, , 536, 851

Castander, F. J. et al. 2025, Astron. Astrophys., 697, A5

Castro, T. et al. 2023, Astron. Astrophys., 671, A100

Catelan, P. 1995, Mon. Not. Roy. Astron. Soc., 276, 115

Chartier, N., Wandelt, B., Akrami, Y., & Villaescusa-Navarro, F. 2021, Mon. Not. Roy. Astron. Soc., 503, 1897

Colavincenzo, M. et al. 2019, Mon. Not. Roy. Astron. Soc., 482, 4883

Coles, P. & Lucchin, F. 1995, Chichester: Wiley, |c1995, -1

Contarini, S., Marulli, F., Moscardini, L., et al. 2021, MNRAS, 504, 5021

Contarini, S., Pisani, A., Hamaus, N., et al. 2023, ApJ, 953, 46

Contarini, S., Ronconi, T., Marulli, F., et al. 2019, MNRAS, 488, 3526

Cooray, A. & Sheth, R. 2002, , 372, 1

Corda, S., Veenboer, B., & Tolley, E. 2022, in 2022 IEEE/ACM International Workshop on HPC User Support Tools (HUST), 44–47

Crocce, M., Pueblas, S., & Scoccimarro, R. 2006, MNRAS, 373, 369

Cropper, M. S. et al. 2025, Astron. Astrophys., 697, A2

Dagum, L. & Menon, R. 1998, IEEE Computational Science and Engineering, 5, 46

Dark Energy Survey Collaboration, Abbott, T., Abdalla, F. B., et al. 2016, , 460, 1270

David, H., Gorbatov, E., Hanebutte, U. R., Khanna, R., & Le, C. 2010, in Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design, 189–194

Davies, C. T., Cautun, M., Giblin, B., et al. 2021, , 507, 2267

Davies, C. T., Cautun, M., & Li, B. 2019, , 490, 4907

De Rubeis, E., Gheller, C., Lacopo, G., et al. 2025, in 2025 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 388–395

de Theije, P. A. M., van Kampen, E., & Slijkhuis, R. G. 1998, Mon. Not. Roy. Astron. Soc., 297, 195

Dodelson, S. & Schneider, M. D. 2013, Phys. Rev. D, 88, 063537

Dolag, K., Borgani, S., Murante, G., & Springel, V. 2009, MNRAS, 399, 497

Doroshkevich, A. G. 1970, Astrophysics, 6, 320

Duncan, M. J., Levison, H. F., & Lee, M. H. 1998, AJ, 116, 2067

Dvali, G. R., Gabadadze, G., & Porrati, M. 2000, Phys. Lett. B, 485, 208

d'Errico, M., Facco, E., Laio, A., & Rodriguez, A. 2021, Information Sciences, 560, 476

Efstathiou, G., Davis, M., Frenk, C. S., & White, S. D. M. 1985, ApJS, 57, 241

Ehlers, J. & Buchert, T. 1997, Gen. Rel. Grav., 29, 733

Ester, M., Kriegel, H., Sander, J., et al. 1996, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)

Farber, R. 2011, CUDA application design and development (Elsevier)

Feng, Y., Chu, M.-Y., Seljak, U., & McDonald, P. 2016, Mon. Not. Roy. Astron. Soc., 463, 2273

Fontanot, F., De Lucia, G., Hirschmann, M., et al. 2020, , 496, 3943

Fumagalli, A., Castro, T., Borgani, S., & Valentini, M. 2025a, A&A, 697, A140

Fumagalli, A. et al. 2021, A&A, 652, A21

Fumagalli, A. et al. 2024, Astron. Astrophys., 683, A253

Fumagalli, A. et al. 2025b [[arXiv]2510.13509]

Goz, D., Bertocco, S., Tornatore, L., & Taffoni, G. 2019, in Intelligent Computing, ed. K. Arai, R. Bhatia, & S. Kapoor (Cham: Springer International Publishing), 179–193

Goz, D., Ieronymakis, G., Papaefstathiou, V., et al. 2020, Computation, 8

Greengard, L. & Rokhlin, V. 1987, Journal of Computational Physics, 73, 325

Haigh, C., Chamba, N., Venhola, A., et al. 2021, , 645, A107

Hamaus, N., Pisani, A., Sutter, P. M., et al. 2016, PRL, 117, 091302

Hamaus, N., Sutter, P. M., & Wandelt, B. D. 2014, PRL, 112, 251302

Hamaus, N. et al. 2022, A&A, 658, A20

Hartlap, J., Simon, P., & Schneider, P. 2007, , 464, 399

Heitmann, K. et al. 2019, Astrophys. J. Suppl., 245, 16

Hinneburg, A. & Gabriel, H.-H. 2007, in International symposium on intelligent data analysis, Springer, 70–80

Hinneburg, A. & Keim, D. A. 1998, in Knowledge Discovery and Data Mining

Hu, W. & Sawicki, I. 2007, Phys. Rev. D, 76, 064004

Ilic, S., Langer, M., & Douspis, M. 2013, A&A, 556, A51

Ishiyama, T. et al. 2021, Mon. Not. Roy. Astron. Soc., 506, 4210

Ivezić, Ž. et al. 2019, Astrophys. J., 873, 111

Jahnke, K. et al. 2025, Astron. Astrophys., 697, A3

Jenkins, A., Frenk, C. S., Pearce, F. R., et al. 1998, Astrophys. J., 499, 20

Jennings, E., Li, Y., & Hu, W. 2013, MNRAS, 434, 2167

Kasichayanula, K., Terpstra, D., Luszczek, P., et al. 2012, in 2012 Symposium on Application Accelerators in High Performance Computing, IEEE, 64–73

Kitaura, F.-S., Yepes, G., & Prada, F. 2014, Mon. Not. Roy. Astron. Soc., 439, 21

Kovács, A. 2018, , 475, 1777

Krause, E. et al. 2017 [[arXiv]1706.09359]

Kümmel, M., Álvarez-Ayllón, A., Bertin, E., et al. 2022, arXiv e-prints, arXiv:2212.02428

Lacopo, G., Gheller, C., De Rubeis, E., et al. 2025a, Green computing toward SKA era with RICK

Lacopo, G. et al. 2025b

Lacopo, L. et al. 2025., accepted.

Laureijs, R., Amiaux, J., Arduini, S., et al. 2011, arXiv e-prints, arXiv:1110.3193

Lee, J. & Park, D. 2009, ApJL, 696, L10

Lehman, K., Schuster, N., Lucie-Smith, L., et al. 2025, arXiv e-prints, arXiv:2502.05262

Lepinzan, M. D., Davies, C. T., Castro, T., et al. 2025, Astron. Astrophys., 704, A51

Lepinzan, M. D., Lacopo, G., Goz, D., et al. 2025, arXiv e-prints, arXiv:2510.02873

Levi, M. et al. 2013 [[arXiv]1308.0847]

Li, B. 2011, , 411, 2615

Lippich, M., Sánchez, A. G., Colavincenzo, M., et al. 2018, Monthly Notices of the Royal Astronomical Society, 482, 1786

LSST Science Collaboration, Abell, P. A., Allison, J., et al. 2009, arXiv e-prints, arXiv:0912.0201

Maksimova, N. A., Garrison, L. H., Eisenstein, D. J., et al. 2021, Mon. Not. Roy. Astron. Soc., 508, 4017

Manera, M., Scoccimarro, R., Percival, W. J., et al. 2012, Monthly Notices of the Royal Astronomical Society, 428, 1036

Martineau, M., McIntosh-Smith, S., & Gaudin, W. 2016, in 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 338–347

McElwain, M. W., Feinberg, L. D., Perrin, M. D., & et al. 2023, Publications of the Astronomical Society of the Pacific, 135, 058001

Melchior, P., Moolekamp, F., Jerdee, M., et al. 2018, Astronomy and Computing, 24, 129

Mellier, Y. et al. 2024 [[arXiv]2405.13491]

Message Passing Interface Forum. 2023, MPI: A Message-Passing Interface Standard Version 4.1

Mo, H., van den Bosch, F. C., & White, S. 2010, Galaxy Formation and Evolution

Monaco, P. 1995, ApJ, 447, 23

Monaco, P. 1997, MNRAS, 287, 753

Monaco, P. 2016, Galaxies, 4, 53

Monaco, P., Fontanot, F., & Taffoni, G. 2007, , 375, 1189

Monaco, P., Sefusatti, E., Borgani, S., et al. 2013, MNRAS, 433, 2389

Monaco, P., Theuns, T., & Taffoni, G. 2002a, MNRAS, 331, 587

Monaco, P., Theuns, T., Taffoni, G., et al. 2002b, Astrophys. J., 564, 8

Monaco, P. et al. 2025, A&A, in press [[arXiv]2507.12116]

Moresco, M. et al. 2022, LRR, 25, 6

Moretti, C., Mozzon, S., Monaco, P., Munari, E., & Baldi, M. 2020, Mon. Not. Roy. Astron. Soc., 493, 1153

Munari, E., Monaco, P., Sefusatti, E., et al. 2017, Mon. Not. Roy. Astron. Soc., 465, 4658

Nadkarni-Ghosh, S. & Singhal, A. 2016, Monthly Notices of the Royal Astronomical Society, 457, 2773

Navarro, J. F., Frenk, C. S., & White, S. D. M. 1997, Astrophys. J., 490, 493

Nelson, D. et al. 2019, Comput. Astrophys. Cosmol., 6, 2

Neyrinck, M. C. 2008, MNRAS, 386, 2101

Nicolis, A., Rattazzi, R., & Trincherini, E. 2009, Phys. Rev. D, 79, 064036

Norberg, P., Baugh, C. M., Gaztañaga, E., & Croton, D. J. 2009, Monthly Notices of the Royal Astronomical Society, 396, 19

Oddo, A., Sefusatti, E., Porciani, C., Monaco, P., & Sánchez, A. G. 2020, JCAP, 03, 056

Paillas, E., Lagos, C. D. P., Padilla, N., et al. 2017, Monthly Notices of the Royal Astronomical Society, 470, 4434

Park, D. & Lee, J. 2007, PRL, 98, 081301

Payerne, C., Murray, C., Combet, C., et al. 2023, Mon. Not. Roy. Astron. Soc., 520, 6223

Peebles, P. J. E. 1980, The large-scale structure of the universe

Perico, E. L. D., Voivodic, R., Lima, M., & Mota, D. F. 2019, A&A, 632, A52

Pisani, A., Sutter, P. M., Hamaus, N., et al. 2015, PRD, 92, 083531

Pisani, A. et al. 2019 [[arXiv]1903.05161]

Platen, E., van de Weygaert, R., & Jones, B. J. T. 2007, MNRAS, 380, 551

Potter, D., Stadel, J., & Teyssier, R. 2017, Comput. Astrophys. Cosmol., 4, 2

Press, W. H. & Schechter, P. 1974, Astrophys. J., 187, 425

Radinovic, S. et al. 2023, A&A, 677, A78

Rizzo, L. A., Villaescusa-Navarro, F., Monaco, P., et al. 2017, JCAP, 01, 008

Romelli, E. et al. 2025 [[arXiv]2503.15305]

Ronneberger, O., Fischer, P., & Brox, T. 2015, arXiv e-prints, arXiv:1505.04597

Sachs, R. K. & Wolfe, A. M. 1967, ApJ, 147, 73

Sahni, V. & Coles, P. 1995, Phys. Rept., 262, 1

Salvalaggio, J., Castiblanco, L., Noreña, J., Sefusatti, E., & Monaco, P. 2024, JCAP, 08, 046

Schuster, N., Hamaus, N., Dolag, K., & Weller, J. 2023, JCAP, 2023, 031

Schuster, N., Hamaus, N., Dolag, K., & Weller, J. 2024, JCAP, 2024, 065

Schuster, N., Hamaus, N., Pisani, A., et al. 2019, JCAP, 2019, 055

Schuster, N., Hamaus, N., Pisani, A., Dolag, K., & Weller, J. 2025, arXiv e-prints, arXiv:2509.07092

Shandarin, S. F. & Zeldovich, Y. B. 1989, RMP, 61, 185

Shaw, R. A., Fotopoulou, S., Birkinshaw, M., Maddox, N., & Stewart, H. 2025, RAS Techniques and Instruments, 4, rzaf006

Sheth, R. K., Mo, H. J., & Tormen, G. 2001, Mon. Not. Roy. Astron. Soc., 323, 1

Sheth, R. K. & Tormen, G. 1999, Mon. Not. Roy. Astron. Soc., 308, 119

Sheth, R. K. & Tormen, G. 2002, Mon. Not. Roy. Astron. Soc., 329, 61

Sheth, R. K. & van de Weygaert, R. 2004, MNRAS, 350, 517

Shukla, N., Romeo, A., Caravita, C., et al. 2025, in Proceedings of the 22nd ACM International Conference on Computing Frontiers: Workshops and Special Sessions, CF '25 Companion (New York, NY, USA: Association for Computing Machinery), 177–184

Singh, J., Monaco, P., & Tan, J. C. 2023, Mon. Not. Roy. Astron. Soc., 525, 969

Song, Y., Hu, B., Ruan, C.-Z., Moretti, C., & Monaco, P. 2024, JCAP, 07, 093

Song, Y., Moretti, C., Monaco, P., & Hu, B. 2022, Mon. Not. Roy. Astron. Soc., 516, 5762

Spergel, D., Gehrels, N., Baltay, C., et al. 2015, arXiv e-prints, arXiv:1503.03757

Springel, V. 2005, MNRAS, 364, 1105

Springel, V. 2005, MNRAS, 364, 1105

Springel, V., White, S. D. M., Jenkins, A., et al. 2005, , 435, 629

Springel, V., White, S. D. M., Tormen, G., & Kauffmann, G. 2001, MNRAS, 328, 726

Sutter, P. M., Lavaux, G., Hamaus, N., et al. 2015, A&C, 9, 1

Sutter, P. M., Pisani, A., Wandelt, B. D., & Weinberg, D. H. 2014, MNRAS, 443, 2983

Taffoni, G., Bertocco, S., Coretti, I., et al. 2020a, in Proceedings of the Future Technologies Conference (FTC) 2019, ed. K. Arai, R. Bhatia, & S. Kapoor (Cham: Springer International Publishing), 427–446

Taffoni, G., Bertocco, S., Coretti, I., et al. 2020b, in Proceedings of the Future Technologies Conference (FTC) 2019, ed. K. Arai, R. Bhatia, & S. Kapoor (Cham: Springer International Publishing), 427–446

Taffoni, G., Mignone, A., Tornatore, L., et al. 2024, in Software and Cyberinfrastructure for Astronomy VIII, ed. J. Ibsen & G. Chiozzi, Vol. 13101, International Society for Optics and Photonics (SPIE), 131010X

Tasitsiomi, A., Kravtsov, A. V., Wechsler, R. H., & Primack, J. R. 2004, , 614, 533

Tassev, S., Zaldarriaga, M., & Eisenstein, D. 2013, JCAP, 06, 036

Taylor, A., Joachimi, B., & Kitching, T. 2013, Monthly Notices of the Royal Astronomical Society, 432, 1928

Tomba, F. et al. in prep., in prep.

Tomov, S., Haidar, A., Ayala, A., Schultz, D., & Dongarra, J. 2019

Tramacere, A., Paraficz, D., Dubath, P., Kneib, J.-P., & Courbin, F. 2016, Monthly Notices of the Royal Astronomical Society, 463, 2939

Veropalumbo, A., Binetti, A., Branchini, E., et al. 2022, JCAP, 09, 033

Villaescusa-Navarro, F. et al. 2020, Astrophys. J. Suppl., 250, 2

Vogelsberger, M., Genel, S., Springel, V., et al. 2014, Nature, 509, 177

Warren, M. S., Abazajian, K., Holz, D. E., & Teodoro, L. 2006, ApJ, 646, 881

Watson, W. A., Iliev, I. T., D'Aloisio, A., et al. 2013, MNRAS, 433, 1230

Weinberg, D. H., Mortonson, M. J., Eisenstein, D. J., et al. 2013, , 530, 87

Xavier, H. S., Abdalla, F. B., & Joachimi, B. 2016, Mon. Not. Roy. Astron. Soc., 459, 3693

Xu, G.-H. 1995, ApJS, 98, 355

Zel'dovich, Y. B. 1970, , 5, 84