SISSA
Scuola
Internazionale
Superiore di
Studi Avanzati

Physics Area - PhD course in
Astrophysics and Cosmology

From cosmic voids to collapsed structures: HPC methods for Astrophysics and Cosmology

Candidate:
Tommaso Ronconi

Advisors:
Andrea Lapi
Matteo Viel

Academic Year 2019-20

Abstract

Computational methods, software development and High Performance Computing awareness are of ever-growing importance in Astrophysics and Cosmology. In this context, the additional challenge comes from the impossibility of reproducing experiments in the controlled environment of a laboratory, making simulations unavoidable for testing theoretical models. In this work I present a quite heterogeneous ensemble of projects we have performed in the context of simulations of the large scale structure of the Universe. The connection being the development and usage of original computational tools for the analysis and post-processing of simulated data.

In the first part of this manuscript I report on the efforts to develop a consistent theory for the size function of cosmic voids detected in biased tracers of the density field. Upcoming large scale surveys will map the distribution of galaxies with unprecedented detail and up to depths never reached before. Thanks to these large datasets, the void size function is expected to become a powerful statistics to infer the geometrical properties of space-time. In spite of this, the existing theoretical models are not capable of describing correctly the distribution of voids detected, neither in unbiased nor in biased simulated tracers.

We have improved the void selection procedure, by developing an algorithm that redefines the void ridges and, consequently, their radii. By applying this algorithm, we validate the volume conserving model of the void size function on a set of unbiased simulated density field tracers. We highlight the difference in the internal structure between voids selected in this way and those identified by the popular VIDE void finder. We also extend the validation of the model to the case of biased tracers. We find that a relation exists between the tracer used to sample the underlying dark matter density field and its unbiased counterpart. Moreover, we demonstrate that, as long as this relation is accounted for, the size function is a viable approach for studying cosmology with voids.

Finally, by parameterising the size function in terms of the linear effective bias of tracers, we perform an additional step towards analysing cosmic voids in real surveys. The proposed size function model has been accurately calibrated on halo catalogues, and used to validate the possibility to provide forecasts on the cosmological constraints, namely on the matter density parameter, Ω_M, and on the normalisation of the linear matter power spectrum, σ_8.

The second part of the manuscript is focused in presenting the hybrid C++/python implementation of ScamPy, our empirical framework for "painting" galaxies on top of the

Dark Matter Halo/Sub-Halo hierarchy obtained from N-body simulations.

Our confidence on the reliability of N-body Dark Matter-only simulations stands on the argument that the evolution of the non-collisional matter component only depends on the effect of gravity and on the initial conditions. The formation and evolution of the luminous component (i.e. galaxies and intergalactic baryonic matter) are far from being understood at the same level as the dark matter.

Among the possible approaches for modelling the luminous component, *empirical* methods are designed to reproduce observable properties of a target (observed) population of objects at a given moment of their evolution. With respect to *ab initio* approaches (i.e. hydrodynamical N-body simulations and semi-analytical models), empirical methods are typically cheaper in terms of computational power and are by design more reliable in the high redshift regime.

Building an empirical model of galaxy occupation requires to define the hosted-object/hosting-halo connection for associating to the underlying DM distribution its baryonic counterpart. The method we use is based on the sub-halo clustering and abundance matching (SCAM) scheme which requires observations of the 1- and 2-point statistics of the target population we want to reproduce. This method is particularly tailored for high redshift studies and thereby relies on the observed high-redshift galaxy luminosity functions and correlation properties.

The core functionalities of ScamPy are written in C++ and exploit Object Oriented Programming, with a wide use of polymorphism, to achieve flexibility and high computational efficiency. In order to have an easily accessible interface, all the libraries are wrapped in python and provided with an extensive documentation. I present the theoretical background of the method and provide a detailed description of the implemented algorithms. We have validated the key components of the framework, demonstrating it produces scientifically meaningful results with satisfying performances.

Finally, we have tested the framework in a proof-of-concept application at high-redshift. Namely, we paint a mock galaxy population on top of high resolution dark matter only simulation, mimicking the luminosity and clustering properties of high-redshift Lyman Break Galaxies retrieved from recent literature. We use these mock galaxies to infer the ionizing radiation spatial and statistical distribution during the period of Reionization.

Contents

Chapter 1

Hic et nunc

Computational methods are of crucial importance in Astrophysics and Cosmology. The reason for that is twofold. On the one side, given the impossibility to reproduce the physical phenomena involved within the controlled environment of a laboratory, computer simulations are the only tool that cosmologists and astrophysicists have to test directly their hypothesis on the processes responsible for the observations. Since the complexity of the models to be computed grows as the components to be modelled increase and the physical processes become more refined, the computational cost has a consequent increase. On the other side, datasets coming from observations of the Universe are constantly growing in number and size, requiring for methods of ever growing efficiency to be analysed. Furthermore, considered the huge amount of data that will be harvested on a daily basis, future experiments such as JWST (Gardner et al. 2006), DESI (Levi et al. 2013), Euclid (Amendola et al. 2018) and LSST (Ivezić et al. 2019) will require methods for big data live analysis and reduction.

Astrophysical and Cosmological research invariably involves numeric. On the theoretical side, the use of numerical methods is unavoidable for modelling non analytically solvable systems (e.g. the non-linear evolution of structures or complex hydro-dynamical processes). On the observational side, data mining and reduction tools are as important as cutting edge technologies for unravelling the mysteries of the Universe.

Under this perspective, High Performance Computing (HPC) methods and advanced programming techniques become an enabling technology for doing science. Awareness of the HPC methods and best-practices has unavoidably grown over the years in the community, both during formation and later on with careers in research specifically oriented on the development of solid computational tools. Therefore, as much as experiments are accurately designed to have the longest life-span possible, scientific software applications started to be intended for a long term use and for the deployment to the community.

In this work I present the two main projects I have been working on in the past four years, where the leitmotiv have been the development and usage of computational tools, intended for release to the public. The scientific focus is centred on the study of the Large Scale Structure (LSS) of the Universe, i.e. the statistical distribution of galaxies, clusters and voids on scales above the Megaparsec.

The growth of the primordial quantum fluctuations in the gravitational potential leads to the observed LSS. As a result, studying the structures which populate the Universe should provide information about the Universe as whole, its energy content and the processes which dictated its evolution from the primordial stages to what is observed today.

Since on sufficiently large scales the predominant interaction is gravity, the statistical distribution of Dark Matter (DM) structures in the Universe should only depend on the initial conditions and on the effect of gravity. Cosmological models are therefore described in terms of General Relativity, where the effect of gravity is encoded in the geometry of space-time.

In this work I decided to avoid the traditional rephrasing of the derivation of the Friedmann Equations and the consequent definition of cosmological models from arguments of General Relativity. First because there are plenty of textbooks that cover the topic much more in dept than I could possibly hope to produce in this manuscript. On the second place, since I present research produced on a quite heterogeneous set of topics, I provide for each of them a contextualisation in the dedicated sections.

The common ground among the several topics covered in this manuscript is the study of evolved structures of the Universe, which result from the collapse or the expansion of initial primordial fluctuations. The growth of primordial perturbations on the density field can be studied linearly until the contrast in density with respect to the background is small ($\delta \ll 1$), afterwards approximated solutions and computer simulations are the only methods to predict the behaviour of the LSS. In the rest of this Chapter I therefore provide a general introduction to linear theory of structure growth (Section 1.1), in which I also provide context for the topics that are covered in the next chapters and finally, in Section 1.2, I give an overview on the structure of this manuscript, also providing a list of published papers where our results have been presented to the community.

1.1 On the growth of cosmological structures

The currently accepted cosmological model is called *flat*-ΛCDM. It is defined by six main parameters and on the assumption of flatness. The energy content of the Universe is dominated by a cosmological constant, Λ, and by Cold Dark Matter (CDM). The current density distribution is the result of initial Gaussian adiabatic fluctuations seeded in an inflational period of fierce accelerated expansion.

The structures we are able to observe in the present Universe result from the gravitational collapse of these initial fluctuations in the density field. On the large scale their occurrence dramatically depends not only on the action of gravity itself but on the background evolution too. Namely, the evolution of a perturbation on the energy distribution is the result of the action of gravity on a space in which the geometry depends on the energy distribution itself. Almost all viable existing theories for structure formation are hierarchical: the matter distribution evolves through a sequence of ever larger structures. Hierachical scenarios of structure formation have proven to be successful in predicting the evolution of both positive and negative fluctuations around the average.

It can be derived that the average energy density at redshift z in a flat Universe is

$$\rho_c(z) \equiv \frac{3H^2(z)}{8\pi G} \ , \tag{1.1}$$

where $H(z)$ is the Hubble parameter which measures the expansion velocity of the system and G is the gravitational constant. We can use Eq. (1.1) to define the dimensionless density parameter of component w at redshift z as

$$\Omega_w(z) \equiv \frac{\rho_w(z)}{\rho_c(z)} \ . \tag{1.2}$$

If we assume adiabatic expansion, each energetic component, w, in the Universe evolves following

$$\Omega_w(z) = \Omega_{0,w}(1+z)^{3(1+w)} \ , \tag{1.3}$$

where w is the equation of state parameter which varies for each different species, $\Omega_{0,w}$ is the density parameter of component w measured in the present Universe and z is the redshift.

Prompted into the second Friedmann Equation, these definitions lead to an expression for the evolution of the Hubble parameter in redshift:

$$H^2(z) = H_0^2(1+z)^2\big[1 - \sum_i \Omega_{0,w_i} + \sum_i \Omega_{0,w_i}(1+z)^{1+3w_i}\big] \ , \tag{1.4}$$

where $\sum_i \Omega_{0,w_i} \equiv \Omega_{\rm tot}$ is the sum of all the i-th components of the cosmological fluid, labelled with their value of w_i, and where the subscript 0 indicates a measure of the parameter at redshift $z = 0$. The quantity $1 - \Omega_{\rm tot}$ is a measure of curvature and therefore in a flat universe it should be equal to zero.

The non linear structure of the cosmic web, consisting of clusters, voids, filaments and walls is a natural outcome of the gravitational instability of the anisotropies injected in the cosmological fluid by the initial inflatory expansion of the Universe. This scenario is confirmed by an overwhelming body of observational and theoretical evidencies. Matter is attracted by overdense regions and evacuated from underdense ones, thus amplifying already existing inhomogeneities.

Assuming that the Universe is homogeneous and isotropic and that its evolution is adiabatic, gravitational instability can be studied by perturbing the following system of hydrodinamical equations:

$$\dot{\rho} + 3H(z)\rho = 0 \qquad \text{mass conservation} \tag{1.5a}$$

$$\dot{\mathbf{v}} + (\mathbf{v}\cdot\nabla)\,\mathbf{v} = -\frac{1}{\rho}\nabla\rho - \nabla\Phi \qquad \text{Euler equation} \tag{1.5b}$$

$$\nabla^2\Phi = 4\pi G\rho \qquad \text{Poisson equation} \tag{1.5c}$$

$$p(\rho, S) = p(\rho) \qquad \text{equation of state} \tag{1.5d}$$

$$\dot{S} = 0 \qquad \text{adiabatic condition} \tag{1.5e}$$

where $\rho \equiv \rho_{\rm B}$ is the background density, $\mathbf{v}$ is the velocity vector of a fluid element, Φ the gravitational potential, p the pressure and S the entropy. A dimensionless density fluctuation, in a Universe expanding as from Eq. (1.4), is defined as

$$\delta(\mathbf{x},t) \equiv \frac{\delta\rho(\mathbf{x},t)}{\rho_{\rm B}(t)} \ . \tag{1.6}$$

By requiring this perturbation to be small, $\delta \ll 1$, the system (1.5) can be linearized. The solving equation for density is a differential equation which, in Fourier space, has the form

$$\ddot{\delta}_{\mathbf{k}} + 2H(t)\dot{\delta}_{\mathbf{k}} + \left[k^2c_s^2 - 4\pi G\rho_{\rm B}(t)\right]\delta_k = 0 \ , \tag{1.7}$$

where

$$\delta_{\mathbf{k}} = \delta(\mathbf{k},t) = \frac{1}{(2\pi)^3}\int \mathrm{d}^3\mathbf{x}^{-i\mathbf{k}\cdot\mathbf{x}}\delta(\mathbf{x},t) \tag{1.8}$$

is the Fourier transform of $\delta(\mathbf{x},t)$, $k = |\mathbf{k}|$ is the amplitude of the Fourier mode, $c_s^2 \equiv (\partial p/\partial\rho)_S$ is the square of the sound speed and a dot denotes a derivative with respect to time t.

Eq. (1.7) is the dispersion relation, where the term $2H(t)\dot{\delta}_{\mathbf{k}}$ is the Hubble friction, while $k^2c_s^2\delta_{\mathbf{k}}$ accounts for the characteristic velocity field of the fluid. Both these terms oppose to the growth of fluctuations.

Being a second order differential equation, Eq. (1.7) has solutions for $\delta(\mathbf{x},t)$ with growing and decaying modes in the form

$$\delta(\mathbf{x},t) = A(\mathbf{x})D_+(t) + B(\mathbf{x})D_-(t) \tag{1.9}$$

where A and B are functions of the comoving coordinate $\mathbf{x}$ and D_+ and D_- are functions of the time coordinate. Growing solutions are those having scale larger than the Jeans scale lenght which, in physical units, is defined as

$$\lambda_{\rm J} \equiv c_s\sqrt{\frac{\pi}{G\rho_{\rm B}}} \tag{1.10}$$

where the dependence to the cosmological model is implicitly expressed through c_s and $\rho_{\rm B}$.

From the observation of the temperature anisotropies in the Cosmic Microwave Background, we know that density fluctuations appear at some very early time. It therefore follows that, at later time, decaying solutions have been fading away and only growing solutions remain. We can simplify the notation for the growth factor to $D_+(t) \to D(t)$ or, equivalently, $D(z)$.

Substituting the time dependence with a dependency on redshift, Eq. (1.7) can be rewritten into an equation for the growth factor in real space

$$\ddot{D} + 2H(z)\dot{D} - \frac{3}{2}\Omega_{0,m}H_0^2(1+z)^3D = 0 \ . \tag{1.11}$$

For an Einsten-de Sitter Universe with $\Omega_m = 1$, the solution is trivially $D \propto 1/(1+z)$. For a flat two-component Universe with $\Omega_M < 1$ and $\Omega_\Lambda = 1 - \Omega_m$, the solution for the growth factor has the integral form

$$D(z) = \frac{H(z)}{H_0} \int_z^\infty \frac{\mathrm{d}z'(1+z')}{H^3(z')} \left[\int_0^\infty \frac{\mathrm{d}z'(1+z')}{H^3(z')} \right]^{-1} \tag{1.12}$$

where the factor in brackets normalizes $D(z)$ to 1 at redshift $z = 0$. Whenever the cosmological model cannot be approximated to the latter two cases, no general integral expression exists. Eq. (1.11) has therefore to be solved either analytically case-by-case or by using numerical integration methods.

We have provided a primer to the linear theory of structure formation in an expanding Universe. As long as $\delta \ll 1$ holds, density fluctuations simply "grow in place" in comoving coordinates proportionally to the growth factor:

$$\delta(\mathbf{x}, z) = \delta(\mathbf{x}, z_i) \frac{D(z)}{D(z_i)} \ . \tag{1.13}$$

The cosmological density field is a superposition of fluctuations that independently evolve following Eq. (1.13). In Fourier space the statistics of the several Fourier modes with different amplitude can be modelled. This implies that any comparison between theory and observations will also have to be statistical. The power spectrum of the density field is defined as the variance of the amplitudes at a given value of $\mathbf{k}$:

$$\langle \delta(\mathbf{k}) \delta^*(\mathbf{k}') \rangle = (2\pi)^3 P(k) \delta_D^{(3)}(\mathbf{k} - \mathbf{k}') \tag{1.14}$$

where $\delta_D^{(3)}$ is the 3-dimensional Dirac delta function. In definition (1.14) the substitution $P(\mathbf{k}) = P(k)$ is a consequence of the Cosmological Principle.

The shape of the initial fluctuation spectrum is assumed to be imprinted in the Universe at some arbitrarily early time. Since the accepted inflation scenario states that the seeds of today's structures are generated by stochastic quantum fluctuations in a scalar field (i.e. the inflaton), the initial $P(k)$ should produce a power-law form

$$P(k) = Ak^n \tag{1.15}$$

where the spectral index is generally assumed to be close to unity, $n \approx 1$ (Zeldovich 1972), and the amplitude has to be measured from observations.

In real space, the assumption of stochastical generation of fluctuations implies their amplitudes have a Gaussian distribution. Although the mean value of the perturbation $\delta(\mathbf{x}) = \delta$ across the statistical ensemble is identically zero by definition, its mean square value, i.e. its variance σ^2, is not. It can be shown that

$$\sigma^2 \equiv \langle |\delta^2| \rangle = \frac{1}{2\pi^2} \int_0^\infty P(k) k^2 \, \mathrm{d}k \ , \tag{1.16}$$

in the limit where the number of realisations is infinite and assuming the Cosmological Principle. The latter expression is valid for each position in space. Measuring σ^2, though,

requires the reconstruction of the density field, which is far from being trivial. In fact, especially in today's Universe, the observable mass tracers are not distributed smoothly across space.

Therefore, instead of using a punctual variance, it is convenient to describe the fluctuation field as a function of some resolution scale R. Let $\langle M \rangle$ be the mean mass found inside a spherical volume of radius R, then we can define a density fluctuation from discrete tracers as

$$\delta_M = \frac{M - \langle M \rangle}{\langle M \rangle} \ . \tag{1.17}$$

With this definition, from Eq. (1.16) follows the definition of mass variance:

$$\sigma_M^2 \equiv \langle \delta_M^2 \rangle = \frac{\langle (M - \langle M \rangle)^2 \rangle}{\langle M \rangle^2} \ , \tag{1.18}$$

which is the convolution of the punctual variance with a window function $W(\mathbf{x}, R)$- S ince

$$\delta_M(\mathbf{x}) = \delta(\mathbf{x}) \otimes W(\mathbf{x}, R) \ . \tag{1.19}$$

Eq. (1.18) and Eq. (1.19), prompted into Eq. (1.16), provide a definition of the mass variance dependent on the power spectrum:

$$\sigma_M^2 = \frac{1}{(2\pi)^3} \int \mathrm{d}\mathbf{k}^3 \langle P(k) \rangle \hat{W}^2(\mathbf{k}, R) \ , \tag{1.20}$$

where $\hat{W}(\mathbf{k}, R)$ is the Fourier transform of the window function which implicitly makes the expression mass dependent through its dependence on the scale R.

Since the higher values of k tend to be averaged out within the window volume, σ_M^2 is dominated by perturbation components with wavelength $\lambda \sim k^{-1} > R$. The Zeldovic spectrum (1.15) is a growing function of k, thus waves with much larger λ have only a small contribution and vice-versa.

When the density contrast of perturbations approaches unity ($\delta \approx 1$), linear theory cannot be applied anymore, since its basic assumption fades. It is necessary to abandon the small-perturbations approximation and develop a non-linear theory for the evolution of anisotropies.

After the linear regime breaks, when δ becomes comparable to unity, a mildly non-linear regime sets in. During this regime, even though the density contrast may remain less than unity, the perturbations distribution function starts to evolve into a non-Gaussian shape. The power spectrum changes by virtue of a complex cross-talk between different wave-modes.

We can observe structures that complete their evolution without deviating from the mildly non-linear regime. This is the case of under-densities evolving into cosmic voids. The growth of negative perturbations drives the internal density to diminish while they expand, eventually encountering the physical limit $\delta_{\min} = -1$.

Further into the non linear regime, over-dense bound structures form. The gravitational potential due to the baryonic content in these objects eventually becomes relevant and the simple one-component treatment presented cannot be applied anymore. As a result of hydrodynamical effects, star formation, heating and cooling of gas, the spatial distribution of galaxies may be very different from the distribution of the DM, even on large scales.

Within sufficiently large volumes, it is generally believed that relative fluctuations in the object number counts and matter density fluctuations are proportional to each other. It is therefore a fair assumption that the point-like distribution of haloes, galaxies or clusters, $n(\mathbf{r}) \equiv \sum_i \delta_D^{(3)}(\mathbf{r}-\mathbf{r}_i)$, bears a simple functional relationship to the underlying density contrast $\delta(\mathbf{r})$. This is expressed by a linear biasing prescription:

$$\frac{\delta\, n(\mathbf{r})}{\langle n \rangle} = b\, \delta(\mathbf{r}) \tag{1.21}$$

where b is the (linear) bias parameter. With such an ansatz, linear theory can be used on large scales to relate galaxy clustering statistics to those of the underlying density fluctuations, e.g. $\sigma^2_{\rm gxy} = b^2 \sigma^2_M$.

These considerations though break down when the clustering is highly non linear (i.e. on scales smaller than few Megaparsecs). Furthermore, linear and higher order perturbation theories do not provide a rigorous explanation for how the clustering of galaxies differs from that of dark matter. Nevertheless, it provides a halo-based description of the dark matter distribution of large scale structure. This is extremely useful because, following the pioneering work of White & Rees (1978), the idea that galaxies form within such dark matter haloes has gained increasing popularity, also thanks to the modelling through large cosmological simulations with baryonic hydro-dynamical effects implemented, e.g. Illustris (Vogelsberger et al. 2014b,a), EAGLE (Schaye et al. 2015), Magneticum (Dolag et al. 2016), Illustris TNG (Nelson et al. 2019).

In this picture, the physical properties of galaxies are determined by the halos in which they form. Therefore, the statistical properties of a given galaxy population are determined by the properties of the parent halo population. This assumption allows the development of a powerful tool for modelling the dependence of galaxy clustering and galaxy type relative to the underlying dark matter distribution. Such an approach is usually referred to as the Halo Model. We describe its basic assumptions and derive relevant galaxy clustering statistics in Section 3.1.1.

Not only it allows to determine the astrophysical parameters regulating the galaxy distribution within haloes but it also provides an instrument for relating the observational properties of a given galaxy population to simulations of the dark matter density field.

1.2 Published works

This Thesis is based on the two main projects I have been working on in the last four years. My work as a PhD student in the Astrophysics and Cosmology Group at SISSA was focused on the development and testing of original computational tools.

I have completed a project on the exploitation of the size function of cosmic voids as a cosmological probe. Cosmic voids occupy most of the volume of the Universe, and their statistical properties can be exploited for constraining dark energy, as well as for testing theories of gravity. Nevertheless, in spite of their growing popularity as cosmological probes, a gap of knowledge between cosmic void observations and theory still persists. In our research we attempted in covering this gap, by developing a consistent theory for the size distribution of cosmic voids identified in biased tracers of the density field.

The second part of my research was focused on the development of a stand-alone software package for the analysis of observed clustering statistics of different order and the production of empirical models for galaxy mock catalogues. We have extensively tested our software both scientifically and in terms of performance and the application has been made publicly available. Empirical models are by design particularly suitable for addressing the modelling of the high redshift Universe, but they rely on the availability of high redshift observations of the population to be modelled. We have used our application to perform a preliminary study on the spatial distribution of ionized bubbles during Reionization, using observations of Lyman Break Galaxies in the redshift window $4 \leq z \leq 7$.

Part I of this work is dedicated to the distribution of cosmic voids as a function of their size. For my Master Thesis I had developed C++ methods for the reduction and analysis of cosmic void catalogues in N-body simulations. The work has been published in Ronconi & Marulli (2017) and the source code has been deployed within the CosmoBolognaLib, a large collection of C++ libraries for cosmological calculations (Marulli et al. 2016). In the first years of my PhD I have finalized this work, collaborating with the Clustering Group in Bologna University. This collaboration have produced two publications, both exploiting the computational tools presented in Ronconi & Marulli (2017):

- in Ronconi et al. (2019) we demonstrate that the number count statistics of cosmic voids in biased tracers of the density field is completely determined by the cosmological model and by a relation between the density contrast of void tracers and Dark Matter inside voids.
- In Contarini et al. (2019) we have further validated our theoretical framework, extending it to redshift higher than $z = 0$. We find a tight relation between the large scale tracer bias and the one measured inside cosmic voids and we use this relation to parameterise the size function model. This allows to infer cosmological constraints from the one point statistics of cosmic voids.

In Part II I present the HPC-driven development of a stand-alone application for the production of empirically motivated mock-galaxy catalogues. The source code is publicly available at github.com/TommasoRonconi/scampy. Documentation and examples are provided at scampy.readthedocs.io and the code is indexed at Ronconi et al. (2020b).

- The library is presented in Ronconi et al. (2020a) where we also provide validation of its functionalities, some benchmarking of the performances and a simple application

to the case of high redshift galaxies producing ionizing radiation. We provide a detailed description of the implemented algorithms in Chapter 3. Implementation details and benchmarking are discussed in Chapter 4. In Chapter 5 we use our library to produce catalogues of mock Lyman break galaxies, we then use their luminosity to infer the production of ionizing photons at different redshift and analyse the resulting large-scale ionized-hydrogen distribution.

- A further application of the library in which we use its functionalities on a real galaxy catalogue has been presented in Bonavera et al. (2020). We have used the `halo_model` module of our library to obtain an independent estimation of the HOD parameters that correspond to the foreground galaxies acting as lenses for a work focused on the study of the magnification bias cosmological constraining power. This effort is described in Section 3.4 as a further validation of our framework.

Part I

Cosmic Voids

Chapter 2

Cosmic void size function on biased tracers

Cosmic voids are large underdense regions where matter is evacuated and then squeezed in between their boundaries. While galaxy clusters encapsulate most of the mass in the Universe, voids dominate it in terms of volume. Voids are only mildly non-linear objects since, during their evolution, they encounter a crucial physical limit: matter density cannot be less than zero, that is the minimum possible density contrast is $\delta = -1$. Conversely to what happens to overdensities, the void evolution tends to reduce the possible non-sphericity of the initial density perturbations (Icke 1984). These properties suggest that the employment of a simplified spherical expansion model may be more accurate in describing the void formation and evolution than it is in the description of overdensities (Blumenthal et al. 1992).

In the last decade, cosmic voids have gathered a large popularity as cosmological tools. With typical sizes over tens of megaparsecs, they are by far the largest observable structures in the Universe. For this reason they are particularly suited to provide information about several hot topics. They are indeed extreme objects, meaning that they generate from the longest wavelengths of the matter perturbation power spectrum. Moreover, their underdense nature makes them suitable for a wide range of dark energy measurements. Indeed, the *super-Hubble velocity field* within voids causes a suppression in structure growth, making them a pristine environment in which to study structure formation (Benson et al. 2003; Van de Weygaert & Platen 2011), and a test-bench for constraining dark matter (DM), dark energy and theories of gravity (Lavaux & Wandelt 2012; Li et al. 2012; Sutter et al. 2014; Massara et al. 2015; Hamaus et al. 2015, 2016; Pollina et al. 2016; Kreisch et al. 2019).

The excursion-set model has been applied to the study of underdensities, in particular to predict the size function of cosmic voids detected in the DM field (Sheth & van de Weygaert 2004, hereafter SvdW). It is based on the assumption that cosmic voids are nearly spherical regions which have gone through shell-crossing. The latter represents the time when the inner, more underdense, matter shells inside voids have reached the outer, as a result of the differential outward acceleration they experience. Jennings et al.

(2013) then proposed a modification of this model that makes it in better agreement with the actual size distribution of cosmic voids measured in both simulated and real void catalogues.

The first goal of our work is to further validate the Jennings et al. (2013) size function model on simulated void catalogues detected from unbiased density field tracers, at different redshifts and numerical resolutions. Then we extend the model to account for the more realistic case of biased tracers, covering the existing gap between the theoretical and observational sides of the problem. These results are described in Ronconi et al. (2019) and have been obtained by exploiting the computational machinery presented in Ronconi & Marulli (2017). Finally, we provide a summary of the follow-up work, which has been published in Contarini et al. (2019), in which we demonstrate the cosmological constraining power of the framework we have developed.

This Chapter is organised as follows. In Section 2.1 the existing void size function theoretical models are presented. We focus in particular on their realm of applicability, underlining the existing issues in modelling void statistics in biased matter distributions. We then propose a modification of the Jennings et al. (2013) void size function model to account for the tracer bias. Section 2.2 is dedicated to the validation of our fiducial model. We investigate its reliability using void catalogues extracted from a set of cosmological N-body simulations with different selections. Specifically, while in Section 2.2.1 we make use of DM particle simulations, meaning that voids are detected from unbiased tracers of the density field, in Section 2.2.2 we extend our study to voids identified in the distribution of biased DM haloes. Finally, in Section 2.2.3, we demonstrate the constraining power of the framework we have developed. In Section 2.3 we recap the results and draw our conclusions.

2.1 Methodology

Whether it is possible to exploit cosmic voids as cosmological probes is a matter of how reliable is our capability to model their properties. The most straightforward measure we can compute from whatever survey of extragalactic sources is the abundance of sources as a function of a particular feature. In the case of cosmic voids, this is the size function.

In Section 2.1.1, we first present the existing theoretical models for the size function of cosmic voids extracted from unbiased tracers of the DM density field. Then we introduce our new parameterisation to describe the void size function in the more realistic case of void catalogues extracted from biased tracer distributions. In Section 2.1.2, we introduce the detailed procedure that we will use later on to identify the cosmic voids in a set of DM-only N-body simulations, and to clean the catalogue.

The simulations employed in the present work have been performed with the C-GADGET module (Baldi et al. 2010), a modified version of the (non-public) GADGET3 N-body code (Springel 2005), while the software exploited for the post-processing and the data analysis[1] has been presented in Ronconi & Marulli (2017, RM17 hereafter),

[1]Specifically, we use the numerical tools to handle cosmic void catalogues provided by the CosmoBolognaLib, V5.1 (Marulli et al. 2016). This consists of a large set of C++/Python *free software*

where we also provided a first description of our new theoretical size function model and cleaning method.

2.1.1 Size function

The development of any theoretical model of void size function relies on the definition of cosmic voids. Let us stick with the common assumption that an isolated void evolves from an initial spherical top-hat depression in the density field. This is analogous to the assumption that an isolated DM halo results from the spherical collapse of a top-hat peak in the density field. The evolution of both overdensities (peaks) and underdensities (depressions) can be described via the classical spherical evolution model (Blumenthal et al. 1992). In the overdensity case, a halo is said to have formed at the moment its density contrast reaches a level corresponding to either the virialisation of the spherical perturbation (the critical linear overdensity $\delta_c \approx 1.69$, for an EdS universe, see e.g. Bond et al. 1991; Paranjape et al. 2012) or, with a milder assumption, when the perturbation reaches the *turn-around* ($\delta_t \approx 1.06$, for an EdS universe[2]), and detaches from the overall expansion of the Universe.

On the other hand, cosmic voids do not ever detach from the overall expansion, but instead they expand with a *super-Hubble flow*. The expansion rate is inversely proportional to the embedded density, therefore shells centred around the same point are expected to expand faster as closer as they are to the centre. This eventually leads to the inner shells reaching the outer, the so-called *shell-crossing*, which is typically assumed to be the moment when a void can be said to have formed. In linear theory, the shell-crossing occurs at a fixed level of density contrast ($\delta_v^L \approx -2.71$, for an EdS universe[3]). This threshold can be used in an excursion-set framework to predict the typical distribution of voids, analogously to the case of halo formation by spherical collapse (Bond et al. 1991; Zentner 2007). With these assumptions, SvdW developed a model of the void size function that, furthermore, considers the *void-in-cloud* side effect, which accounts for the squeezing of voids that happen to evolve within larger scale overdensities.

In this framework, the probability distribution of voids of a certain size is given by

$$f_{\ln\sigma}(\sigma) = 2\sum_{j=1}^{\infty} j\pi x^2 \sin(j\pi\Delta)\exp\left[-\frac{(j\pi x)^2}{2}\right], \tag{2.1}$$

where

$$x = \frac{\Delta}{|\delta_v^L|}\,\sigma \tag{2.2}$$

and

$$\Delta = \frac{|\delta_v^L|}{\delta_c + |\delta_v^L|}\,. \tag{2.3}$$

libraries, freely available at the following GitHub repository: https://github.com/federicomarulli/CosmoBolognaLib.

[2]We will generically refer to both the *critical* and the *turn-around* densities with the same symbol, δ_c.

[3]We will use the superscript L to identify linear extrapolated quantities, while the superscript NL will mark their non-linear counterparts.

The size-dependence of Eq. (2.1) is given by the square root of the variance,

$$\sigma^2(r) = \frac{1}{2\pi}\int k^2 P(k) W^2(k,r) \mathrm{d}k\,, \tag{2.4}$$

where $P(k)$ is the matter power spectrum, $W(k,r)$ the window function, and r is the smoothing scale, that is the radius of the spherical underdense region we define as void. In this work we apply the approximation proposed by Jennings et al. (2013), which is accurate at the $\sim 0.2\%$ level:

$$f_{\ln\sigma}(\sigma) = \begin{cases} \sqrt{\frac{2}{\pi}}\frac{|\delta_v^L|}{\sigma}\exp\left(-\frac{{\delta_v^L}^2}{2\sigma^2}\right) & x \leq 0.276 \\ 2\sum_{j=1}^{4} j\pi x^2 \sin(j\pi\Delta)\exp\left[-\frac{(j\pi x)^2}{2}\right] & x > 0.276\,. \end{cases} \tag{2.5}$$

With the kernel probability distribution of Eq. (2.1), it is straightforward to obtain the size-abundance distribution of cosmic voids by applying:

$$\left.\frac{\mathrm{d}n}{\mathrm{d}\ln r}\right|_{linear} = \frac{f_{\ln\sigma}(\sigma)}{V(r)}\frac{\mathrm{d}\,\ln\sigma^{-1}}{\mathrm{d}\,\ln r}\,. \tag{2.6}$$

In order to derive the void size function in the non-linear regime, a conservation criterion has to be applied. The SvdW size function relies on the assumption that the total number of voids is conserved when going from linear to non-linear regimes[4]. While reaching shell-crossing, underdensities are expected to have expanded by a factor $a \propto (1+\delta_v^{NL})^{-1/3}$, thus a correction in radius by this factor is required:

$$\left.\frac{\mathrm{d}\,n}{\mathrm{d}\,\ln r}\right|_{\mathrm{SvdW}} = \left.\frac{\mathrm{d}\,n}{\mathrm{d}\,\ln(a\,r)}\right|_{linear}\,. \tag{2.7}$$

Jennings et al. (2013) argued that such a prescription is unphysical, since this leads to a volume fraction occupied by voids which is larger than the total volume of the Universe. They thus introduced a *volume conserving* model (hereafter Vdn model) in which it is instead assumed that the total volume occupied by cosmic voids is conserved in the transition to non-linearity. Nonetheless, when shell-crossing is reached, voids are thought to recover the overall expansion rate, and continue growing with the Hubble flow (Blumenthal et al. 1992; Sheth & van de Weygaert 2004). The conservation of volume is achieved by appling:

$$\left.\frac{\mathrm{d}\,n}{\mathrm{d}\,\ln r}\right|_{\mathrm{Vdn}} = \left.\frac{\mathrm{d}\,n}{\mathrm{d}\,\ln r}\right|_{\mathrm{linear}}\frac{V(r^L)}{V(r)}\frac{\mathrm{d}\,\ln r^L}{\mathrm{d}\,\ln r}\,, \tag{2.8}$$

where r^L indicates the radius predicted by linear theory (i.e. not accounting for the conversion factor a).

[4]This is the same assumption which is implicitly made to derive the halo mass function.

Several authors have tested the SvdW model on both simulated DM density fields and in mock halo catalogues, finding out that it systematically underpredicts the void comoving number density (see e.g. Colberg et al. 2005; Sutter et al. 2012; Pisani et al. 2015; Nadathur & Hotchkiss 2015). To overcome this mismatch, the underdensity threshold δ_v^L is commonly left as a free parameter, tuned on simulated halo catalogues. This severely affects the possibility of using the void size function as a cosmological probe. Jennings et al. (2013) have shown that the Vdn model does not require such a fine-tuning, as long as the void catalogue is properly cleaned from spurious voids. However, their results are limited to the case of cosmic voids detected from simulated DM distributions. We extend their study to the case of biased samples, such as mock DM halo catalogues, which are more representative of the real case.

Several authors have underlined that the tracer bias has to be taken into account in order to extract unbiased cosmological information from the number counts of cosmic voids detected in galaxy redshift surveys (see e.g. Pollina et al. 2019). In fact, the comoving number density and sizes of voids traced by the DM distribution are different from the ones of the voids traced by a biased DM halo density field.

In RM17 we introduced a simple modification of the void size function model that accounts for this effect. It follows from the results by Pollina et al. (2017), that showed that the DM density field within voids, $\delta_{v,\,\mathrm{DM}}^{NL}$, is linearly related to the density field traced by some biased tracers, $\delta_{v,\,\mathrm{tr}}^{NL}$:

$$\delta_{v,\,\mathrm{tr}}^{NL} = b\,\delta_{v,\,\mathrm{DM}}^{NL}\ . \tag{2.9}$$

Therefore, the threshold density to be used in the size function model of cosmic voids detected from biased tracers has to be corrected taking into account Eq. (2.9). Moreover, to recover the linear extrapolated value $\delta_{v,\,\mathrm{tr}}^{L}$, we use the fitting formula provided by Bernardeau (1994), namely

$$\delta^{NL} = (1 - \delta^L/\mathcal{C})^{-\mathcal{C}}\,, \tag{2.10}$$

with $\mathcal{C} = 1.594$. Eq. (2.10) provides a good fit, with errors below 0.2%, for standard cosmological models with any values of Ω_M and Ω_Λ, especially in the case of underdense regions ($\delta^L < 0$), and it is exact in the limit $\Omega_\mathrm{M} \to 0$, for $\Omega_\Lambda = 0$. Combining Eq. (2.10) with Eq. (2.9), it gives

$$\delta_{v,\,\mathrm{tr}}^{L} = \mathcal{C}\left[1 - (1 + b\,\delta_{v,\,\mathrm{DM}}^{NL})^{-1/\mathcal{C}}\right]. \tag{2.11}$$

Our recipe for a void size function that works regardless of whether the void catalogue is affected or not by a tracer bias is to use the value of δ_v^L found with Eq. (2.11) in the probability function given by Eq. (2.1).

2.1.2 Void finding and data reduction

There is not a general concordance in the definition of voids. Indeed, many different techniques have been proposed and applied over the years (see e.g. Colberg et al. 2008; Elyiv et al. 2015). A significant part of our work has been dedicated to cover this gap of

knowledge, through the development of a procedure to standardize the outcome of void finders of different types (see also RM17). In this work we use the public void finder VIDE[5] (Sutter et al. 2015, Void IDentification and Examination Toolkit), which implements an enhanced version of ZOBOV (Neyrinck 2008, ZOnes Bordering On Voidness) to construct voids with a watershed algorithm, whatever the nature of the tracers and independently of the presence of a biasing factor. The VIDE catalogues obtained are then *cleaned* with the pipeline developed and presented in RM17. This allows us to align the objects included in the void catalogue with the theoretical definition of cosmic voids used to derive the void size function (Section 2.1.1).

To implement our cleaning algorithm, we follow the procedure described in Jennings et al. (2013), which can be divided into three steps.

- *Step (i):* we consider two selection criteria to remove *non-relevant* objects. Underdensities whose effective radius is outside a user-selected range $[r_{min}, r_{max}]$ are erased from the input catalogue; if the effective radius, R_{eff}, is not provided, it is automatically set to r_{max}. We then remove underdensities with central density higher than $(1 + \delta_v^{NL})\overline{\rho}$, where δ_v^{NL} is a given non-linear underdensity threshold, and $\overline{\rho}$ is the mean density of the sample. We also included a third optional substep to deal with ZOBOV-like void catalogues. Specifically, we prune the underdensities that do not satisfy a statistical-significance criterion, $\Delta_v > \Delta_{v,0}$, where $\Delta_{v,0}$ is a multiple of the typical density contrast due to Poisson fluctuations. Threshold values for several N-σ reliabilities are reported in Neyrinck (2008). The user can select which of these three sub-steps to run.

- *Step (ii):* we rescale the underdensities to select shell-crossing regions. To satisfy this requirement, we impose the same density threshold as the one used by theoretical models (Eq. (2.1)). To this end, our algorithm reconstructs the density profile of each void candidate in the catalogue, exploiting the highly optimised and parallelised chain-mesh algorithm implemented in the CBL[6]. After this step, the void effective radius is computed as the largest radius from the void centre which encloses an underdensity equal to δ_v.

- *Step (iii):* finally, we check for overlaps. This last step is aimed to avoid double counts of cosmic voids. The collection of regions left from the previous steps are scanned one by one, checking for overlappings. When two voids do overlap, one of them is rejected, selecting the void with the smaller central density or, equivalently, with the larger density contrast (see e.g. Neyrinck 2008)[7]. Also this step is optimised by exploiting the chain-mesh algorithm.

[5] http://www.cosmicvoids.net

[6] The chain-mesh object implements a spatial locality algorithm to speed up neighbour searches. It is an hybrid between a linked list and an hash table and it relies on vectors of the C++ standard template library.

[7] The overlap-check acts on the rescaled voids by comparing either the central density or the internal density ratio (i.e. the ratio between the overall density contrast measured inside the void and its central density contrast) of two overlapping voids, discarding one of the two. Since in step *(ii)* of the algorithm we impose the overall density embedded by the voids to be equal to a fixed value, these two measures are

This cleaning method is necessary to match the distribution of voids measured in simulated catalogues with the predictions of the Vdn model. The rescaling part of the procedure (step *(ii)*) is based on the requirement that cosmic voids have to embed a density matching the shell-crossing prescription defined by the model. This is crucial in order to have a reliable description of their distribution in terms of spherical expansion. Specifically, spheres are grown around the centre of voids selected by the adopted void finder algorithm up to the scale inside which an a-priori chosen value of underdensity is reached. The latter value is typically chosen as the non-linear counterpart of the shell-crossing threshold, namely $\delta_v^{NL} \approx -0.795$ at redshift $z = 0$ for an EdS universe.

Even though it is reasonable to select this particular value as the critical threshold in both the model and the cleaning method, there are no restrictions in this sense and whatever value is acceptable, as long as it is used in both measuring (that is when applying the cleaning algorithm) and modelling the distribution (that is when building the modified model described in Section 2.1.1). This is a key issue: the choice of the underdensity threshold, that has to be embedded by the voids identified in whatever tracer distribution, must match the threshold δ_v^L in the model (Eq. (2.1)) used to predict their distribution. Given that when measuring the size distribution of cosmic voids we are dealing with a non-linear universe, while the theoretical model is derived from linear theory, a prescription to convert the threshold back and forth (such as the fitting relation from Bernardeau 1994) is necessary.

Impact of the cleaning steps

Figure 2.1 shows the void size distribution at the different steps of the cleaning algorithm. The original void sample has been detected with the VIDE void finder (Sutter et al. 2015) from a ΛCDM N-body simulation with 256^3 DM particles, and side length of 128 Mpc/h. The different symbols show the void size function after each step of the procedure, as indicated by the labels.

The first selection criterion removes the voids with a radius either smaller than twice the mean particle separation, that for the simulation used is 0.5 Mpc/h, or larger than half the box side length. The effect of this first step is negligible in the size range shown in the Figure. Then the algorithm removes all the voids with central density higher than $0.21 \cdot \overline{\rho}$ (second selection criterion of the first cleaning step).

The VIDE void radii enclose a compensation wall, that is, an overdense region at the border of the density basin where the matter evacuated from the inner part accumulates. The second cleaning step of our algorithm provides void radii that are always smaller

strictly related for our rescaled voids: voids with smaller central density have a larger internal density ratio and vice versa. In the original algorithm presented in RM17, choosing whether to discard the void with the larger central density (smaller internal density ratio) or the one with the smaller central density (larger internal density ratio) was left to the user. Selecting among these two choices reduces to deciding whether to keep the larger voids (which typically have a smaller central density, thus, a larger internal density ratio) or the smaller ones. Given that, as also stated in the original ZOBOV paper (Neyrinck 2008), voids with a larger internal density ratio are statistically more significant, we decided to remove this free parameter in the newer version of our algorithm which now automatically applies the criterion we dubbed *larger favoured* in the previous versions.

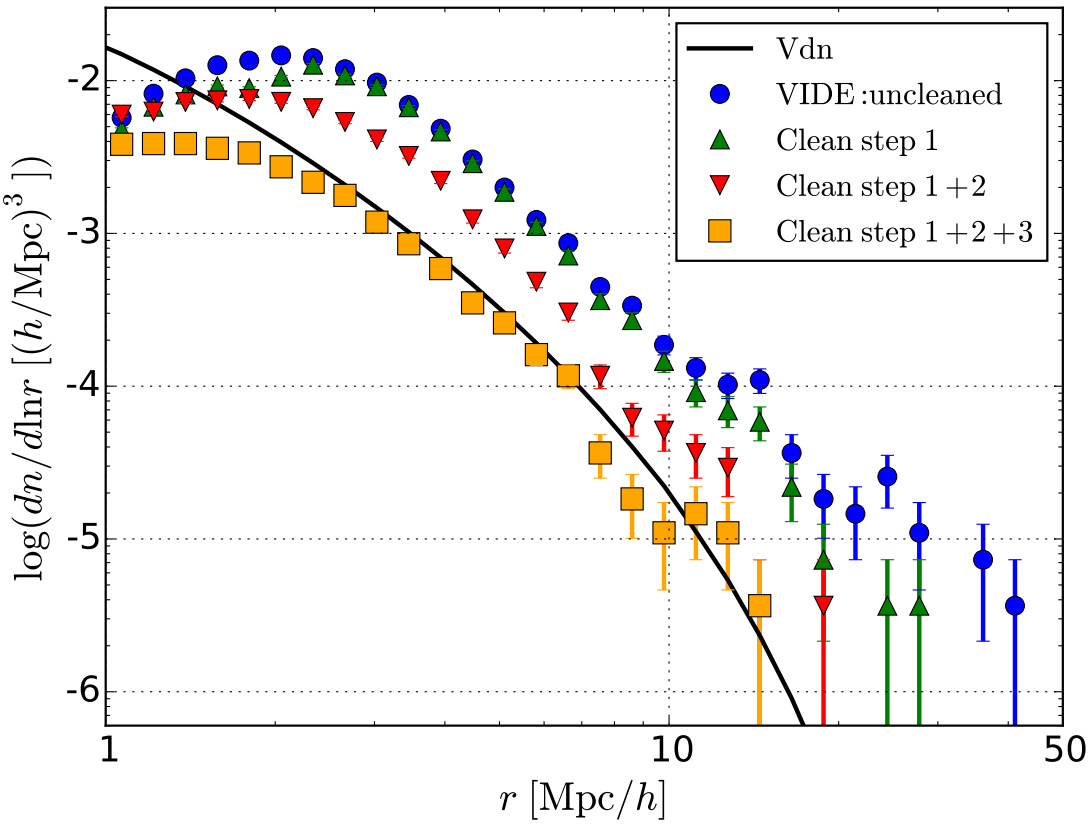

Figure 2.1: The void size function at the different steps of the cleaning procedure. The blue dots show the distribution of voids detected by VIDE from a ΛCDM N-body simulation. The green triangles are obtained by applying both the r_{min}-r_{max} criterion and the central density criterion (first step). The red upside-down triangles show the void distribution after having rescaled the voids (second step), while the orange squares are the final output of the cleaning algorithm after the overlapping criterion (third step) is applied. The black line shows the Vdn model prediction.

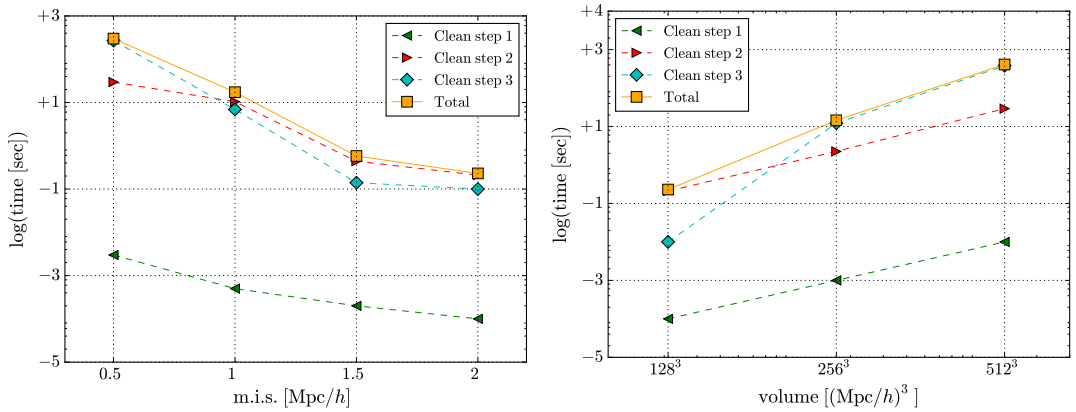

Figure 2.2: Time performances of the cleaning algorithm. The orange dots show the total amount of time spent by the code, while the other symbols refer to each step of the cleaning method, as indicated by the labels. *Left panel*: time at fixed volume ($V = 128^3$ (Mpc/h)^3) as a function of the mean inter-particle separation. *Right panel*: time at fixed resolution (m.i.s. = 2 Mpc/h) as a function of volume.

than the VIDE ones, thus substantially modifying the void distribution, which is shifted to smaller radii. Moreover, the total number count is decreased due to the removal of those underdensities whose radii have been shrunken to zero.

Figure 2.1 shows that all the steps of our cleaning procedure are required to match the theoretical predictions. The good agreement between the size distribution of voids in our cleaned catalogue and the Vdn model demonstrates that there is no need to fine-tune the model parameters (as has been done in previous works, e.g. Colberg et al. 2005; Sutter et al. 2012; Pisani et al. 2015; Nadathur & Hotchkiss 2015).

Time performances of the cleaning algorithm

We test the time performances of the presented cleaning algorithm as a function of both the density and volume of the input catalogue. The analysis has been performed on one single cluster node with 2 GHz of clock frequency and 64 Gb of RAM memory. The left panel of Figure 2.2 shows the computational time as a function of resolution, that is measured in terms of the mean inter-particle separation (m.i.s.). Both the time spent by each single step of the procedure and the global time are shown. The volume of the simulation used for this analysis is fixed at $V = 128^3$ (Mpc/h)^3. The first step of the cleaning method is the least time-consuming, while the other two have a similar impact: while the second step contributes the most to the total time at low resolution (high m.i.s.), it becomes less relevant with respect to the third step at higher resolution (small m.i.s.). The total amount of time spent by the algorithm, as well as that spent by each single step, increases almost exponentially with increasing resolution.

A similar behaviour is found as a function of volume. The performances of the algorithm at fixed resolution (m.i.s. = 2 Mpc/h) and with varying total volume are

Table 2.1: Fiducial cosmological parameters of the N-body simulations used in this work.

h_0	Ω_{CDM}	Ω_b	Ω_Λ	$\mathcal{A}_s$	n_s
0.7	0.226	0.0451	0.7289	$2.194 \cdot 10^{-9}$	0.96

shown in the right panel of Fig. 2.2. As in the fixed volume case, the first step is the least time-consuming. The third step is less time-consuming at small volumes than the second step, and it takes more time at larger volumes.

The outlined behaviour shows that the time spent by the algorithm increases exponentially with increasing number of voids. In the first case (Fig. 2.2, left panel), a larger number of voids are detected at small radii when the resolution is increased, while in the second case (Fig. 2.2, right panel) more voids are found at all radii when increasing the volume.

2.2 Results

We test the procedure outlined in Section 2.1 with both a set of DM-only N-body simulation snapshots, and a set of mock halo catalogues extracted by means of a Friends-of-Friends (FoF) algorithm and applying different mass selection cuts. Specifically, we consider a suite of N-body simulations of the standard ΛCDM cosmology performed with the C-GADGET module (Baldi et al. 2010) for different box sizes and particle numbers. Our largest box corresponds to the ΛCDM run of the L-CoDECS simulations (Baldi 2012), with a volume of $(1\,\mathrm{Gpc}/h)^3$ and a total number of 1024^3 particles, with a mass resolution of $\sim 6 \cdot 10^{10}\ M_\odot/h$. The smaller simulations have been run with the same code specifically for this work, and share with the largest box the same fiducial cosmological parameters (Komatsu et al. 2011, consistent with WMAP7 constraints, see Table 2.1), spanning a range of lower, equal, and higher particle mass and spatial resolutions, with the aim of testing the dependence of our analysis on these numerical parameters. The main properties of the different simulations are reported in Table 2.2.

As a first step, in Section 2.2.1 we validate our procedure on the four N-body simulations with 256^3 particles listed in Table 2.2. These simulations vary in mass resolution and mean inter-particle separation (m.i.s.). We gathered snapshots at four different redshifts ($z = 0, 0.5, 1, 1.5$) for each of these four simulations.

Afterwards, in Section 2.2.2 we extend our analysis to the case of biased tracers. The set of halo catalogues is extracted from a snapshot at redshift $z = 0$ of the simulation with box-side length $L_{\mathrm{box}} = 1000\ \mathrm{Mpc}/h$. We could not use the same simulations used in the validation part of the work because of the extremely low number of voids identified by VIDE in those small boxes.

Table 2.2: Our set of cosmological simulations with the corresponding relevant physical quantities: $L_{\rm box}$ is the simulation box-side length in units of [Mpc/h]; N_p is the total number of particles; m.i.s. is the mean inter-particle separation in the box in units of [Mpc/h]; in the last column, the particle mass in units of [$10^{10}\ M_\odot/h$] is reported.

$L_{\rm box}$ [Mpc/h]	N_p	m.i.s. [Mpc/h]	particle mass [$10^{10}\ M_\odot/h$]
1000	1024^3	≈ 1.00	5.84
500	256^3	≈ 2.00	56.06
256	256^3	1.00	7.52
128	256^3	0.50	0.94
64	256^3	0.25	0.12

2.2.1 Unbiased tracers

In order to validate our procedure, we first run the VIDE void finder on top of each of the 16 simulation snapshots described before with varying resolution and redshift . The resulting void catalogues are then cleaned with our pipeline. We evolve the shell-crossing threshold, $\delta_v^L = -2.71$, as

$$\delta_v^L(z) = \delta_v^L \cdot \frac{\mathcal{D}(z)}{\mathcal{D}(0)}\ , \tag{2.12}$$

by means of the growth-factor $\mathcal{D}(z)$ at redshift z. Applying the fitting formula given in Eq. (2.10), we obtain its non-linear extrapolation, $\delta_v^{NL}(z)$, and use this value as a threshold in the cleaning pipeline. Doing so, all the voids in our catalogues embed a fixed density

$$\rho_v = \langle\rho\rangle\ \left[1 + \delta_v^{NL}(z)\right]\,, \tag{2.13}$$

where $\langle\rho\rangle$ is the average matter density of the snapshot.

Firstly, we investigate the difference between the void density profiles before and after the application of our cleaning algorithm. It is generally believed that, in the standard ΛCDM cosmology, voids have a self-similar internal structure, which does not depend on the tracers used to define them (see e.g. Ricciardelli et al. 2013, 2014; Hamaus et al. 2014). The self-similarity assures that the internal structure of voids can be characterised by a single parameter, the void effective radius, $r_{\rm eff}$, which is the radius of a sphere embedding the same volume embedded by a given cosmic void. Despite the wide range of values this parameter covers, it is possible to average the internal density distribution to recover a common behaviour for voids with approximately the same $r_{\rm eff}$.

It has been noticed that, to recover self-similarity, some criteria have to be applied to the selection of objects to be classified as voids (Nadathur & Hotchkiss 2015). Nevertheless, the most crucial role seems to be played by the cut on the minimum density of the candidates, $\rho_{\rm min}$, which has to be sufficiently low (that is $\rho_{\rm min} \lesssim 0.3\langle\rho\rangle$). All voids undergoing our cleaning procedure with our chosen value of δ_v^{NL} satisfy this requirement

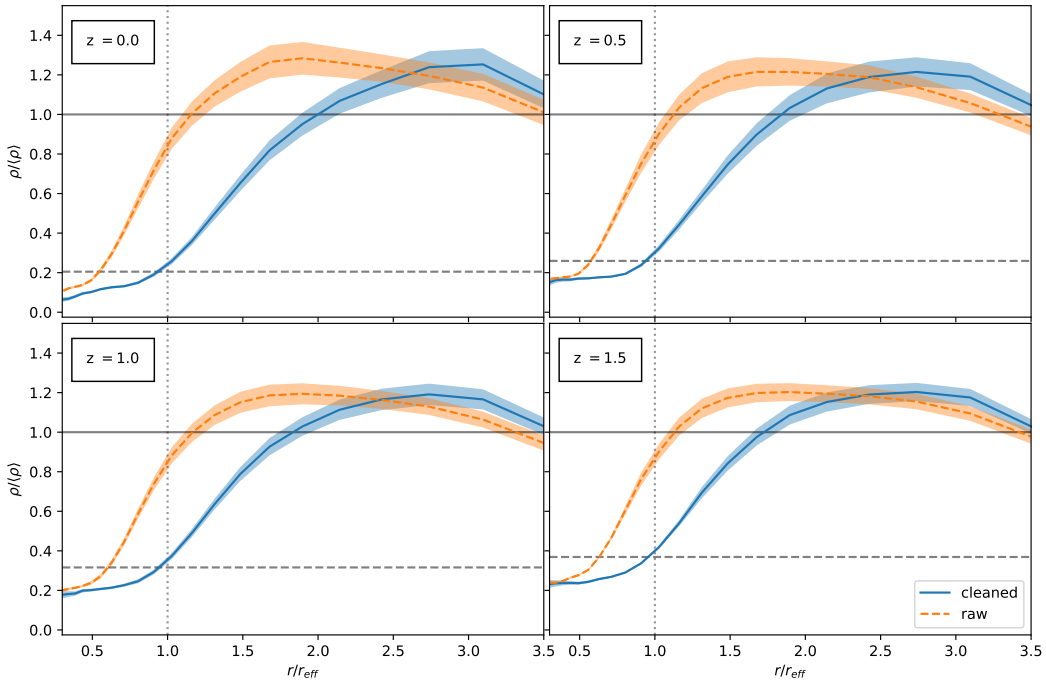

Figure 2.3: Comparison between the density profile within voids obtained with VIDE before (dashed orange line) and after (solid blue line) applying our cleaning algorithm. Each panel shows the result of averaging all the density profiles in a narrow bin of radii (with $1 < r_{\rm eff}[{\rm Mpc}/h] < 1.5$) and the 2σ confidence as a shaded region around the mean, at four different redshifts. The two horizontal lines show the position of the shell-crossing threshold (dashed line) and of the background density (solid line). The vertical dotted line highlights the position of the effective radius.

by construction. We are therefore confident that the objects we select should behave in a self-similar manner.

We select all the cosmic voids in the range of radii 1 Mpc/h $< r_{\rm eff} <$ 1.5 Mpc/h, from the catalogue with box side length $L_{\rm box} = 128$ Mpc/h, at four different redshifts. This is a compromise between having a large box (thus having more objects) and having a high resolution. Nonetheless, the results we are going to present are similar for the other simulations used in this section. The radius window has been chosen to have a sufficiently high number of objects with radii far enough from the lower limit imposed by the spatial resolution of the box that, for this catalogue, is 0.5 Mpc/h (see Table 2.2).

In Fig. 2.3 we show the stacked void profiles before and after cleaning the void catalogues with our algorithm. The stacked voids are all centred in the VIDE centres. We measure the cumulative number density of particles as a function of the distance from the centre, given in units of $r_{\rm eff}$, for each of the selected voids, particle-per-particle from 0.3 $r_{\rm eff}$ to 3.5 $r_{\rm eff}$. We divide this range of radii in 30 equally spaced bins and average the profiles. This procedure is repeated before and after applying the cleaning algorithm.

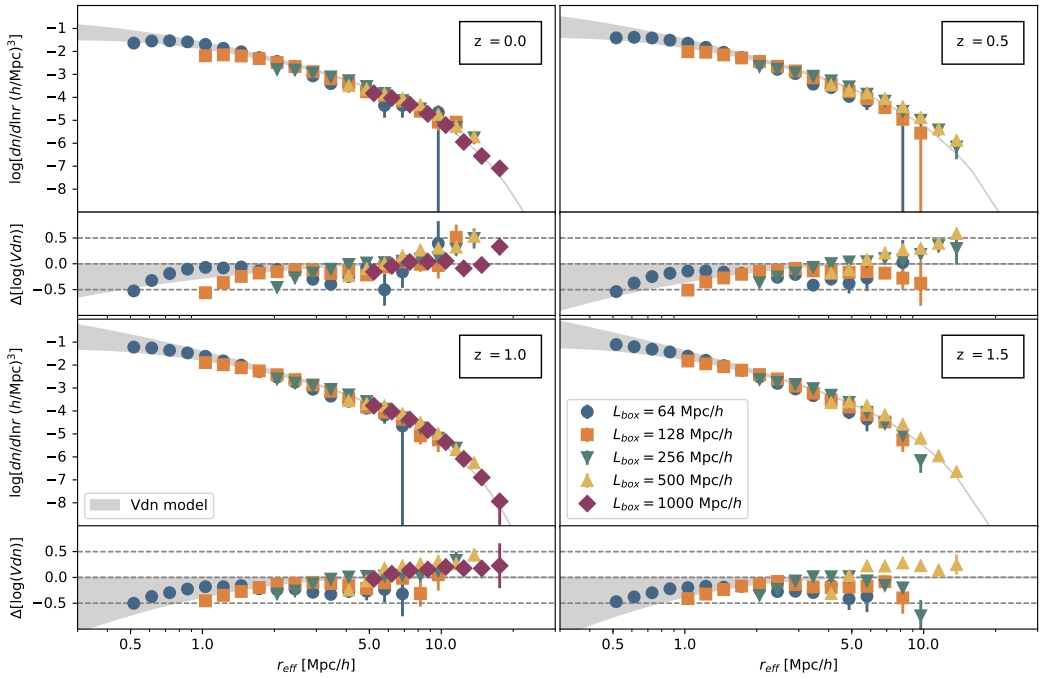

Figure 2.4: *Upper part of each panel*: Void size function prediction (grey shaded region) at four different redshifts (different panels) compared to the distribution of cosmic voids after applying the cleaning method described in Section 2.1.2 (different markers correspond to different simulations, the values of $L_{\rm box}$ in legend corresponding to those of Table 2.2). *Lower part of each panel*: Logarithmic differences between the measured distribution and the Vdn model prediction. For the largest simulation ($L_{\rm box} = 1000$ Mpc$/h$) we only have snapshots at $z = 0$ and $z = 1$, thus the corresponding abundances are missing from the $z = 0.5$ and $z = 1.5$ panels.

From Fig. 2.3 it is possible to appreciate how differently we define the ridges of a void: the orange dashed profiles show the profiles of voids identified with VIDE, while the blue solid ones are for the same voids after cleaning. The vertical dotted lines mark the void radius in both cases. We also show the background density level (solid horizontal black line) and the underdensity threshold of Eq. (2.12) computed for the different redshifts (dashed horizontal black line) as a reference. It can be noticed that, while VIDE void limits reside close to the compensation wall and embed a density that approaches the background, our voids are, by construction, defined in much deeper regions of the density field.

These observations have to be kept in mind to understand how the algorithm we apply to clean the cosmic void catalogues affects the void structure. Thanks to this modification we can be confident that the voids we are using to measure the size function are as close as possible to the objects for which we are able to construct the theoretical model.

Fig. 2.4 compares the measured and modelled void size functions, defined as the comoving number density of voids in logarithmic bins of effective radii, divided by the logarithmic extent of the bins. The four panels show the results at different redshifts. The different symbols in the upper part of each panel show the void size distributions measured in the simulation snapshots at different resolutions, while the grey shaded regions represent the model predictions. The model depends on two thresholds, the one of the shell-crossing, which is fixed at the value $\delta_v^L = -2.71$, corresponding to our choice of threshold in the void catalogue cleaning, and the overdensity one, δ_c, used to account for the *void-in-cloud* effect. There is a persisting uncertainty on the definition of the latter, since it is not clear if this effect becomes relevant at turn-around (corresponding to a density contrast of $\delta_c \approx 1.06$) or when the overdensity collapses (at a density contrast $\delta_c \approx 1.69$). Since in realistic occurrences we do not expect the spatial resolution of tracers to be high enough to inspect the radii range in which the void size function becomes sensitive to the *void-in-cloud* effect, we leave the overdensity threshold free to vary in the range $1.06 \leq \delta_c \leq 1.69$, thus the shaded region of Fig. 2.4.

We cut the distribution measured in each of the simulations at twice the m.i.s. of the given simulation box (see Table 2.2). To highlight the difference between the theoretical model predictions and the measured distributions, we show in the lower part of each panel the logarithmic difference between the simulation data and the model. Since voids are wide underdense regions, their measured properties and number density can be significantly affected by the spatial resolution of the sample. At scales comparable to the spatial resolution of the simulation, voids are not well represented by the mass tracers of the underlying density field. This causes a loss of power in the number counts, which can be noticed at the smallest scales in each panel of Fig. 2.4, in spite of the applied cut. At large scales, instead, the measurements are limited by the simulation box extension. Since our cleaning algorithm does not consider the periodic boundary conditions of the N-body simulations yet, the largest peripheral objects of the box may exceed the box boundaries. As a consequence, we cannot trust voids close to the boundaries of the simulation box, and therefore we choose to reject them from our analysis. Contrary to the limits at small radii, which cannot be overcome without increasing the resolution of the sample, this inaccuracy in the void counts at large radii could be faced with an upgrade of the void finding algorithm to account for the periodicity of the box. Nonetheless, since our final goal is to provide a framework to exploit the distributions of voids observed in real catalogues, this issue is not addressed in this work and we simply cut away these objects and reduce the total volume of the box consistently.

Fig. 2.4 shows that the Vdn model predictions are in overall good agreement with the results from N-body ΛCDM simulations, when cosmic voids are selected from unbiased matter tracers. We notice that the model predictions are closer to the simulation measurements at higher redshifts, in the full range of investigated radii. This might be due to the larger value of the underdensity threshold used for cleaning the catalogue: cosmic voids are less evolved at more distant cosmic epochs and have had less time to evacuate their interiors. A higher threshold means a higher density of tracers, thus, a higher resolution. Concerning the time evolution of the void size function, the density

of the larger voids decreases with increasing redshifts. On the other hand, the smaller scales are less affected. The Vdn model well predicts this behaviour.

2.2.2 Biased tracers

Having established the reliability of the Vdn model in predicting the number counts of cosmic voids detected in a simulated distribution of DM particles, we now extend our analysis to the case of biased tracers of the underlying DM density field. Our ultimate goal is to implement a model capable of predicting the void size distribution for observable tracers of the underlying density field, such as galaxies or clusters of galaxies. In this paper we start focusing on the simplest case of biased tracers, namely DM haloes.

It has been extensively demonstrated that voids in biased tracers of the underlying DM distribution are systematically larger than those predicted by the void size function models (Colberg et al. 2005; Sutter et al. 2012; Pisani et al. 2015; Nadathur & Hotchkiss 2015). Specifically, the typical void sizes increase with the minimum mass of the tracers, that is with their effective bias (Furlanetto & Piran 2006).

To infer cosmological constraints from the number count statistics of cosmic voids, the development of a reliable model independent of the tracers used to detect them is crucial. In this section we make use of the set of halo catalogues obtained by means of a FoF algorithm applied to the N-body simulation snapshot at redshift $z = 0$ with volume 1 $(\mathrm{Gpc}/h)^3$, whose properties are reported in Table 2.2.

These catalogues are obtained by cutting the original FoF samples at 3 different minimum masses: $M_{\mathrm{min}} = \{2 \cdot 10^{12}\ M_\odot/h, 5 \cdot 10^{12}\ M_\odot/h, 10^{13}\ M_\odot/h\}$ (see Table 2.3). These catalogues differ for the biasing factor with respect to the underlying DM distribution.

Our hypothesis is that the existing theoretical size function models fail in predicting the abundances of voids in the observed distributions of tracers because they implicitly assume that voids in biased tracers embed the same level of underdensity of the underlying DM distribution. This is far from being true and can be shown by a simple analysis of the density profiles around the void centres.

We obtain the VIDE void catalogues from each of the DM halo catalogues reported in Table 2.3, and clean them following the same procedure considered in Section 2.2.1 for the unbiased DM distributions. In particular, we keep the cleaning underdensity threshold parameter unchanged, meaning that we set it to $\delta_v^{NL} = -0.795$, as we have done in Section 2.2.1 for the unbiased catalogues at $z = 0$. This choice is dictated by the need of selecting *sufficiently underdense* regions, to be distinguishable from noise, but at the same time *sufficiently dense* in tracers not to incur in resolution issues. Nonetheless, we checked our final conclusions are robust with respect to the adopted threshold (Contarini et al. 2019).

We then divide our catalogue of cosmic voids into a set of logarithmically equally spaced bins within the radii ranges reported in Table 2.3, and compute the stacked density profile in each bin. Fig. 2.5 compares the stacked density profiles of the voids selected in DM halo catalogues with three different mass cuts, measured in the halo and DM distributions. The profiles stacked in each bin of radii varies from a minimum of 5 objects to a maximum of 300 objects and depends on the total number of voids found by

Table 2.3: Characteristic quantities for our set of halo catalogues. We report the minimum halo mass $M_{\rm min}$, the total number of haloes N_H, the corresponding mean numerical density $\langle n \rangle$, the m.i.s., the radii range used for profiling, the coefficients of the linear fit of Eq. (2.14) ($b_{\rm slope}$ and $c_{\rm offset}$) with the value of the residuals (res.), and the ratio between the profile measured in the underlying unbiased DM distribution and in the DM halo distribution ($(\delta_{\rm halo}/\delta_{\rm DM})(r_{\rm eff,h})$). All the catalogues have been obtained by applying different lower mass cuts (reported in the first column) to the FoF halo catalogue extracted from the simulation with 1024^3 particles (see Table 2.2).

$M_{\rm min}$ $M_\odot/h$	N_H 10^5	$\langle n \rangle$ $({\rm Mpc}/h)^{-3}$	m.i.s. Mpc/h	radii range Mpc/h	$b_{\rm slope}$	$c_{\rm offset}$	res.	$(\delta_{\rm halo}/\delta_{\rm DM})(r_{\rm eff,h})$
$2 \cdot 10^{12}$	15.8	$1.58 \cdot 10^{-3}$	8.58	$18 \leq r_{\rm eff} \leq 40$	1.408	0.022	0.005	1.406 ± 0.001
$5 \cdot 10^{12}$	7.6	$7.62 \cdot 10^{-4}$	10.95	$22 \leq r_{\rm eff} \leq 45$	1.488	0.013	0.002	1.522 ± 0.001
10^{13}	3.9	$3.94 \cdot 10^{-4}$	13.64	$27 \leq r_{\rm eff} \leq 50$	1.657	0.026	0.008	1.668 ± 0.003

the algorithm in the considered range. The number decreases going from smaller to larger radii. In order to have a reasonable convergence of the mean profile, we only consider the stacked profiles for which we have at least 30 objects to average. Fig. 2.5 shows also the background density and the shell-crossing underdensity level used to clean the catalogues which, by definition, crosses the profiles at $r = r_{\rm eff}$. The void density profiles measured from the distribution of DM haloes are significantly different with respect to those measured in the background matter density field, except at radii larger than about $2\,r_{\rm eff}$, approximately where the background density level is recovered. Specifically, the density measured within voids in the DM halo distribution is way deeper than that measured in the underlying DM density field.

The clear discrepancy between the halo-void density profiles and the underlying DM density profiles is the reason why we cannot use the Vdn model directly to predict void abundances in biased tracers: theoretical models of the void size function are based on the evolution of underdensities in the DM density field and their shape severely depends on the void density contrast. Ideally, we would like to recover the tracer density contrast, which corresponds to the shell-crossing value. However, since tracers are typically sparse, this is not practically viable: such a density contrast would be too low to be traced with enough statistics. What we can do instead is to fix the density threshold in the tracer distribution and recover the corresponding density contrast in the underlying DM distribution. Given the hypothesis that voids in tracers are centred in the same position of their DM counterparts, we search for the density contrast that voids identified in the tracer distribution reach in the underlying DM distribution. By taking the ratio between the two profiles, as suggested by Pollina et al. (2017), one can infer the relation between the density measured from biased tracers and the underlying unbiased DM density.

Fig. 2.6 shows the ratio between all the stacked profiles obtained for each catalogue, markers with error bars representing the uncertainty on the mean in the bin. Given the high level of uncertainty in the inner regions of the profiles, due to the sparsity of tracers, we exclude all the values with $r \leq 0.5\ r_{\rm eff}$. In agreement with the results by Pollina et al.

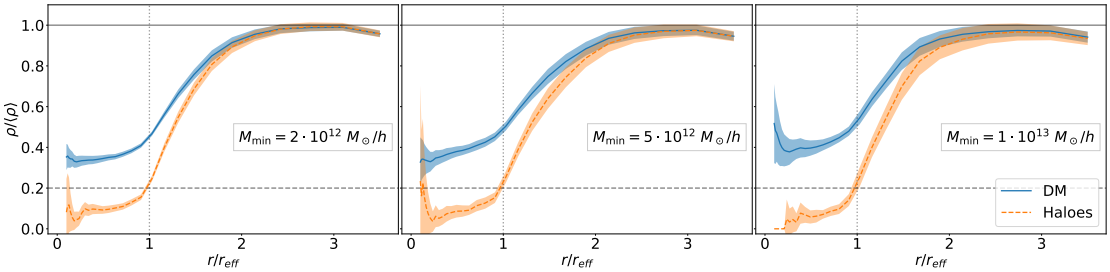

Figure 2.5: Stacked density profiles in different radii bins, for the three catalogues considered ($M_{\rm min} = 2 \cdot 10^{12} M_\odot/h$, $5 \cdot 10^{12} M_\odot/h$, $10^{13} M_\odot/h$, from left to right). Each panel shows the void stacked density profile. The dashed lines show the profiles measured in the DM halo distribution, while the solid lines represent the profile measured in the underlying DM distribution. The shaded bands around the mean profiles mark the 2σ confidence region. These density profiles have been measured using the cleaned VIDE void catalogues. The set of parameters used for the cleaning is the same one used for the unbiased DM voids in Section 2.2.1.

(2017), we find that the densities measured with the two different tracers, that is DM particles and haloes, are linearly related:

$$\delta_{\rm halo} = b_{\rm slope} \cdot \delta_{\rm DM} + c_{\rm offset} \,. \tag{2.14}$$

The best-fit values of the two free parameters, $b_{\rm slope}$ and $c_{\rm offset}$, are reported in Table 2.3. The relation between the best-fit values of $b_{\rm slope}$ and $c_{\rm offset}$ to the large-scale effective bias of the tracer samples, $b_{\rm eff}$, is addressed in Contarini et al. (2019).

The existing theoretical models predict the comoving number density of underdense regions characterised by a given embedded density contrast in the DM density field. Thus, to extend the models to the case of voids identified in the halo density field, we have to assess the density contrast in the underlying DM distribution. This can be obtained by taking the ratio between the density profiles measured in the two different tracers. Estimating it at exactly one effective radius allows us to associate the precise DM density contrast, required by the size function model for voids detected in distributions of biased tracers.

Fig. 2.7 shows the ratio between the density contrast we have requested in the cleaning algorithm for halo-voids ($\delta^{NL}_{\rm halo} = -0.795$) and the value of $\delta^{NL}_{\rm DM}(r_{\rm eff})$ measured from the three different catalogues. Of all the stacked profiles we have obtained, we use for this measurement only those which counted at least 30 objects in the radial bins to avoid issues with the convergence of the mean profile. The mean value of this ratio is therefore the conversion factor between the underdensity thresholds used in the detection and cleaning of cosmic voids in our halo catalogues and the non-linear counterpart of the DM underdensity threshold required by the void size function theoretical model (the values of the mean and standard deviations are reported in Table 2.3). In the considered range of effective radii, the mean value of this ratio is consistent with what can be inferred from a linear fit, as also shown in Fig. 2.7.

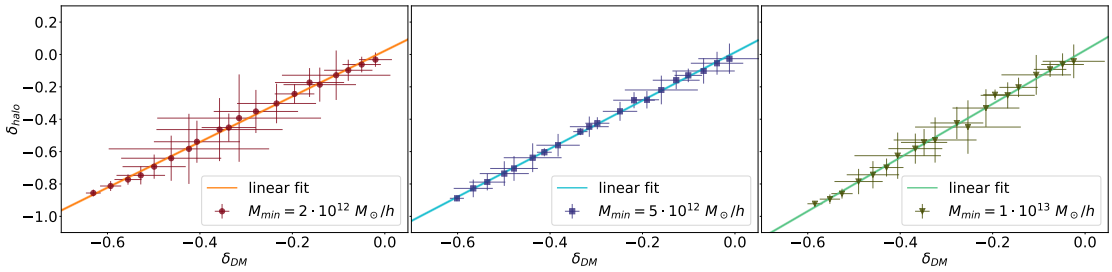

Figure 2.6: Density contrast in the unbiased tracer distribution, $\delta_{\rm DM}$, versus density contrast in the biased tracer distribution, $\delta_{\rm halo}$. Points are obtained gathering the measures of all the stacked profiles; the error bars represent a 1σ uncertainty on the mean values. The solid lines show the linear fit (the fitting parameters $b_{\rm slope}$ and $c_{\rm offset}$ are reported in Table 2.3), while the narrow shaded regions show the uncertainty on the best-fit values given in terms of the residuals (also reported in Table 2.3). Each panel shows the results for one of the halo catalogues of Table 2.3, from left to right: $M_{\rm min} = 2 \cdot 10^{12} M_\odot/h$, $5 \cdot 10^{12} M_\odot/h$, $10^{13} M_\odot/h$.

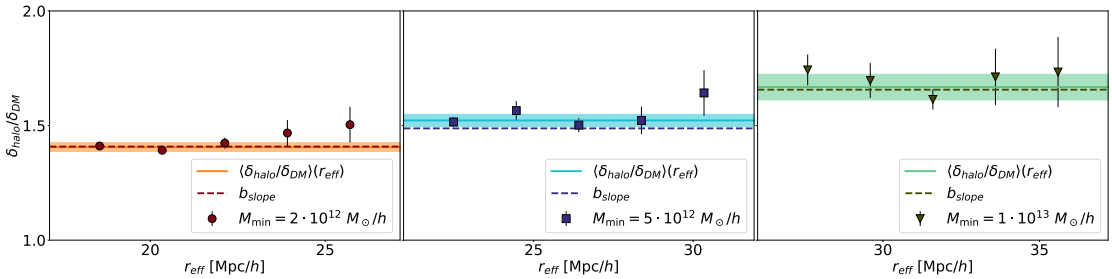

Figure 2.7: The ratio between $\delta_{\rm halo}$ and $\delta_{\rm DM}$ measured at $r = r_{\rm eff}$ for voids with different sizes. Each point is obtained by measuring $\delta_{\rm DM}(r_{\rm eff})$ in different effective radius bins. The dashed lines are the values of $b_{\rm slope}$ measured from the linear fit of Fig. 2.6, while the solid lines are the average values, with the corresponding shaded regions delimiting 2σ deviations from the mean. Each panel shows the results for one of the halo catalogues of Table 2.3, from left to right: $M_{\rm min} = 2 \cdot 10^{12} M_\odot/h$, $5 \cdot 10^{12} M_\odot/h$, $10^{13} M_\odot/h$.

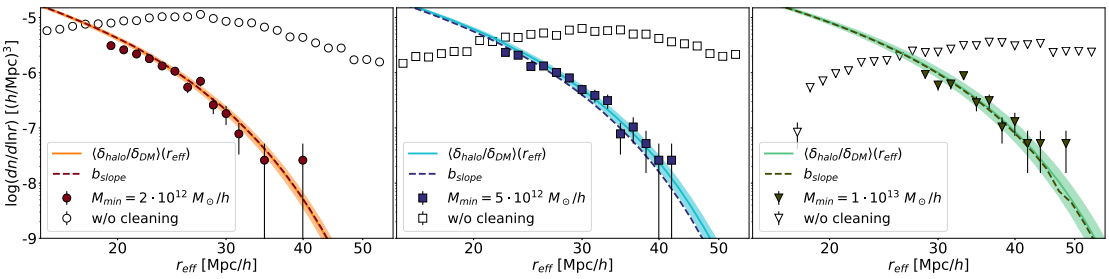

Figure 2.8: Size distribution of VIDE voids after the application of our cleaning prescriptions, measured in three different halo catalogues (markers). The shaded bands represent the 2σ confidence region around the mean values of the Vdn model modified to account for the different levels of DM underdensity enclosed by halo voids (solid lines). The dashed line shows the modified Vdn model in which, instead, we have used a threshold modified with the value of $b_{\rm slope}$. Each panel shows the results for one of the halo catalogues of Table 2.3, from left to right: $M_{\rm min} = 2 \cdot 10^{12} M_\odot/h$, $5 \cdot 10^{12} M_\odot/h$, $10^{13} M_\odot/h$. For comparison, in each plot we show the size-abundance distribution of cosmic voids measured in the VIDE void catalogue before the cleaning is applied (empty markers).

Using the Vdn size function model, we can describe the distribution of voids in the DM distribution underlying the distribution of tracers. If a cosmic void with radius $r_{\rm eff}$ embeds a density contrast $\delta^{\rm NL}_{v,{\rm DM}}$ in the DM density field, then the same radius $r_{\rm eff}$ will embed a density contrast $\delta^{\rm NL}_{v,{\rm tr}}$ in the tracer distribution, with $\delta^{\rm NL}_{v,{\rm tr}}$ and $\delta^{\rm NL}_{v,{\rm DM}}$ given by Eq. (2.9). Therefore, if $\delta^{\rm NL}_{v,{\rm tr}}$ is the threshold we use when cleaning the original void catalogue (Section 2.1.2), then the resulting void size distribution has to be modelled by a size function with an underdensity threshold given by:

$$\delta^{\rm L}_{v,{\rm DM}} = \mathcal{C}[1 - (1 + \delta^{\rm NL}_{v,{\rm DM}})^{-1/\mathcal{C}}], \tag{2.15}$$

where $\delta^{\rm NL}_{v,{\rm DM}}$ is the value shown in Fig. 2.7.

Fig. 2.8 shows the measured size functions of the voids detected in the three different DM halo catalogues, compared to the size function model described above. The shaded band in each plot shows the 2σ confidence region around the average value of $\delta^{\rm NL}_{v,{\rm DM}}$, obtained from the analysis of the void density profiles.

We find an excellent agreement between the measured and predicted void size functions in all the considered catalogues (almost all the points are within the 2σ confidence region), except at the smallest radii, where the spatial resolution of our catalogues does not allow us to have enough statistics of voids with $r_{\rm eff} \lesssim 2.5$ m.i.s., thus causing a loss of power. On the other hand, at large radii the size of the simulation box limits the possibility to obtain a reliable counting of large voids. To investigate the impact of the offset parameter, $c_{\rm offset}$, in Fig. 2.8 we also show the size function model obtained by considering only $b = b_{\rm slope}$ as the correction parameter. This model systematically underestimates the measured void size function, though the mismatch is within 2σ, considering the estimated uncertainties. This suggests that, even though it might not be the

case when working with cosmic void density profiles, as stated in Pollina et al. (2017), the value of the $c_{\rm offset}$ coefficient of the relation cannot be ignored when computing the size function of cosmic voids.

The shell-crossing threshold provides a reasonable value to define a cosmic void. However, the size function model is not forced to be constructed using this threshold. The model is in fact potentially capable of predicting the first-order statistics of density fluctuations whatever the threshold. In particular, in this section we choose to use the shell-crossing threshold ($\delta_v^{\rm NL}$) to clean the catalogues, selecting and rescaling the voids to the radius at which they reach this specific density contrast. Then, in the theoretical size function model (Eq. (2.1)), we substitute the threshold δ_v^L with the value that we measured from the stacked profiles ($\delta_{\rm DM}$), converted to its linear extrapolated counterpart by means of Eq. (2.10). On the one hand, this is required for being representative of the underdensities embedded by the voids traced by biased distributions. On the other hand, it also demonstrates that the use of the shell-crossing threshold is strictly required. The detection of voids is prone to the nature of the tracer used for sampling the underlying DM distribution. Our work highlights that modelling their statistics cannot be done trivially without accounting for this nature.

The results presented in this section rely on the existence of a linear relation between the density contrast embedded by voids in the tracer distribution and the density contrast of the underlying DM distribution. This relation, given by Eq.(2.14), has been calibrated on DM-only N-body simulations of the standard ΛCDM model (see also Pollina et al. 2017; Contarini et al. 2019).

2.2.3 Towards a cosmological exploitation of the void size function

In our following work (Contarini et al. 2019) we extend the validation of the model to higher redshift voids and investigate on the constraining power of the void size function.

Since in most cases it is not possible to infer the underlying DM distribution inside voids, calculating the ratio $(\delta_{\rm halo}/\delta_{\rm DM})(r_{\rm eff})$ (dubbed $b_{\rm punct}$ hereafter) is not a trivial task. For this reason we have first searched for a relation between $b_{\rm punct}$ and the effective bias, $b_{\rm eff}$, which can be independently estimated e.g. from clustering measurements. This relation is displayed in Fig. 2.9. As the figure shows, the data can be well fitted by a simple linear model.

However, the $b_{\rm punct}$ values estimated in the higher bias halo catalogues tend to systematically depart from the fit, at all redshifts. The reason of this slight deviation is related to the method used to find the void centres. In fact, if the detected voids are traced by too few tracers, the VIDE method might not be sufficiently accurate to localise their centres. Computing the spherically-averaged density contrast starting from a point that is not a local minimum of the density field causes systematic errors in the bias measurements. This is a natural consequence of the cleaning procedure: when rescaling the void radii, the selected threshold might be reached at smaller radii if overdense regions are included in the measurement, due to a bad centering. This is an issue especially for catalogues with a high mass selection.

As a possible strategy to alleviate the problem, we repeat our bias measurements

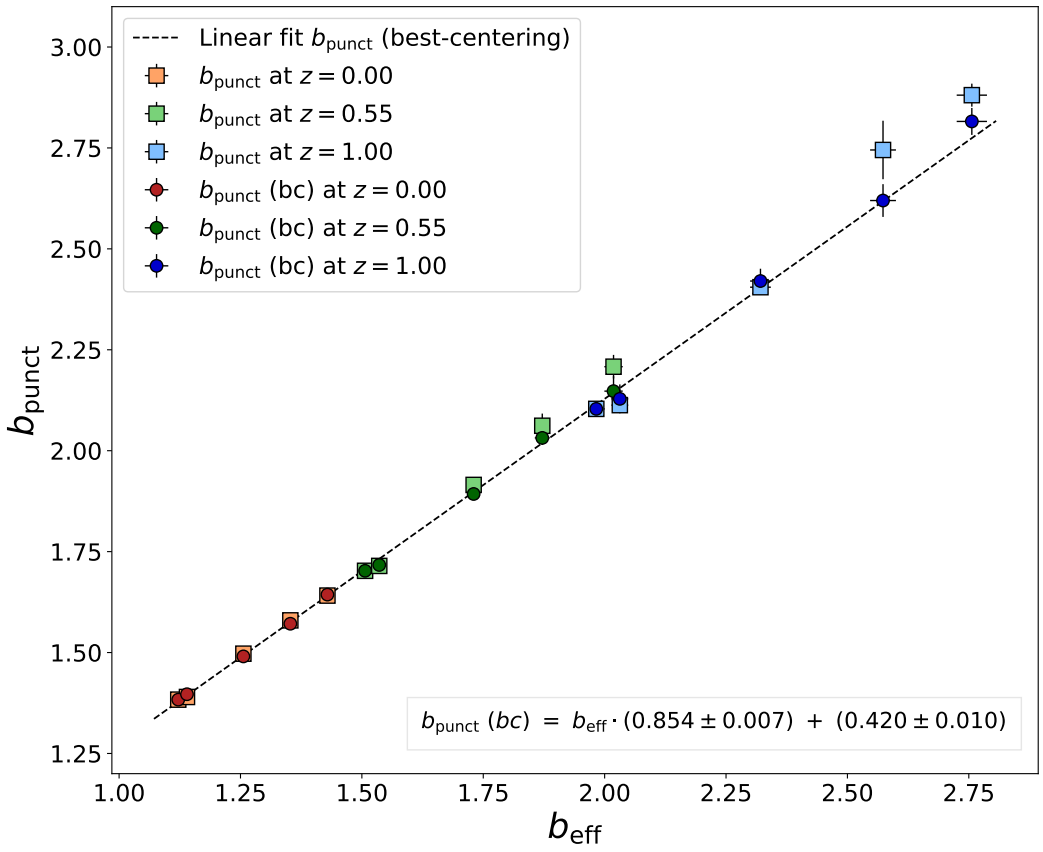

Figure 2.9: Relation between the effective bias ($b_{\rm eff}$) and the bias measured inside voids ($b_{\rm punct} \equiv (\delta_{\rm halo}/\delta_{\rm DM})(r_{\rm eff})$), at different redshifts. The points correspond to measures of $b_{\rm punct}$ from different halo catalogues (Table 2.3), with 1σ errors. The squares are the values of $b_{\rm punct}$, obtained with the method presented in the previous Sections, while the circles are estimated with the *best-centering* technique and correspond to $b_{\rm punct}$(bc). The black line is the linear fit of the $b_{\rm punct}$(bc) values. The best-fit parameters are reported in the label in the lower right corner.

using in all cases the centre positions of the voids detected in the catalogues with the lowest mass-cut. We will refer to this method as our *best-centering* technique, and we will call $b_{\rm punct}$ (bc) the corresponding bias. As shown in Fig. 2.9, these bias values (shown as coloured circles) are in better agreement with a linear model. Therefore we use them to calibrate the relation between the bias measured on large scales and the one computed inside cosmic voids, obtaining the following equation:

$$b_{\rm punct}\ ({\rm bc}) = b_{\rm eff} \cdot (0.854 \pm 0.007) + (0.420 \pm 0.010)\,. \tag{2.16}$$

This relation can be used to estimate the bias of the tracers inside voids from the effective bias of the whole tracer population. Hereafter, the bias obtained using Eq. (2.16) will be called $f(b_{\rm eff}) \equiv b_{\rm rel}$.

It is important to notice that the best-centering technique is not employable with real mocks, since in that case it is not possible to use more numerous tracers to improve the centre of a void. Nevertheless, in our work we choose to rely on this technique to obtain a better calibration of the relation between $b_{\rm punct}$ and $b_{\rm eff}$. Indeed, it is convenient to calibrate the latter with $b_{\rm punct}({\rm bc})$ to minimise the deviation of the data associated to the catalogues with higher mass selections from the linear fit. Using the best-centering technique to alleviate the problem of the sparsity of the tracers, we are able to extend our pipeline also to catalogues with lower spatial resolution.

We have used the above relation to further validate our void size function model against halo catalogues with redshift $z > 0$. The comparison shown in Fig. 2.10. Each row shows measures at different redshift ($z = 0$, 0.55, 1 from top to bottom) and each column corresponds to a different minimum halo mass (from left to right, as listed in Table 2.3). We show both the size function models obtained by rescaling with $b_{\rm punct}$ (green solid line) and $b_{\rm rel}$ (blue solid line), which appear to be fully consisten, especially at low-redshift and bias value. Discrepancies at higher redshift and bias appear to be not statistically significant. For comparison we also show the model obtained using the effective bias, $b_{\rm eff}$, which systematically under-predicts void counts at all redshift and biases.

In order to investigate on the constraining power of the void size function on the cosmological parameters, we perform a Bayesian statistical MCMC analysis of the measured void size function by sampling the posterior distribution of the parameters σ_8 and Ω_M. The coefficients of the linear relation reported in Eq. (2.16) are also left as free parameters, along with $b_{\rm eff}$. For these additional three parameters, we assume a Gaussian prior distribution centred with mean and standard deviation equal to the values estimated by our previous analysis.

The results for our simulated catalogues with treee different mass cuts and redshifts are reported in Fig. 2.11. The true values of the cosmological parameters are within 68% levels in all cases.

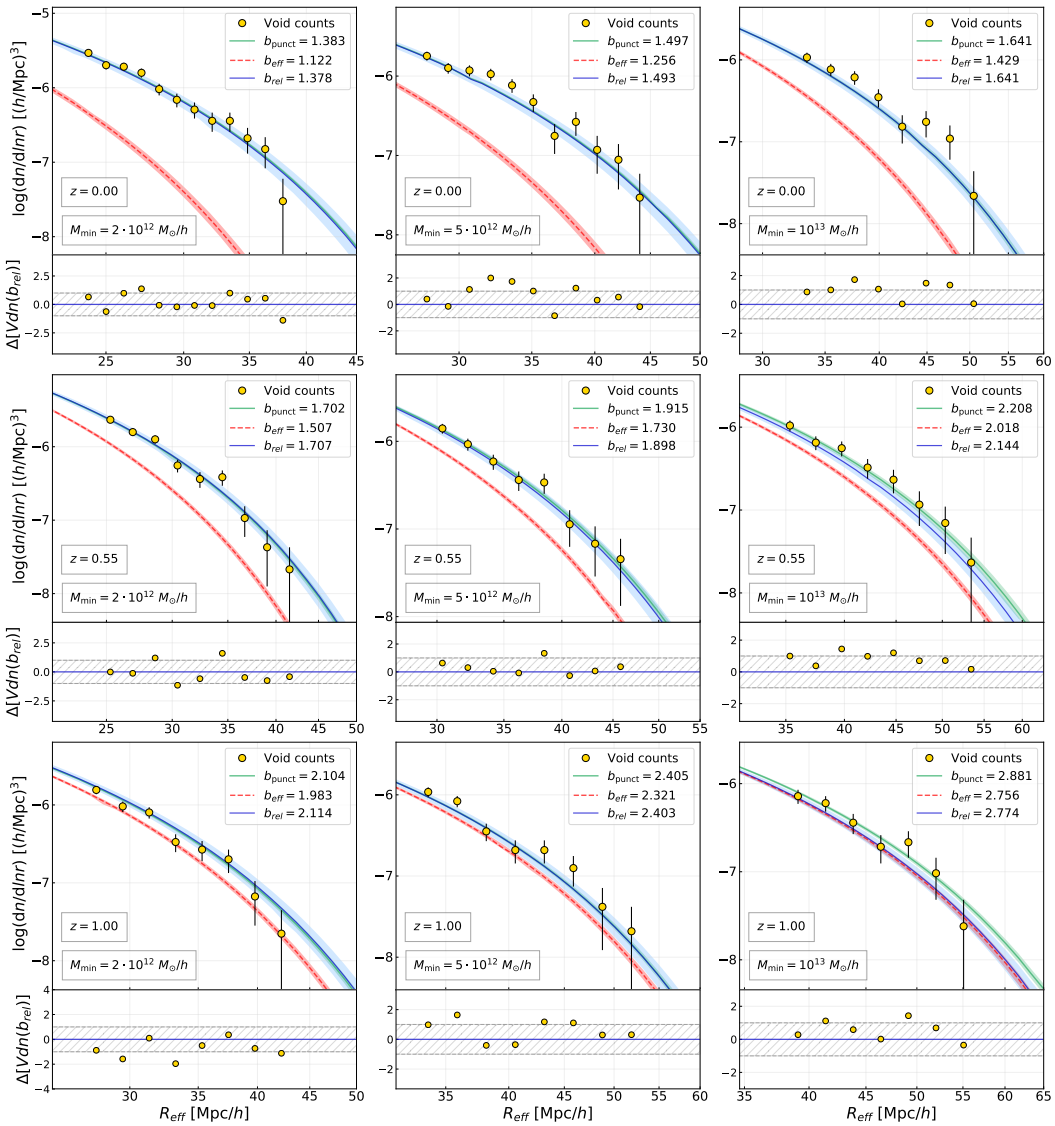

Figure 2.10: The measured size function of the voids (yellow dots) identified in the DM halo catalogues with $M_{\rm min} = 2{\cdot}10^{12}\ M_\odot/h$, $5{\cdot}10^{12}\ M_\odot/h$ and $10^{13}\ M_\odot/h$ (columns from left to right), at redshifts $z = 0$, $z = 0.55$, $z = 1$ (rows from top to bottom). Voids with $R_{\rm eff} < 2.5$ times the mean inter-particle separation are rejected from the analysis. *Upper sub-panels*: the blue dashed lines represent the void size function obtained by rescaling the Vdn model with $b_{\rm rel}$, that is the value of the bias computed from the relation shown in Fig. 2.9. The green solid lines show the model rescaled with the value of $b_{\rm punct}$. The red dashed lines represent the model rescaled with the effective bias, $b_{\rm eff}$. In all cases, the shaded areas indicate the variation of the model obtained applying 1σ errors on the value of the tracer bias. *Lower sub-panels*: the residuals of the void counts, computed as the ratio between the difference *data* $-$ *model* and the *data errors*, where the data are the measured void size function and the model is given by the re-parameterisation of the Vdn model with $b_{\rm rel}$. The grey dashed areas indicate the regions in which the discrepancy between the data and the model is within the data errors.

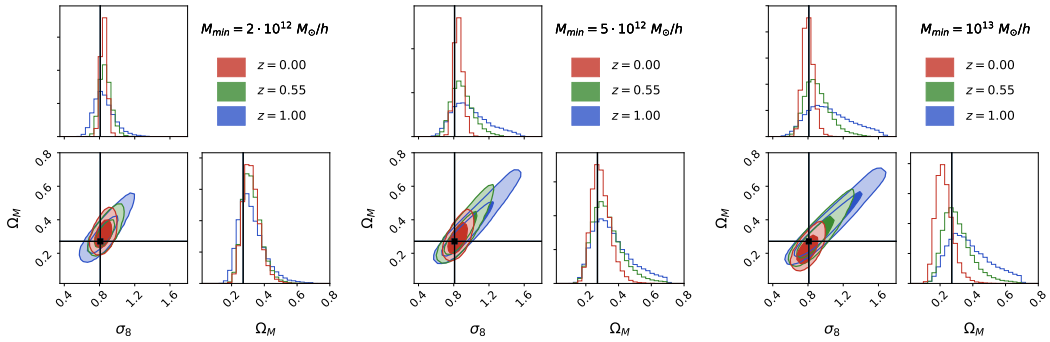

Figure 2.11: 68% and 95% contours in the σ_8 - $\Omega_{\rm M}$ plane, for the halo catalogues with $M_{\rm min} = 2 \cdot 10^{12}\ M_\odot/h$ (*left*), $5 \cdot 10^{12}\ M_\odot/h$ (*centre*), and $10^{13}\ M_\odot/h$ (*right*), obtained by re-parameterising the Vdn model directly with $b_{\rm eff}$, thus without converting this value by means of the Eq. (2.16). The colour of ellipses corresponds to different redshifts: red for $z = 0$, green for $z = 0.55$ and blue for $z = 1$. The prior distributions are uniform for σ_8 and $\Omega_{\rm M}$, and Gaussian for $b_{\rm eff}$. The histograms (top and bottom-right panels) show the marginalised posterior distributions of σ_8 and $\Omega_{\rm M}$, respectively. The black lines represent the true WMAP7 values ($\sigma_8 = 0.809$ and $\Omega_{\rm M} = 0.2711$).

2.3 Summary

The main goal of this work was to develop a theoretical model of the size function of cosmic voids detected from the distribution of biased tracers, extending the Vdn model, up to now validated for the DM distribution only.

The main steps of our analysis can be summarised as follows:

- We have developed a method to clean cosmic void catalogues, whose numerical implementation was presented in RM17. In particular, we have searched for the largest spherical regions embedding a mean density contrast of $0.2\ \overline{\rho}$, with $\overline{\rho}$ being the average density of the considered sample, ensuring that the identified regions do not overlap. The condition on the mean density ensures that the selected underdensities have actually passed through shell-crossing, while the second condition guarantees the volume conservation.

- We have validated the theoretical void size function using a set of DM catalogues extracted from N-body simulations with different characteristics in terms of box size and resolution. We have verified that the predictions of the Vdn model are reliable when modelling the distribution of cosmic voids in the DM density field using the void samples cleaned with our method. Indeed, we have found a good agreement between the model and the measurements for all the redshifts considered and on a large range of void radii.

- To extend the theoretical size function model to the case of voids extracted from the distribution of biased mass tracers, we have built void catalogues from mock

DM halo catalogues with different mass selections. Then we have computed the stacked density profiles, in a large range of radii, both in terms of density of haloes and of the underlying DM distribution.

- We have found a linear relation between the halo-profiles and the DM-profiles inside cosmic voids, in agreement with Pollina et al. (2017).
- By measuring the ratio between the density contrast of stacked profiles in haloes and DM at $r = r_{\rm eff}$, we have found the value of the multiplicative constant b, to be used to model the cosmic void size function in biased distributions (Eq. (2.9)), as the ratio between the density profiles measured in the tracers distribution and the density profiles measured in the underlying DM distribution, at one effective radius $r_{\rm eff}$.
- We have compared the proposed size function model to the ones measured from DM halo catalogues with different mass cuts, finding an excellent agreement.
- Finally, we have presented a linear fit between the effective bias and the bias within cosmic voids, and used it to investigate the constraining power of our framework.

Our overall conclusion is that the number count statistics of cosmic voids is completely determined by the cosmological model and by the relation between the density contrast of void tracers and DM *inside* the voids. This represents a key step towards the use of the void size function as a cosmological probe.

Part II

Painting Galaxies

Chapter 3

ScamPy: empirical modelling of galaxy catalogues

Cosmological N-body simulations are a fundamental tool for assessing the non-linear evolution of the large scale structure (LSS). With the increasing power of computational facilities, cosmological N-body simulations have grown in size and resolution, allowing to study extensively the formation and evolution of dark matter (DM) haloes (Springel et al. 2005; Boylan-Kolchin et al. 2009; Klypin et al. 2011; Angulo et al. 2012; Klypin et al. 2016). Our confidence on the reliability of these simulations stands on the argument that the evolution of the non-collisional matter component only depends on the effect of gravity and on the initial conditions. While for the first, we can rely on a solid theoretical background, with analytical solutions for both the classical gravitation theory and for a wide range of its modifications, for the latter, we have measurements at high accuracy (Planck Collaboration VI 2018) of the primordial power spectrum of density fluctuations.

The formation and evolution of the luminous component (i.e. galaxies and inter-galactic baryonic matter) are far from being understood at the same level as the dark matter. Several possible approaches have been attempted so far to asses this modeling issue, which can be divided into two main categories. On one side, *ab initio* models, such as N-body simulations with a full hydrodynamical treatment and semi-analytical models, that should incorporate all the relevant astrophysical processes, are capable of tracing back the evolution in time of galaxies within their DM host haloes (see Somerville & Davé 2015; Naab & Ostriker 2017, for reviews).

On the other side, *empirical* (or *phenomenological*) models are designed to reproduce observable properties of a target (observed) population of objects at a given moment of their evolution (see, e.g., Wechsler & Tinker 2018, for a review). This latter class of methods is typically cheaper in terms of computational power and time required for running. The development of an approach to model the luminous component without prior assumptions on the baryon physics have emerged from the advent of large galaxy surveys (York et al. 2000; Colless et al. 2001; Lilly et al. 2007; Driver et al. 2011; Grogin et al. 2011; McCracken, H. J. et al. 2012).

Building an empirical model of galaxy occupation requires to define the hosted-

object/hosting-halo connection for associating to the underlying DM distribution its baryonic counterpart. This has been achieved by exploiting several approaches that span from the classical mass-based methods, such as the Halo Occupation Distribution (HOD) scheme (Peacock & Smith 2000; Seljak 2000; White 2001; Berlind & Weinberg 2002; Yang et al. 2003; Zehavi et al. 2004; Zheng et al. 2005; Tinker et al. 2005; Brown et al. 2008; Leauthaud et al. 2012) or the sub-halo abundance matching (SHAM) scheme (Mo & White 1996; Wechsler et al. 1998; Vale & Ostriker 2004; Conroy et al. 2006; Wang et al. 2006, 2007; Moster et al. 2010; Behroozi et al. 2010; Guo et al. 2010; Trujillo-Gomez et al. 2011), to more sophisticated parameterisations that follow the halo evolution in time (Conroy & Wechsler 2009; Yang et al. 2012; Behroozi et al. 2013a; Moster et al. 2013, 2018; Zhu et al. 2020) also allowing for adaptive complexity (Behroozi et al. 2019). At the same time the number of observables that can be generated with such methods increased including galaxy luminosity (e.g. Rodríguez-Puebla et al. 2017; Moster et al. 2018; Somerville et al. 2018), gas (e.g. Popping et al. 2015), metallicity (e.g. Rodríguez-Puebla et al. 2016) and dust (e.g. Imara et al. 2018).

Given their capability to target galaxy formation without biasing the model with baryon physics uncertainties, empirical models complement and help to constrain *ab initio* models. The power of empirical approaches comes from the possibility to infer the DM density field from observations of the biased luminous component (Monaco & Efstathiou 1999; Jasche & Lavaux 2019; Kitaura et al. 2019). The mock catalogues obtained can be used to build precise co-variance matrices in preparation for assessing the uncertainties on cosmological parameters estimates, that will be inferred from next generation LSS observational campaigns, such as DESI (Levi et al. 2013) and Euclid (Amendola et al. 2018). Via the usage of empirical models it is possible to considerably speed up the construction of mock catalogues and are the natural framework for forward modeling of the LSS observable properties (see, e.g. Nuza et al. 2014; Leclercq et al. 2015; Kitaura et al. 2019). Furthermore, where *ab initio* models have struggled to obtain tight parameter constraints (e.g., on the mechanism for galaxy quenching), empirical models are capable of revealing possibly new un-expected physics (see, e.g. Behroozi et al. 2012; Behroozi & Silk 2015).

Ab initio approaches are tuned to reproduce the LSS of the Universe at the present time, and therefore their reliability in the high redshift regime has to be proven. On the other hand, empirical models are by design particularly suitable for addressing the modelling of the high redshift Universe, but they rely on the availability of high redshift observations of the population to be modelled.

Our motivation for the original development of the Application Programming Interface (API) we present in this work is to study a particular window in the high redshift Universe. Specifically, our aim is twofold: *i*) provide a physically robust and efficient way of modelling galaxy populations in the high redshift Universe from a DM-only N-body simulation; *ii*) test applications, such as the modelling of the distribution of the first sources that started to shed light on the neutral medium, triggering the process called *Reionization*. We expect that this tool could have further applications, especially in the context of cross-correlation of different tracers and/or diffuse backgrounds.

ScamPy provides a python interface that uses the Sub-halo Clustering and Abundance Matching (SCAM) prescription for "painting" galaxies on top of DM-only simulations. The SCAM algorithm is an extension of the classical HOD for defining the galaxy-halo connection. This class of methods is widely used in the scientific community but specialised software exists only within larger software packages (e.g. the Halotools package from Hearin et al. 2017, which is part of the Astropy library collection). Our intent is to provide the user with a light and versatile interface able to provide perfomances and extensibility with as little dependence to external software as possible.

We have carefully designed the software to exploit the best features of Python and c++ language. Our intent was not only to achieve high performances of our code but also to make it more accessible, to ease cross-platform installation, and to generally set-up a flexible tool. Since the API we present has been designed to be easily extensible, in the future we will also be able to evolve our current research towards novel directions. Furthermore, this effort would hopefully also encourage new users to adopt our tool. As much as experiments are accurately designed to have the longest life-span possible, we have taken care of designing our software for a long term use.

The API relies on an optimized c++ core implementation of the most computationally expensive sections of the algorithm. This allows, on the one hand, to exploit the performances of a compiled language. On the other hand, it overcomes the limit on the usage of multi-threading for shared memory parallelisation, as otherwise imposed by the python standard library.

ScamPy embeds two main functionalities: on the one side, it is designed for handling and building mock-galaxy catalogues, based on an user-defined parameterisation. On the other side, it provides an extremely efficient implementation of the halo-model, which is used to infer the parameters required by the SCAM algorithm.

We provide a framework for loading a DM halo/sub-halo hierarchy, where the haloes are obtained by means of a friends-of-friends algorithm run on top of cosmological N-body simulations, while the substructures are identified using the SUBFIND algorithm (Springel et al. 2001). Nonetheless, thanks to its extensible design, adapting the API for working with simulations obtained by means of approximated methods, such as COLA (Tassev et al. 2013) or PINOCCHIO (Monaco et al. 2002), would be straightforward.

Once the ScamPy parameters, which regulate the occupation of structures, have been set, we can easily produce the output mock-catalogue by calling the dedicated functions from the same framework we used for loading the DM halo/sub-halo hierarchy.

3.1 Sub-halo Clustering & Abundance Matching

Our approach for the definition of the hosted-object/hosting-halo connection is based on the Sub-halo Clustering and Abundance Matching (SCAM) technique (Guo et al. 2016). With the standard HOD approach, hosted objects are associated to each halo employing a prescription which is based on the total halo mass, or on some other mass proxy (e.g. halo peak velocity, velocity dispersion). On the other side, the SHAM assumes a monotonic relation between some observed object property (e.g. luminosity or stellar mass of a

galaxy) and a given halo property (e.g. halo mass). While the first approach is capable of, and extensively used for, reproducing the spatial distribution properties of some target population, the second is the standard for providing plain DM haloes and sub-haloes with observational properties that would otherwise require a full-hydrodynamical treatment of the simulation from which these are extracted.

The SCAM prescription aims to combine both approaches, providing a parameterised model to fit both some observable abundance and the clustering properties of the target population. The approach is nothing more than applying HOD and SHAM in sequence:

1. the occupation functions for central, $N_{\rm cen}(M_h)$, and satellite galaxies, $N_{\rm sat}(M_h)$, depend on a set of defining parameters which can vary in number depending on the shape used. These functions depend on a proxy of the total mass of the host halo. We sample the space of the defining parameters using a Markov-Chain Monte-Carlo (MCMC) to maximize a likelihood built as the sum of the χ^2 of the two measures we want to fit, namely the two-point angular correlation function at a given redshift, $\omega(\theta, z)$, and the average number of sources at a given redshift, $n_g(z)$:

$$\log \mathcal{L} \equiv -\frac{1}{2}\left(\chi^2_{\omega(\theta,z)} + \chi^2_{n_g(z)}\right) , \tag{3.1}$$

   The analytic form of both $\omega(\theta, z)$ and $n_g(z)$, depending on the same occupation functions $N_{\rm cen}(M_h)$ and $N_{\rm sat}(M_h)$, can be obtained with the standard halo model (see Cooray & Sheth 2002, for a review), which we describe in detail in Section 3.1.1. How these occupation functions are used to select which sub-haloes will host our mock objects is reported in Section 3.2.1.

2. Once we get the host halo/subhalo hierarchy with the abundance and clustering properties we want, as guaranteed by Eq. (3.1), we can apply our SHAM algorithm to link each mass (or, equivalently, mass-proxy) bin with the corresponding luminosity (or observable property) bin.

When these two steps have been performed, the mock-catalogue is built.

3.1.1 The halo model

The modern formulation of the halo-model theory (see Cooray & Sheth 2002, for a review) provides a halo-based description of non-linear gravitational clustering that is widely used in literature to infer the underlying DM statistical properties at both low and high redshift. The key assumption of this model is that the number of galaxies, N_g, in a given dark matter halo only depends on the halo mass, M_h. Specifically, if we assume that $N_g(M_h)$ follows a Poisson distribution with mean proportional to the mass of the halo M_h, we can write

$$\langle N_g \rangle (M_h) \propto M_h \tag{3.2}$$

$$\langle N_g (N_g - 1) \rangle (M_h) \propto M_h^2 \tag{3.3}$$

From these assumptions it is possible to derive correlations of any order as a sum of the contributions of each possible combination of objects identified in single or in multiple haloes. To get the models required by Eq. (3.1) we only need the 1-point and the 2-point statistics. We derive the first as the mean abundance of objects at a given redshift. The average mass density in haloes at redshift z is given by

$$\overline{\rho}(z) = \int M_h \; n(M_h, z) \; \mathrm{d}M_h \tag{3.4}$$

where $n(M_h, z)$ is the halo mass function. With Eq. (3.2) we can then define the average number of objects at redshift z, hosted in haloes with mass $M_{\rm min} \leq M_h \leq M_{\rm max}$, as

$$n_g(z) \equiv \int_{M_{\rm min}}^{M_{\rm max}} \langle N_g \rangle (M_h) \; n(M_h, z) \; \mathrm{d}M_h \; . \tag{3.5}$$

Deriving the 2-point correlation function, $\xi(r, z)$, would require to treat with convolutions, we therefore prefer to obtain it by inverse Fourier-transforming the non-linear power spectrum $P(k, z)$, whose derivation can instead be treated with simple multiplications:

$$\xi(r, z) = \frac{1}{2\pi^2} \int_{k_{\rm min}}^{k_{\rm max}} \mathrm{d}k k^2 P(k, z) \frac{\sin(kr)}{kr} \; . \tag{3.6}$$

$P(k, z)$ can be expressed as the sum of the contribution of two terms:

$$P(k, z) = P_{\rm 1h}(k, z) + P_{\rm 2h}(k, z) \; , \tag{3.7}$$

where the first, dubbed *1-halo term*, results from the correlation among objects belonging to the same halo, while the second, dubbed *2-halo term*, gives the correlation between objects belonging to two different haloes.

The 1-halo term in real space is the convolution of two similar profiles of shape

$$\begin{aligned} \widetilde{u}(k, z|M_h) = & \frac{4\pi\rho_s r_s^3}{M_h} \Big\{ \sin(kr_s) \big[\mathrm{Si}((1+c)kr_s) - \mathrm{Si}(kr_s) \big] \\ & - \frac{\sin(ckr_s)}{(1+c)kr_s} - \cos(kr_s) \big[\mathrm{Ci}((1+c)kr_s) - \mathrm{Ci}(kr_s) \big] \Big\} \; , \end{aligned} \tag{3.8}$$

where c is the halo concentration, ρ_s and r_s are, respectively, the scale density and radius of the NFW profile and the sine and cosine integrals are defined as

$$\mathrm{Ci}(x) = \int_t^\infty \frac{\cos t}{t} \, \mathrm{d}t \;\; \text{and} \;\; \mathrm{Si}(x) = \int_0^x \frac{\sin t}{t} \, \mathrm{d}t \; . \tag{3.9}$$

Eq. (3.8) provides the Fourier transform of the dark matter distribution within a halo of mass M_h at redshift z. Weighting this profile by the total number density of pairs, $n(M_h)(M_h/\overline{\rho})^2$, contributed by haloes of mass M_h, leads to the expression for the 1-halo term:

$$\begin{aligned} P_{\rm 1h}(k, z) & \equiv \int n(M_h, z) \left(\frac{M_h}{\overline{\rho}} \right)^2 \left| \widetilde{u}(k, z|M_h) \right|^2 \mathrm{d}M_h = \\ & = \frac{1}{n_g^2(z)} \int_{M_{\rm min}}^{M_{\rm max}} \langle N_{\rm g}(N_{\rm g} - 1) \rangle (M_h) \; n(M_h, z) \; \left| \widetilde{u}(k, z|M_h) \right|^2 \mathrm{d}M_h , \end{aligned} \tag{3.10}$$

with $n(M_h, z)$ the halo mass function for host haloes of mass M_h at redshift z and where, in the second equivalence, we used Eqs. (3.3) and (3.5) to substitute the ratio $\left(M_h/\overline{\rho}\right)^2$.

The derivation of the 2-halo term is more complex and for a complete discussion the reader should refer to Cooray & Sheth (2002). Let us just say that, for most of the applications, it is enough to express the power spectrum in its linear form. Corrections to this approximation are mostly affecting the small-scales which are almost entirely dominated by the 1-halo component. This is mostly because the 2-halo term depends on the biasing factor which on large scales is deterministic. We therefore have that, in real-space, the power coming from correlations between objects belonging to two separate haloes is expressed as the product between the convolution of two terms and the biased linear correlation function (i.e. $b'_h(M'_h)b_h(M''_h)\xi_{\rm lin}(r,z)$). The two terms in the convolution provide the product between the Fourier-space density profile $\widetilde{u}(k,z|M_h)$, weighted by the total number density of objects within that particular halo (i.e. $n(M_h)(M_h/\overline{\rho})$). In Fourier space, we therefore have

$$\begin{aligned} P_{\rm 2h}(k,z) \equiv & \int n(M'_h)\,\frac{M'_h}{\overline{\rho}}\,\widetilde{u}(k,z|M'_h)\,b(M'_h)\;{\rm d}M'_h \\ & \int n(M''_h)\,\frac{M''_h}{\overline{\rho}}\,\widetilde{u}(k,z|M''_h)\,b(M''_h)\,P_{\rm lin}(k,z)\;{\rm d}M''_h = \\ = \frac{P_{\rm lin}(k,z)}{n_g^2(z)} & \left[\int_{M_{\rm min}}^{M_{\rm max}} \langle N_{\rm g}\rangle(M_h)\,n(M_h)\,b(M_h,z)\,\widetilde{u}_h(k,z|M_h)\;{\rm d}M_h\right]^2 \end{aligned} \tag{3.11}$$

with $b(M_h)$ the halo bias and $P_{\rm lin}(k,z)$ the linear matter power spectrum evolved up to redshift z. For going from the first to the second equivalence we have to make two assumptions. First we assume self-similarity between haloes. This means that the two nested integrals in ${\rm d}M'_h$ and ${\rm d}M''_h$ are equivalent to the square of the integral in the rightmost expression. Secondly, we make use of Eqs. (3.2) and (3.5) to substitute the ratio $M_h/\overline{\rho}$.

The average number of galaxies within a single halo can be decomposed into the sum

$$\langle N_g\rangle(M_h) \equiv N_{\rm cen}(M_h) + N_{\rm sat}(M_h) \tag{3.12}$$

where $N_{\rm cen}(M_h)$ is the probability to have a central galaxy in a halo of mass M_h, while $N_{\rm sat}(M_h)$ is the average number of satellite galaxies per halo of mass M_h. These two quantities are precisely the occupation functions we already mentioned in Section 3.1. Given that no physics motivated functional form exists for $N_{\rm cen}(M_h)$ and $N_{\rm sat}(M_h)$, usually, they are parameterised. By tuning this parameterisation we obtain the prescription for defining the hosted-object/hosting-halo connection.

With the decomposition of Eq. (3.12), we can approximate Eq. (3.3) to

$$\begin{aligned} \langle N_{\rm g}(N_{\rm g}-1)\rangle(M_h) \approx \langle N_{\rm cen}N_{\rm sat}\rangle(M_h) + 2\,\langle N_{\rm sat}(N_{\rm sat}-1)\rangle(M_h) \approx \\ \approx N_{\rm cen}(M_h)\,N_{\rm sat}(M_h) + N^2_{\rm sat}(M_h) \end{aligned} \tag{3.13}$$

Thus we can further decompose the 1-halo term of the power spectrum as the combination of power given by *central-satellite* couples (cs) and *satellite-satellite* couples (ss):

$$P_{1h}(k,z) \approx P_{cs}(k,z) + P_{ss}(k,z)\;, \tag{3.14}$$

When dealing with observations, it is often more useful to derive an expression for the projected correlation function, $\omega(r_p, z)$, where r_p is the projected distance between two objects, assuming flat-sky. From the Limber approximation (Limber 1953) we have

$$\begin{aligned}\omega(r_p, z) = \mathcal{A}\big[\xi(r, z)\big] = \mathcal{A}\big\{\mathcal{F}\big[P(k, z)\big]\big\} = \mathcal{H}_0\big[P(k, z)\big] = \\ = \frac{1}{2\pi}\int k\ P(k, z)\ J_0(r_p k)\ \mathrm{d}k\end{aligned} \tag{3.15}$$

where $J_0(x)$ is the 0^{th}-order Bessel function of the first kind. Reading the expression above from left to right, we can get the projected correlation function by Abel-projecting the 3D correlation function $\xi(r, z)$. From the definition in Eq. (3.6), $\omega(r_p, z)$ is therefore obtained by Abel-transforming the Fourier transform of the power spectrum. This is equivalent to perform a zeroth-order Hankel transform of the power spectrum, which leads to the last equivalance in Eq. (3.15).

Eq. (3.15) though, is valid as long as we are able to measure distances directly in an infinitesimal redshift bin, which is not realistic. Our projected distance depends on the angular separation, θ, and the cosmological distance, $d_C(z)$, of the observed object

$$r_p(\theta, z) = \theta \cdot d_C(z)\ . \tag{3.16}$$

By projecting the objects in our lightcone on a flat surface at the target redshift, we are summing up the contribution of all the objects along the line of sight. Therefore the two-point angular correlation function can be expressed as

$$\omega(\theta, z) = \int \frac{\mathrm{d}V(z)}{\mathrm{d}z}\mathcal{N}^2(z)\ \omega[r_p(\theta, z), z]\,\mathrm{d}z \tag{3.17}$$

where $\frac{\mathrm{d}V(z)}{\mathrm{d}z}$ is the comoving volume unit and $\mathcal{N}(z)$ is the normalized redshift distribution of the target population. If we assume that $\omega(\theta, z)$ is approximately constant in the redshift interval $[z_1, z_2]$, we can then write

$$\omega(\theta, z) \approx \left[\int_{z_1}^{z_2} \mathrm{d}z \frac{\mathrm{d}V(z)}{\mathrm{d}z}\mathcal{N}^2(z)\right] \cdot \omega[r_p(\theta), \overline{z})] \tag{3.18}$$

where $\overline{z}$ is the mean redshift of the objects in the interval.

3.2 The ScamPy library

In this Section we introduce ScamPy, our highly-optimized and flexible API for "painting" an observed population on top of the DM-halo/subhalo hierarchy obtained from DM-only N-body simulations. We will give here a general overview of the algorithm on which our API is based. We refer the reader to Chapter 4 for a description of the key aspects of our hybrid C++/Python implementation, where we point out how the package is intended for future expansion and further optimization. In the next Chapter we also provide an analysis of the API performances. The source code and a guide for

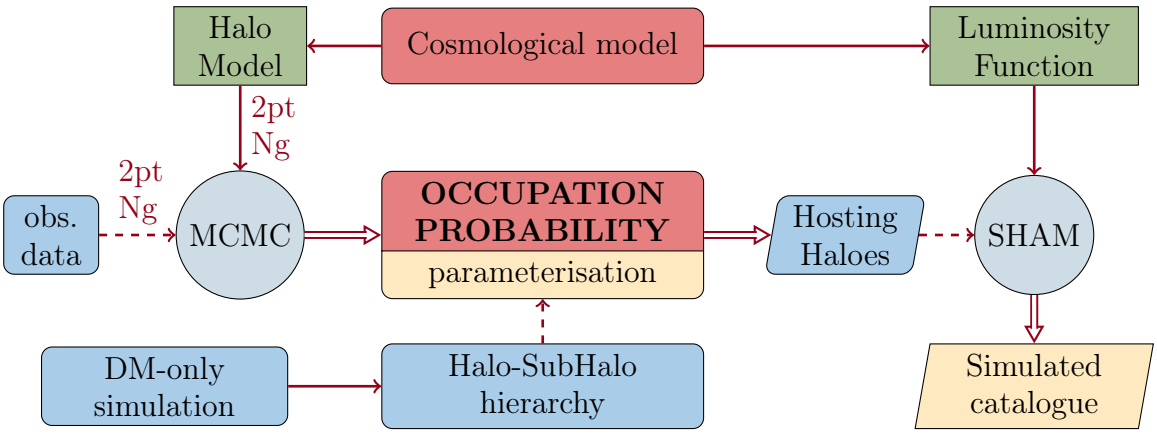

Figure 3.1: Flowchart describing the main components of the algorithm. In red the two main kernel modules. Green rectangles dub models from which the workflow depends. Round gray circles are for engines that operate on some inputs. Cyan is for inputs, yellow and parallelograms for outputs.

installing the library can be obtained by cloning the GitHub repository of the project (github.com/TommasoRonconi/scampy). The full documentation of ScamPy, with a set of examples and tutorials, is available on the website (scampy.readthedocs.io) of the package.

3.2.1 Algorithm overview

In Figure 3.1, we give a schematic view of the main components of the ScamPy package. All the framework is centred around the occupation probabilities, namely $N_{\rm cen}(M_h)$ and $N_{\rm sat}(M_h)$, which define the average numbers of, respectively, central and satellite galaxies hosted within each halo. Several parameterisations of these two functional forms exist. One of the most widely used is the standard 5-parameters HOD model (Zheng et al. 2007, 2009), with the probability of having a central galaxy given by an activation function and the number distribution of satellite galaxies given by a power-law:

$$N_{\rm cen}(M_h) = \frac{1}{2}\left[1 + {\rm erf}\left(\frac{\log M - \log M_{\rm min}}{\sigma_{\log M_h}}\right)\right] \tag{3.19}$$

$$N_{\rm sat}(M_h) = \left(\frac{M_h - M_{\rm cut}}{M_1}\right)^{\alpha_{\rm sat}} \tag{3.20}$$

where $M_{\rm min}$ is the characteristic minimum mass of halos that host central galaxies, $\sigma_{\log M_h}$ is the width of this transition, $M_{\rm cut}$ is the characteristic cut-off scale for hosting satellites, M_1 is a normalization factor, and $\alpha_{\rm sat}$ is the power-law slope. Our API provides users with both an implementation of the Eqs. (3.19) and (3.20), and the possibility to use

their own parameterisation by inheriting from a base `occupation_p` class.[1] Given that both the modeling of the observable statistics (Section 3.1.1) and the HOD method used for populating DM haloes depend on these functions, we implemented an object that can be shared by both these sections of the API. As outlined in Fig. 3.1, the parameters of the occupation probabilities can be tuned by running an MCMC sampling. By using a likelihood as the one exposed in Section 3.1, the halo-model parameterisation that best fits the observed 1- and 2-point statistics of a target population can be inferred.[2]

The chosen cosmological model acts on top of our working pipeline. Besides providing the user with a set of cosmographic functions for modifying and analysing results on the fly, it plays two significant roles in the API. On the one hand, it defines the cosmological functions that are used by the halo model, such as the halo-mass function or the DM density profile in Fourier space. On the other hand, it provides a set of luminosity functions that the user can associate to the populated catalogue through the SHAM procedure. This approach is not the only one possible, as users are free to define their own observable property distribution and provide it to the function that is responsible for applying the abundance matching algorithm.

Once the HOD parameterisation and the observable-property distribution have been set, it is possible to populate the halo/sub-halo hierarchy of a DM-only catalogue. In

Algorithm 1 Schematic outline of the steps required to obtain a mock galaxy catalogue with ScamPy.

```
// Load Halo/Subhalo hierarchy
// (e.g.  from SUBFIND algorithm)
halo_cat = catalogue(  chosen from file  )

// Choose occupation probability function
OPF = OPF( HOD parameters )

// Populate haloes
gxy_array = halo_cat.populate( model = OPF )

// Associate luminosities
gxy_array = SHAM( gxy_array, SHAM parameters )
```

Alg. 1, we outline the steps required to populate a halo catalogue with mock observables. We start from a halo/subhalo hierarchy obtained by means of some algorithm (e.g. SUBFIND) that have been run on top of a DM-only simulation. This is loaded into a `catalogue` structure that manages the hierarchy dividing the haloes in *central* and *satellite* subhaloes.[3]

[1] The documentation of the library comprehends a tutorial on how to achieve this.

[2] In the documentation website we will provide a step-by-step tutorial using `Emcee` (Foreman-Mackey et al. 2013).

[3] For the case of SUBFIND run on top of a GADGET snapshot this can be done automatically using the `catalogue.read_from_gadget()` function. We plan to add similar functions for different halo-finders

Our `catalogue` class comes with a `populate()` member function that takes an object of type occupation probability as argument and returns a trimmed version of the original catalogue in which only the central and satellite haloes hosting an object of the target population are left. We give a detailed description of this algorithm in Section 3.2.1. When this catalogue is ready, the SHAM algorithm can be run on top of it to associate at each mass a mock-observable property. Cumulative distributions are monotonic by construction. Therefore it is quite easy to define a bijective relation between the cumulative mass distribution of haloes and the cumulative observable property distribution of the target population. This algorithm is described in Section 3.2.1.

Populating algorithm

Input subhalo catalogues are trimmed into hosting subhalo catalogues by passing to the `populate()` member function of the class `catalogue` an object of type `occupation_p`.

In Algorithm 2, we describe this halo occupation routine. For each halo i in the

Algorithm 2 Description of the `populate` (model = OPF) function. This is an implementation of the HOD prescription, where the assumptions made to define the halo model (i.e. the average number of objects within a halo follows a Poisson distribution with mean $\langle N_g \rangle(M_h)$) are accounted for.

```
// Iterate over all the haloes in catalogue
for halo in catalogue do

    // Compute probability of central
    p_cen ← model.N_cen( halo.mass )

    // Define a binomial random variable
    select ← random.Binomial(1, p_cen)
    if select then
        halo ← central

    // Compute average number of satellites
    N̄_sat ← model.N_sat( halo.mass )

    // Define a Poisson random variable
    N_sat = random.Poisson( N̄_sat )
    halo ← select randomly N_sat objects among satellites
```

catalogue, we compute the values of $\langle N_{\rm cen} \rangle(M_i)$ and $\langle N_{\rm sat} \rangle(M_i)$. To account for the assumptions made in our derivation of the halo model, we select the number of objects each halo will host by extracting a random number from a Poisson distribution. For the occupation of the central halo this reduces to extracting a random variable from a Binomial distribution: $N_{\rm cen} = \mathscr{B}(1, \langle N_{\rm cen} \rangle_i)$. While, in the case of satellite subhaloes,

(e.g. ROCKSTAR, Behroozi et al. (2013b), and SPARTA, Diemer (2017)) in future extensions of the library.

we extract a random Poisson variable $N_{\rm sat} = \mathscr{P}(\langle N_{\rm sat}\rangle_i)$, then we randomly select $N_{\rm sat}$ satellite subhaloes from those residing in the $i^{\rm th}$ halo.

In Fig. 3.2, we show a 4 Mpc/h thick slice of a simulation with box side lenght of 64 Mpc/h, the background colour code represents the density field traced by all the subhaloes found by the SUBFIND algorithm, smoothed with a Gaussian filter, while the markers show the positions of the subhaloes selected by the populating algorithm. We will show in Section 3.3.1 that this distribution of objects reproduces the observed statistics. It is possible to notice how the markers trace the spatial distribution of the underlying DM density field.

Abundance matching algorithm

When the host subhaloes have been selected we can run the last step of our algorithm. The `abundance_matching()` function implements the SHAM prescription to associate to each subhalo an observable property (e.g. a luminosity or the star formation rate of a galaxy). This is achieved by defining a bijective relation between the cumulative distribution of subhaloes as a function of their mass and the cumulative distribution of the property we want to associate them.

An example of this procedure is shown in Fig. 3.3. We want to set, for each subhalo, the UV luminosity of the galaxy it hosts. In the left panel, we show the cumulative mass-distribution of subhaloes, $\mathrm{d}N(M_{\rm subhalo})$, with the dashed green region being the mass resolution limit of the DM subhaloes in our simulation after the populating algorithm has been applied. On the right panel we show the cumulative UV luminosity function, which is given by the integral

$$\Phi(M^{\rm UV} < M^{\rm UV}_{\rm lim}) = \int_{-\infty}^{M^{\rm UV}_{\rm lim}} \frac{\mathrm{d}\Phi}{\mathrm{d}M^{\rm UV}}\,\mathrm{d}M^{\rm UV} \tag{3.21}$$

where $M^{\rm UV}_{\rm lim}$ is the limiting magnitude of the survey data we want to reproduce (marked by a dashed red region in Fig 3.3). We find the abundance corresponding to each mass bin (gray step line in the left panel) and we compute the corresponding luminosity by inverting the cumulative luminosity function obtained with Eq. (3.21):

$$M^{\rm UV}(M_{\rm subhalo}) = \Phi^{-1}[\,\mathrm{d}N(M_{\rm subhalo})]\ . \tag{3.22}$$

The result of this matching is shown by the orange crosses in the right panel of Fig. 3.3. At the time we are writing, the scatter around the distribution of luminosities can be controlled by tuning the bin-width used to measure $\mathrm{d}N(M_{\rm subhalo})$. We plan to extend this functionality of the API by adding a parameter for tuning this scatter to the value chosen by the user.

In Figure 3.2, the colour gradient of markers highlights the increase in their associated luminosity (from lighter to darker colour, going from fainter to brighter object).

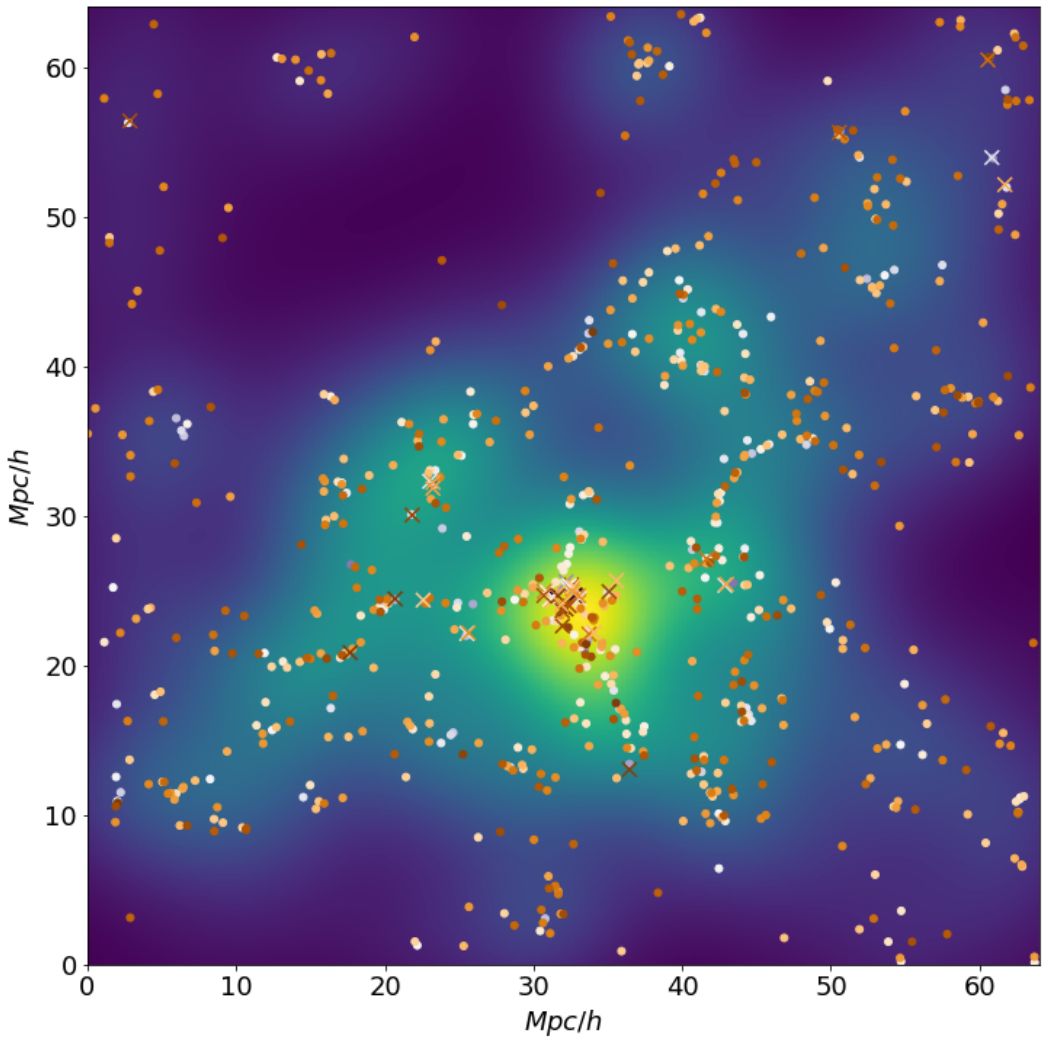

Figure 3.2: A 4 Mpc/h thick slice of a populated catalogue obtained from a DM-only simulation with 64 Mpc/h box side lenght. The colour code on the background shows the smoothed DM density field (with density increasing going from darker to brighter regions) while the markers show our mock galaxies (with color representing lower to higher luminosity going from brighter to darker). Circles are for centrals and crosses for satellites.

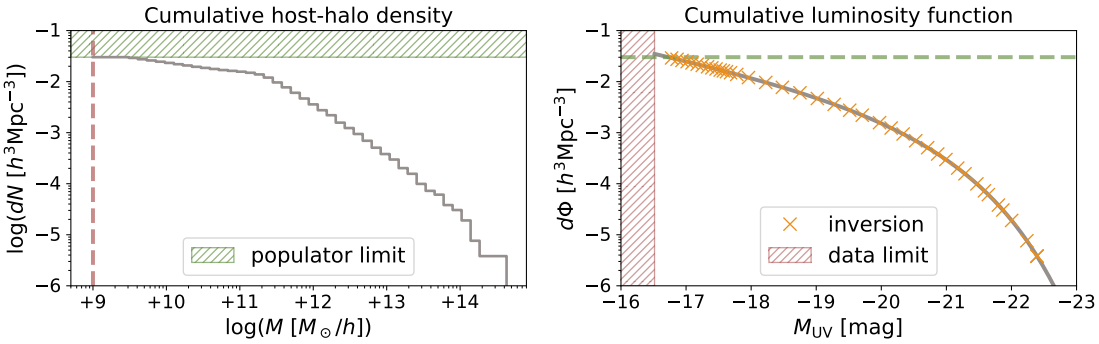

Figure 3.3: Abundance matching scheme. *Left panel:* cumulative number density of host subhaloes divided into regular logarithmic bins (solid gray step-line). The green dashed band shows the limit imposed by the resolution in mass of the simulation, which is inherited by the host catalogue obtained with the populator algorithm. *Right panel:* cumulative luminosity function (solid gray line). The dashed red band shows the limit imposed by the magnitude limit of the observed target population. Orange crosses mark the positions, bin-per-bin, of the abundances measured in each bin of the left panel. In both the left and the right panel we reported with a dashed line of corresponding colour the limit imposed by the resolution of the catalogue (dashed green line in right panel) and by the magnitude limit of the survey (dashed red line in the left panel).

3.3 Verification & Validation

We have extensively tested all the functions building up our API in all their unitary components. We developed a testing machinery, included in the official repository of the project, to run these tests in a continuous integration environment. This will both guarantee consistency during future expansions of the library, as long as providing users with a quick check that the build have been completed successfully.

In this Section, we show that our machinery is producing the expected results. Specifically, in Section 3.3.1 we show that the mock-catalogues obtained with ScamPy reproduce the observables we want. We have also tested our API for the accuracy in reproducing cross-correlations in Section 3.3.2. Even though there is no instruction in the algorithm that guarantees this behaviour, using the halo model it is trivial to obtain predictions for the cross-correlation of two different populations of objects.

All the validation tests have been obtained by assuming a set of reasonable values for the HOD parameters of Eqs. (3.19) and (3.20). All the parameters used are listed in Table 3.1. We will refer to these sets of parameters as *fiducial model* in the rest of this manuscript.

The resulting occupation probabilities have been then used to populate a set of halo/-subhalo catalogues. These catalogues have been obtained by running on the fly the FoF and SUBFIND (Springel et al. 2001) algorithms on top of a set of cosmological N-body simulations. The DM snapshots have been obtained by running the (non-public) P-

Table 3.1: Fiducial values of the HOD parameters at the different redshifts inspected. The HOD model is defined by Eqs. (3.19) and (3.20).

z	$M_{\rm min}$ $[10^{10}\ M_\odot/h]$	$\sigma_{\log M}$	$M_{\rm cut}$ $[M_\odot/h]$	M_1 $[10^{12}\ M_\odot/h]$	$\alpha_{\rm sat}$
0	5.0	0.3	0.0	1.0	1.0
2	2.0	0.3	0.0	1.0	1.0
4	2.0	0.3	0.0	1.0	1.0
6	1.0	0.3	0.0	0.5	1.2
8	1.5	0.2	0.0	0.5	1.2

Table 3.2: Our set of cosmological simulations with the corresponding relevant physical quantities: $N_{\rm part}$ is the total number of DM particles; $M_{\rm part}$ is the mass of each particle; $L_{\text{box-side}}$ is the side lenght of the simulation box; $z_{\rm min}$ is the minimum redshift up to which the simulation has been evolved.

name	$N_{\rm part}$	$M_{\rm part}$	$L_{\text{box-side}}$	$z_{\rm min}$
`lowres`	512^3	$8.13 \times 10^7\ M_\odot/h$	64 Mpc/h	0
`midres`	1024^3	$1.02 \times 10^7\ M_\odot/h$	64 Mpc/h	2
`highres`	1024^3	$1.27 \times 10^6\ M_\odot/h$	32 Mpc/h	2

GADGET-3 N-body code (which is derived from the GADGET-2 code, Springel 2005). In Table 3.2 we list the different simulation boxes we used for testing the library. Given the large computational cost of running high resolution N-body codes, only the `lowres` simulation box has been evolved up to redshift $z = 0$, while we stopped the others at redshift $z = 2$. The cosmological parameters used for all these simulations are summarised in Table 3.3.

3.3.1 Observables

Here we present measurements obtained after both the populating algorithm of Section 3.2.1 and the abundance matching algorithm of Section 3.2.1 have been applied to the DM-only input catalogue. Applying the abundance matching algorithm does not modify the content of the populated catalogue, besides associating to each mass an additional observable-property.

Table 3.3: Fiducial cosmological parameters of the N-body simulations used in this work.

h	$\Omega_{\rm CDM}$	Ω_b	Ω_Λ	σ_8	n_s
0.7	0.3	0.045	0.7	0.8	0.96

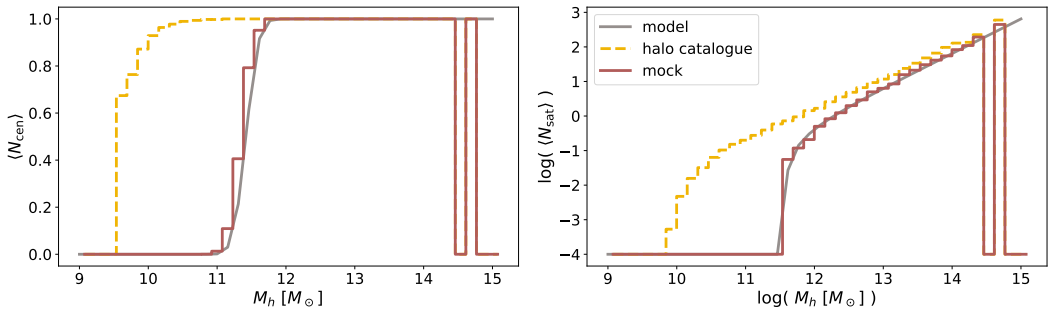

Figure 3.4: Occupation probability functions for central subhaloes (*left panel*) and satellite subhaloes (*right panel*). The gray solid line marks the $z = 0$ fiducial model of Table 3.1. The yellow step-wise dashed line and the red step-wise solid line mark the distributions measured on the subhalo catalogue before and after having applied our populating algorithm.

In the two panels of Fig. 3.4 we show the abundances of central and satellite subhaloes, as a function of the halo mass. The dashed yellow step-lines show the distribution in the DM-only input catalogue, while the gray solid line marks the distribution defined by the fiducial occupation probability functions. By applying the populating algorithm (Section 3.2.1) to the input catalogue we obtain the distributions marked by the solid red step-lines, which are in perfect agreement with the expected distribution.

We then draw a random Gaussian sample around the halo model estimate for the objects abundance and their clustering using the above selection of occupation probabilities (Eqs. (3.5) and (3.6) respectively). These random samples build up our *mock dataset*. We then run an MCMC sampling of the parameter space, with the likelihood of Eq. (3.1), to infer the set of parameters that best fit the mock dataset. For sampling the parameter space we use the Emcee (Foreman-Mackey et al. 2013) Affine Invariant MCMC Ensemble sampler, along with the ScamPy python interface to the halo model estimates of $n_g(z)$ and $\xi(r, z)$.

After having obtained the best-fit parameters, we produce 10 runs of the full pipeline described in Alg. 1. In doing this, we are producing 10 different realisations of the resulting mock catalogue. Since the selection of the host-subhaloes is not deterministic, this procedure allows to obtain an estimate of the errors resulting from the assumptions of the halo model. Finally, we use the Landy-Szalay (Landy & Szalay 1993) estimator in each of the populated catalogues to measure the 2-point correlation function:

$$\xi(r) = \frac{DD(r) - 2DR(r) + RR(r)}{RR(r)} \tag{3.23}$$

where $DD(r)$ is the normalised number of unique pairs of subhaloes with separation r, $DR(r)$ is the normalised number of unique pairs between the populated catalogue and a mock sample of objects with random positions, and $RR(r)$ is the normalised number of unique pairs in the random objects catalogue. We then measure, with Eq. (3.23), the

clustering in each of the 10 realisations and we compute the mean and standard deviation of these measurements in each radii bin.

The results are shown in Fig. 3.5, for redshift $z = 0$, and in Fig. 3.6, for redshifts $z = 2, 4, 6, 8$. In the upper panel of Fig. 3.5 we show the mock dataset with triangle markers and errors, the lines show the halo model best-fit estimate of the 2 point correlation function (with the different contributes of the 1- and 2-halo terms). The circle markers show the average measure obtained from our set of mock-catalogues.

In the lower panel of Fig. 3.5 and in the four panels of Fig. 3.6 we show the relative distance between the measure performed on the catalogue populated with the best-fit parameterisation of the occupation probabilities ($\xi_{\rm b.fit}$) and on a catalogue populated with the fiducial value of the parameters ($\xi_{\rm fid}$). Comparing the measurements in the two different populated catalogues, instead of comparing with the model itself, guarantees that discrepancies due to box-size and resolution of the simulation used are mitigated in the distance ratio plot.

As it is shown in the lower panel of Fig. 3.5, at redshift $z = 0$, the catalogue populated with the best-fit parameters reproduces the clustering properties of the fiducial catalogue with a distance lower than 15% on most of the scales inspected.

The discrepancies at small scales could depend on several effects. In particular:

- Implementation of the populating algorithm: galaxies can be assigned to satellite haloes randomly or by rank-ordering them based on some property of the sub-halo. The two methods might produce different levels of clustering within the halo.

- The sub-halo finder used: SUBFIND is known for not resolving completely the sub-halo hierarchy closer to the halo centre.

- Simulation resolution: discretization of the dark matter distribution limits the smallest sub-halo that can be modeled. This also results in a larger scatter when going at higher redshift (as shown in Fig. 3.6) where the average size of a halo gets smaller.

Concerning the latter point, it is worth to mention that, even thought the measurement is less precise, the average distance from the expected result is still lower than $10 \div 15\%$. Finally, in both Fig. 3.5 and Fig. 3.6, we mark with empty circles measurements taken at radii $r > 6.4$ Mpc$/h$, where the limited size of the simulation box affects the statistics.

All the discrepancies we find are a known weakness of the HOD method. In literature there have been a lot of effort in quantifying and correcting this effect (see e.g. Beltz-Mohrmann et al. 2019; Hadzhiyska et al. 2019, for two recent works), which, as already mentioned, is thought to result from a concurrence of box-size effects, cosmic-variance and assembly bias. Hadzhiyska et al. (2019), in particular, find an average distance of 15% between the HOD prediction and the clustering measured in hydro-dynamical N-body simulations.

The requirement of reproducing the 1-point statistics of the original catalogue is necessary to have the expected observational property distribution in the output mock-catalogue. This requirement guarantees that the abundance matching scheme will start

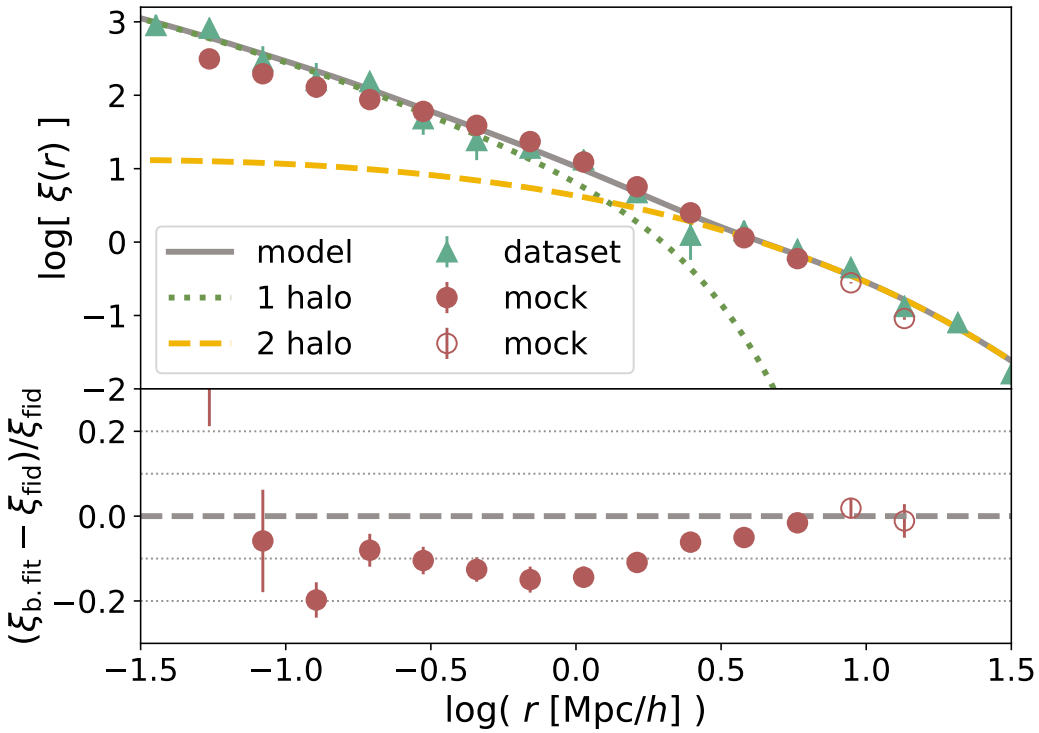

Figure 3.5: Validation of the two point correlation function at redshift $z = 0$. *Upper panel:* comparison between the mock clustering dataset (triangles), the best-fit halo-model prediction (solid line) and the mean and standard deviation of the clustering measured with the Landy-Szalay estimator on the 10 realisations (circles and errors), we also show the modelled 1-halo (dotted line) and 2-halo (dashed line) terms for reference. *Lower panel:* red markers show the distance ratio between the measurement obtained on the mock catalogue with the best-fit parameters and the averaged measurement obtained on the mock catalogue with fiducial parameters. The dashed line shows 0% distance between the two. In both panels, empty circles mark measurements at $r > 6.4$ Mpc/h, where the limited box size affects the precision of the result.

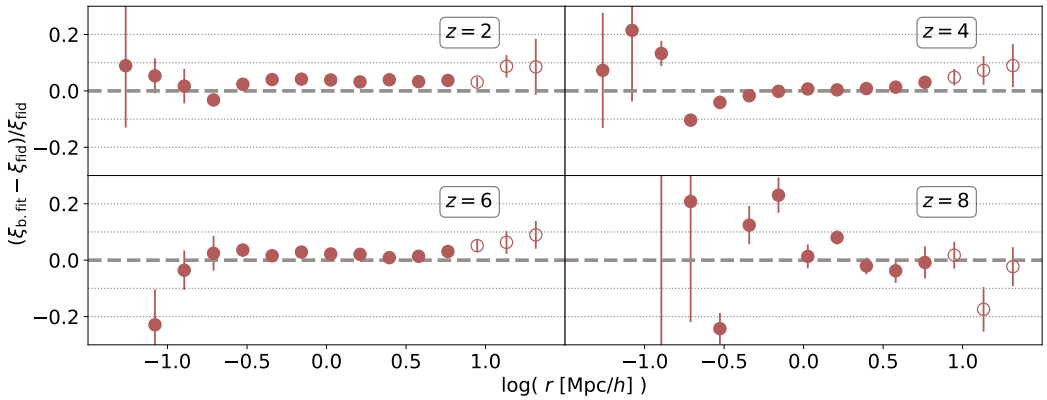

Figure 3.6: Validation of the two point correlation function at redshift $z = 2$ (upper left panel), $z = 4$ (upper right panel), $z = 6$ (lower left panel), $z = 8$ (lower right panel). Red markers show the distance ratio between the averaged-measurement obtained on the mock catalogue with the best-fit parameters and the measurement obtained on the mock catalogue with fiducial parameters. The dashed line shows 0% distance between the two. As in Fig. 3.5, empty circles mark measurements at $r > 6.4$ Mpc$/h$

associating the observational property from the right position in the cumulative distribution, i.e. from the abundance corresponding to the limiting value that said property has in the survey.

In Fig. 3.7, we show the example case of the UV luminosity function. We mark with orange circles the cumulative luminosity function measured on the mock-catalogue after the application of our API. For comparison we also show the luminosity function model we are matching (gray solid line) and the observation limit of the target population (red dashed region).

As it is shown in the lower panel of Fig. 3.7, the distance ratio between the expected distribution and the mock distribution is lower than $\approx 10\%$ over all the range of magnitudes.

The halo-model prediction for the total abundance of sources is $n_g^{\rm hm}(z) = 3.49 \cdot 10^{-2}$ $[h^3{\rm Mpc}^{-3}]$, while we measure $n_g^{\rm pop}(z) = (3.06 \pm 0.04) \cdot 10^{-2}$ $[h^3{\rm Mpc}^{-3}]$ in the populated catalogue. Such a miss-match is somehow expected. In fact, it has been shown (e.g. Sinha et al. 2018) that the HOD model presents difficulties when fitting multiple statistics. For the distribution of sources shown in Fig. 3.7, we correct this error by extrapolating the limiting magnitude (marked by the hatched region) to the value matching the abundance of the populated galaxies.

Sinha et al. (2018) also show that the clustering statistics with the higher constraining power depends on the galaxy population that has to be modeled. Since we are running our analysis on a simulated dataset, with the aim of validating the computational framework, we are not pushing this analysis further. Nevertheless, in a real-life application, the likelihood of Eq. (3.1) should be adjusted considering these observations, depending on

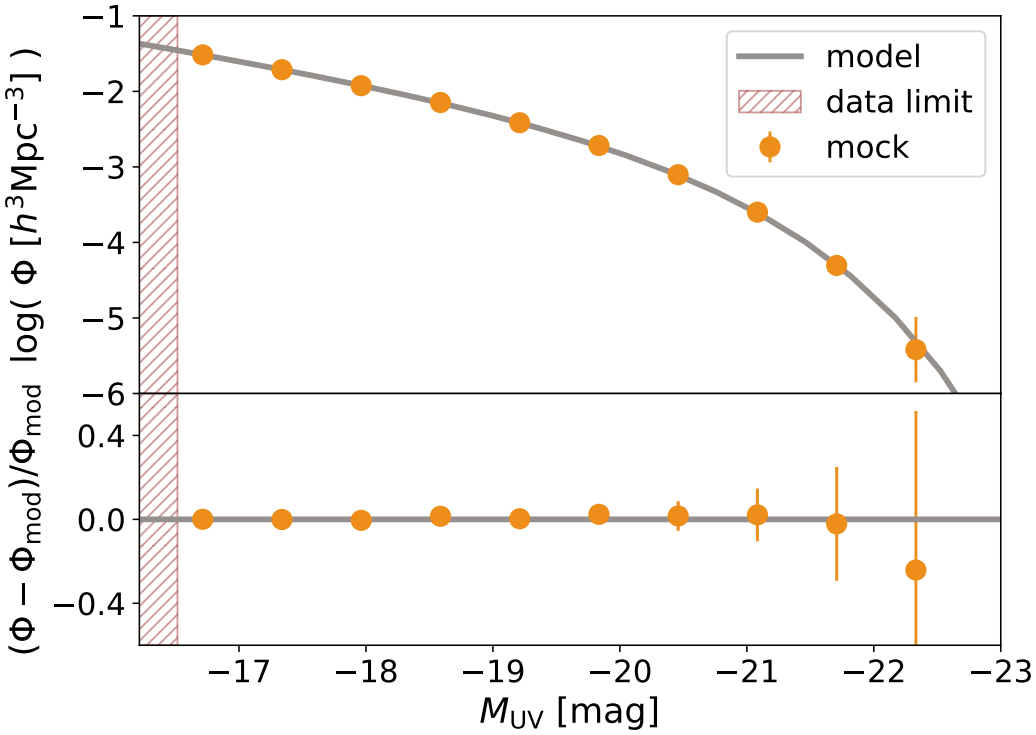

Figure 3.7: Cumulative luminosity function at redshift $z = 0$. *Upper panel:* the gray solid line marks the model prediction while the orange circles with errors mark the distribution measured on the populated catalogue. The hatched red region marks the limiting magnitude $M_{\rm lim}^{\rm UV}$. *Lower panel:* distance ratio between the luminosity function measured on the populated catalogue and the model.

the scientific goal of the application. The `halo_model` class of our library provides a wide range of cosmological statistics that can be modeled, leaving to the users the responsibility of choosing the one that better suits their needs.

3.3.2 Multiple populations cross-correlation

Even though in our API there is no prescription for this purpose, it is interesting to test how the framework performs in predicting the cross-correlation between two different populations. This quantity measures the fractional excess probability, relative to a random distribution, of finding a mock-source of population 1 and a mock-source of population 2, respectively, within infinitesimal volumes separated by a given distance.

It is simple to modify Eqs. (3.10) and (3.11) to get the expected power spectrum of the cross-correlation (Cooray & Sheth 2002). For the 1-halo term this is achieved by splitting the $(M_h/\overline{\rho})^2$ of Eq. (3.10) in the contribution of the two different populations, which leads to the following equation:

$$P_{\rm 1h}^{(1,2)}(k,z) = \frac{1}{n_g^{(1)}(z) n_g^{(2)}(z)} \cdot \int_{M_{\rm min}}^{M_{\rm max}} N_g^{(1)}(M_h) N_g^{(2)}(M_h) n_h(M_h) |\widetilde{u}_h(k, M_h, z)|^2 {\rm d}M_h \tag{3.24}$$

where quantities referring to the two different populations are marked with the superscripts (1) and (2).

For the case of the 2-halo term, obtaining an expression for the cross-correlation requires to divide the two integrals of Eq. (3.11) in the contributions of the the two different populations, leading to

$$P_{\rm 2h}^{(1,2)}(k,z) = \frac{P_m(k,z)}{n_g^{(1)}(z) n_g^{(2)}(z)} \cdot \left[\int_{M_{\rm min}}^{M_{\rm max}} N_g^{(1)}(M_h) n_h(M_h) b_h(M_h,z) \widetilde{u}_h(k, M_h, z) {\rm d}M_h\right] \cdot \left[\int_{M_{\rm min}}^{M_{\rm max}} N_g^{(2)}(M_h) n_h(M_h) b_h(M_h,z) \widetilde{u}_h(k, M_h, z) {\rm d}M_h\right] \tag{3.25}$$

We get the cross-correlation of the two mock-populations using a modification (González-Nuevo et al. 2017) of the Landy-Szalay estimator

$$\xi^{(1,2)}(r) = \frac{D_1D_2(r) - D_1R_2(r) - D_2R_1(r) + R_1R_2(r)}{R_1R_2(r)} \tag{3.26}$$

where $D_1D_2(r)$, $D_1R_2(r)$, $D_2R_1(r)$ and $R_1R_2(r)$ are the normalized data1-data2, data1-random2, data2-random1 and random1-random2 pair counts for a given distance r.

In Fig. 3.8 the red circles show the cross-correlation measured with Eq. (3.26) for two different mock-populations with a dummy choice of the occupation probabilities

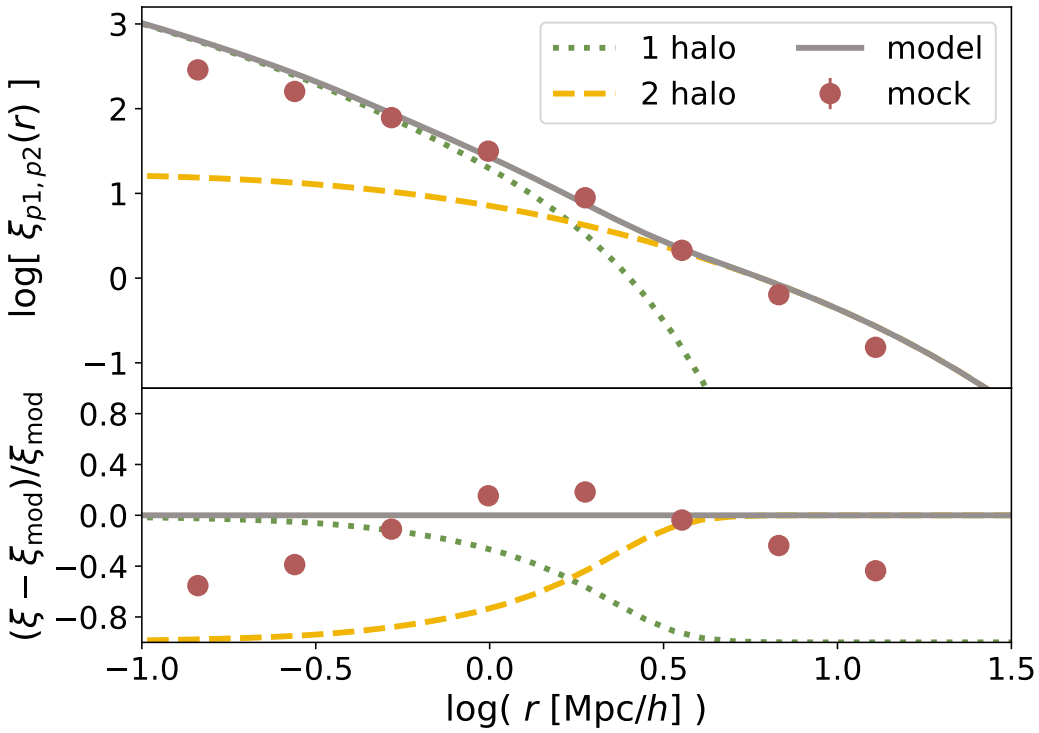

Figure 3.8: Comparison between the cross-correlation function, measured with the modified Landy-Szalay estimator of Eq. (3.26), between two dummy mock-populations at redshift $z = 0$. The lower panel shows the distance ratio between the measurement and the model prediction.

parameters. Errors are measured using a bootstrap scheme with 10 sub-samples. For comparison, we also show the halo-model prediction of the two-point correlation function, obtained with Eqs. 3.24 and 3.25, separated in 1-halo and 2-halo term contribution. The lower panel of Fig. 3.8 shows the distance ratio between the measure and the model prediction, which is lower than 40% over almost all the scales inspected.

3.4 Application on real galaxy catalogue

In order to produce priors for the HOD analysis performed in Bonavera et al. (2020) we have used the `halo_model` module of ScamPy to infer the astrophysical parameters of the model assuming flat wide priors. The aim of the work was to test the capability of the magnification bias (i.e. a weak gravitational lensing effect) produced on high redshift sub-millimetre galaxies as a cosmological probe. Our analysis was necessary to have an independent estimation of the HOD parameters that correspond to the foreground galaxies acting as lenses.

As detailed in González-Nuevo et al. (2017), the foreground sources were drawn from

Field	n_g[Mpc^{-3}]	$\sigma_{n_g[\mathrm{Mpc}^{-3}]}$
G09	1.528e-3	6.095e-6
G12	1.052e-3	4.023e-6
G15	1.180e-3	4.601e-6
med	1.217e-3	6.095e-6

Table 3.4: Number density of sources in the three selected GAMA fields, with the cuts described in the text.

the GAMA II (Driver et al. 2011; Baldry et al. 2010, 2014; Liske et al. 2015) spectroscopic survey. In Figure 3.9 we show the redshift distribution of galaxies in the 3 different GAMA fields of our sample.

For the auto-correlation analysis, we focus on the three largest fields: G09, G12 and G15. They constitute $\sim 63\%$ of the whole GAMA survey and they are the only fields with complete overlap with the H-ATLAS fields, from which the background sample used in the work is drawn. We expect no relevant differences in their correlation properties compared to the rest of the survey. Moreover, we perform a further reduction in the foreground sample by selecting the normalising redshift quality (catalogue column labelled "NQ") greater than 2 and the spectroscopic redshift (catalogue column labelled "Z") in the range $0.05 < z_{\mathrm{spec}} < 0.6$. The resulting median redshift of the sample of sources is $z_{\mathrm{med}} = 0.23$.

We estimate the number density of sources by computing the volume of each field as $V_{\mathrm{field}} = A_{\mathrm{field}} \cdot \Delta d_C$, where $\Delta d_C = d_C(z_{\mathrm{max}}) - d_C(z_{\mathrm{min}})$ is the difference in comoving distance between the maximum and minimum redshift selected. The area of the field is computed as $A_{\mathrm{field}} = d_{\mathrm{hav}}(RA) \cdot d_{\mathrm{hav}}(Dec)$ where d_{hav}(RA) and d_{hav}(Dec) are the haversine distances

$$d_{\mathrm{hav}}/r = \arcsin\left(\sqrt{\sin\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1 \cos\phi_2 \left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{3.27}$$

along the RA and Dec directions, respectively.

In Table 3.4 we show the measured abundances obtained for the 3 different selected fields with the cuts described above. The last row of the table shows the median value of n_g and maximum error value, σ_{n_g}, among the 3 fields.

We compute the angular correlation function $\omega(\theta)$ in each field by applying the Landy-Szalay estimator

$$\omega(\theta) = \frac{DD(\theta) - 2DR(\theta) + RR(\theta)}{RR(\theta)} \tag{3.28}$$

and estimate the error on this measure by dividing each field in 128 bootstrap samples. In Figure 3.10, we show the covariance matrices obtained for the angular correlation function with the bootstrapping done in the three fields. As the picture shows, the large number of bootstrap samples guarantees convergence in the estimation of the uncertainties. The measured angular correlation is shown in the left panel of Fig. 3.11, marked with red dots and errors.

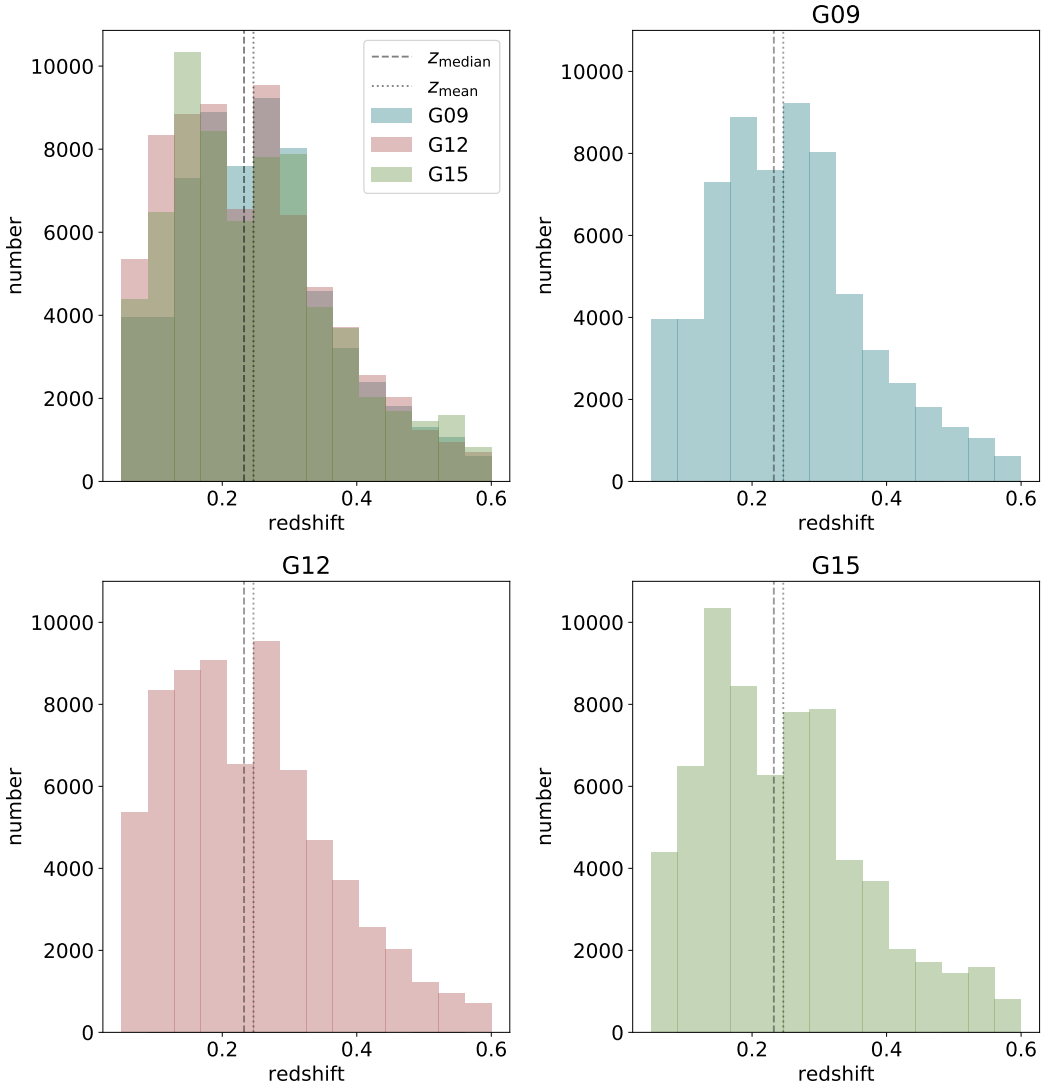

Figure 3.9: Redshift distributions of the GAMA II fields dubbed `G09`, `G12` and `G15`. The vertical dashed and dotted black lines show the median and mean redshift of the final sample, respectively.

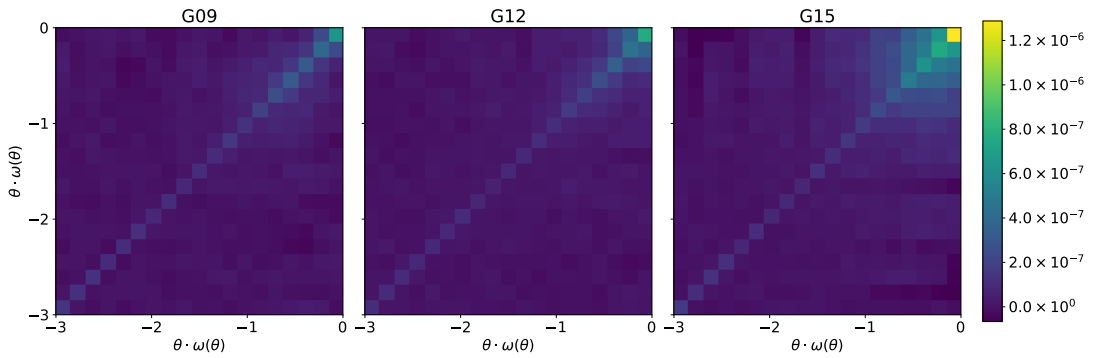

Figure 3.10: Covariance matrices plotted as the product $\theta \cdot \omega(\theta)$ for the 3 different GAMA fields. The error estimate is obtained by drawing 128 bootstrap samples in each of the fields considered.

In the HOD model used the number of central galaxies is parameterised as the step function

$$N_{\rm cen}(M_h) = \begin{cases} 0 & \text{if } M_h < M_{\rm min} \\ 1 & \text{otherwise} \end{cases} \tag{3.29}$$

where the only free parameter is $M_{\rm min}$. The satellite galaxies occupation is instead given by the conditional power law

$$N_{\rm sat}(M_h) = N_{\rm cen}(M_h) \cdot \left(\frac{M_h}{M_1}\right)^{\alpha_{\rm sat}} \tag{3.30}$$

with the additional free parameters M_1 and $\alpha_{\rm sat}$.

To estimate the above set of parameters, we performed a MCMC analysis using the open source emcee software package (Foreman-Mackey et al. 2013). In this analysis we generated approximately 10^6 posterior samples to ensure a good statistical sampling after convergence. The parameter space sampling has been done by maximising the likelihood of Eq. (3.1), built as the sum of the χ^2 of the two measures we want to fit, namely the two-point angular correlation function at a given redshift, $\omega(\theta, z)$ as given in Eq. (3.18), and the average number of sources at a given redshift, $n_g(z)$ as given in Eq. (3.5). In this analysis, we consider an additional free parameter for the redshift-dependent multiplicative factor of Eq. (3.18), defined as

$$\text{norm} \equiv \int_{z_1}^{z_2} {\rm d}z \frac{{\rm d}V(z)}{{\rm d}z} \mathscr{N}^2(z) \ . \tag{3.31}$$

Since most of the sources in our sample have redshift $\sim z_{\rm med}$ (as shown in Fig. 3.9), the error introduced by this simplification is negligible.

The posterior distributions for the model parameters are shown on the right panel of Fig. 3.11 while, on the left panel, we show a comparison between the best-fit halo-model prediction of $\omega(\theta)$ with the values measured from our sample.

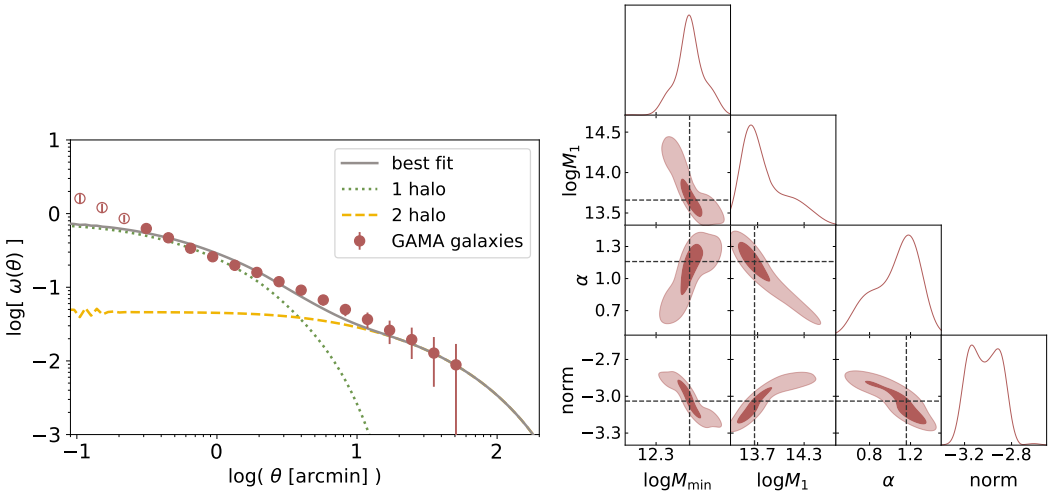

Figure 3.11: *left panel:* Measured angular auto-correlation function of the sample (red circles). The best-fit halo model prediction is shown as a comparison (total, gray solid line; 1-halo term, green dotted line; 2-halo term, yellow dashed line). *right panel:* Posterior distributions for the model parameters in the auto-correlation data analysis. The contours for these plots are set to 0.393 and 0.865, instead of 0.68 and 0.95, to allow direct comparison with the 1D histograms above the contours.

The model parameters are well constrained and they provide a good fit to the observed auto-correlation function, obtaining a χ^2_ω of 8.836 for 11 degrees of freedom (d.o.f.; corresponding to a reduced χ^2_ω of 0.803) and a $\chi^2_{n_g}$ of 0.001 (0.0001). Moreover, the values obtained for $\log(M_{\rm min}/M_\odot)$ and $\log(M_1/M_\odot)$ ($12.362^{+0.002}_{-0.007}$ and $13.64^{+0.003}_{-0.09}$, respectively) are consistent with what can be found in the literature ($\log(M_{\rm min}/M_\odot) \in [11.6, 13.6]$ and $\log(M_1/M_\odot) \in [13.0, 14.5]$, for the flat prior case). The derived α parameter ($1.17^{+0.06}_{-0.05}$) is also in agreement with the literature values ($\alpha \in [0.5, 1.37]$, for the flat priors case), although slightly toward the upper limit.

As a further exercise, we have divided our original sample of galaxies into 4 redshift bins and performed again the analysis above. The properties of the 4 different samples obtained along with the best-fitting values of the parameters are summarised in Table 3.5. As also shown in Figure 3.12, the best fitting model provides a good prediction of the observed auto-correlation function. The reduced $\chi^2_{\rm tot} = \chi^2_\omega + \chi^2_{n_g}$ is less than 1 in all cases. Nevertheless, it has to be mentioned that in the higher redshift window (sample B4) the inferred $M_{\rm min}$ and M_1 parameter values are higher than the values that can be found in literature. This might be due to some lensing effect caused by galaxies in the closer samples (from B1 to B3) and is worth further investigation.

Our overall conclusion is that our library can be confidently used to infer the astrophysical parameters of the HOD.

Name	Δz	$z_{\rm med}$	N_g	$\log \frac{M_{\rm min}}{M_\odot}$	$\log \frac{M_1}{M_\odot}$	α	log norm	$\chi^2_{\rm 11dof}$
B1	$[0.1, 0.2)$	0.15	62536	11.571	12.696	1.067	-2.455	0.38
B2	$[0.2, 0.3)$	0.26	60782	11.984	13.050	1.080	-2.538	0.33
B3	$[0.3, 0.5)$	0.36	49772	12.599	13.749	1.311	-2.729	0.41
B4	$[0.5, 0.8)$	0.56	10249	13.385	15.997	0.863	-2.601	0.36

Table 3.5: Summary of the galaxy samples properties. First column: identification name of the sample; second column: redshift window; third column: median redshift of galaxies in the sample; fourth column: total number of objects observed; fifth to eighth column: best-fitting parameters of the halo model; last column: chi-squared reduced accounting for the 11 degrees of freedom of the problem.

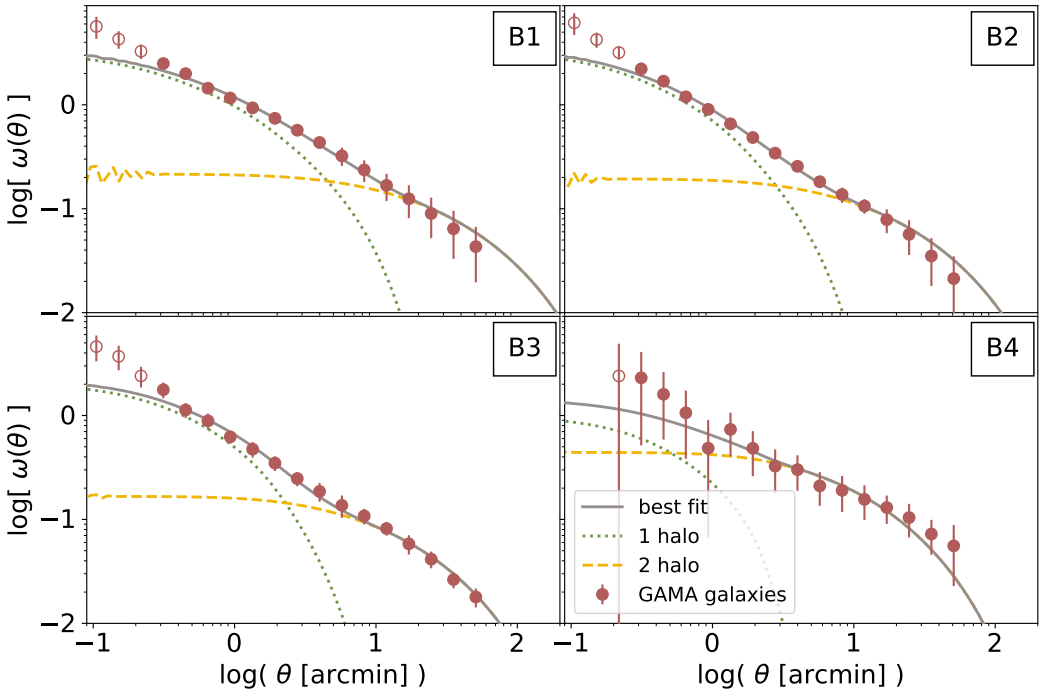

Figure 3.12: Measured angular auto-correlation function of the four redshift samples (red circles). The best-fit halo model prediction is shown as a comparison (total, gray solid line; 1-halo term, green dotted line; 2-halo term, yellow dashed line).

Chapter 4

HPC-driven development

In this Chapter we comment some design choices made for optimizing the performances of the library. A detailed discussion on the implementation is provided in Section 4.1. For some benchmarking measures of the API performances we refer the reader to Section 4.2.

With our hybrid implementation we have deployed a library that exploits both the performance efficiency of a compiled language (C++) as well as the flexibility of an interpreted language (Python). The C++ core of the library has multi-threaded sections to run in parallel the most computationally demanding calculations of the workflow. This choice has been made in order to circumvent the Global Interpreter Lock (GIL) which would otherwise force the Python application to run on a single thread. Spawning threads directly from the C++ core is more efficient than using most of the Python packages for multi-threading. Nonetheless this behaviour can be suppressed by setting the corresponding environment variable accordingly.

Functions that do not spawn threads by default are those meant for modelling cosmological statistics (e.g. clustering and abundances) and functions that have a pure Python implementation.

Since the former functions would ideally be used in a MCMC framework, they run on a single thread. In this way all the cores of the CPU are available for parallelizing the parameter space sampling. Nevertheless, we have carefully optimized and benchmarked such functions (more details in Section 4.1 and 4.2, respectively): on a single thread, one single computation of the likelihood in Eq. (3.1) for a typical problem size (i.e. a dataset with approximately 10 degrees of freedom) takes approximately 1 millisecond, running on a common laptop.

Concerning the pure Python implementation, the `catalogue` class and the `populate` function are the only sections of the library that would gain from a multi-threaded core, given that they operate on lists of independent objects. Such functionality has not been implemented yet but would be a natural evolution of our library. At the current state of the implementation, both these operations take times in the order of $1 \div 10$ seconds for a sub-halo table with $10^{4\div5}$ objects. Loading and populating sub-haloes have $\mathcal{O}(N_{\rm sub})$ scaling, where $N_{\rm sub}$ is the number of sub-haloes loaded into the catalogue. The dependence of these time measurements to the machine architecture is negligible.

4.1 API structure & bushido

In the context of software development for scientific usage and, in general, whenever the development is intended for the use in Academia, the crucial aspects that would make the usage flexible are often overlooked.

In the development of ScamPy, we have considered the good practices in software development, such as cross-platform testing and the production of reasonable documentation for the components of the API. We have outlined a strategy for keeping the software ordered and easy to read while maintaining efficient the computation. The usage of advanced programming techniques, along with the design of a handy class dedicated to interpolation, also allowed to boost the performances of our code.

In this Appendix, we describe the framework we have developed, highlighting the best programming practices used, and commenting on the design choices.

The overall structure can be divided broadly into 4 main components:

- **C++ core** - it mainly deals with the most computationally expensive sections of the algorithm.
- **Python interface** - it provides the user interface and implements sections of the algorithm that do not need to be severely optimised.
- **Tests**, divided into **unit tests** and **integration tests**, are used for validation and consistency during code development.
- **Documentation**, provides the user with accessible information on the library's functionalities.

The organization of the source code is modular. Test and documentation sections are treated internally as modules of the library, and their development is, to some extent, independent to the rest of the API. Furthermore, not being essential for the API operation, their build is optional.

The Meson Build System deals with compilation and installation of the library. Much like the well-known CMake (reference website), it allows to ease the compilation and favours portability while automatizing the research and eventual download of external dependencies.

4.1.1 Modularization

The C++ and Python implementations are treated separately and have different modularization strategies. As we already anticipated, the c++ language is adopted to exploit the performances of a compiled language. Nonetheless, it also allows for multi-threading parallelisation on shared memory architectures. This would not be normally possible in standard python because of the Global Interpreter Lock, which limits the processor to execute exactly only one thread at a time.

Each logical piece of the algorithm (see Section 3.2.1 and Figure 3.1) has been implemented in a different module. This division has been maintained both in the core

c++ implementation and in the python interface. Bridging over the two languages has been obtained through the implementation of source C++ code with a C-style interface enclosed in an `extern "C"` scope to produce shared-libraries with C-style mangling. To wrap the compiled c++ libraries in python we use the CTypes module. This choice was made because CTypes is part of the Python standard. Therefore no external libraries or packages are needed. This choice favours portability and eases compilation.

All of the C++ modules are organized in different sub-directories with similar structure:

- `src` sub-directory, containing all the source files (`.cpp` extension);
- `include` sub-directory, containing all the header files (`.h` extension);
- a `meson.build` script for building.

All the Python implementation is hosted in a dedicated sub-directory of the repository. Each module of the python interface to the API is coded in a separate file. The python dependencies to the c++ implementation are included in the source files at compile time by the build system.

In Table 4.1, we list all the Python-modules provided to the user. They are divided between the C++ wrapped and the Python only ones. All of them are part of the `scampy` package that users can import by adding a

```
/path/to/install_directory/python
```

to their `PYTHONPATH`.

4.1.2 External dependencies

Scientific codes often severely depend on external libraries. Even though a golden rule when programming, especially with a HPC intent, is to *not reinvent the wheel*, external dependencies have to be treated carefully. If the purpose of the programmer is to provide their software with a wide range of functionalities, while adopting external software where possible, the implementation can quickly become a *dependency hell*.

For this reason, we decided to keep the dependence on external libraries to a reasonable minimum. The leitmotiv being, trying not to be stuck on bottlenecks requiring us to import external libraries while maintaining the implementation open to the usage along with the most common scientific software used in our field.

The `c++` section of the API depends on the following external libraries:

- **GNU Scientific Library** (Galassi et al. 2009, version 2 or greater): this library is widely used in the community and compiled binary packages are almost always available in HPC platforms.
- **FFTLog** (Hamilton 2000): also this library is a must in the cosmology community. In our API, we provide a `c++` wrap of the functions written in Fortran90. We have developed a patch for the original implementation that allows to compile the project with Meson (see fftlog_patch on GitHub for details).

Table 4.1: Python modules of the API. The first column lists the module names and the second provides a short description of the module purpose. We divided the table in two blocks, separating the modules of the package that depend on the C++ implementation from those that have a pure Python implementation.

Module	**Purpose**
	Wrapped from C-interface
`interpolator`	Templated classes and functions for cubic-spline interpolation in linear and logarithmic space
`cosmology`	Provides the interface and an implementation for cosmological computations that span from cosmographic to Power-Spectrum dependent functions, computations are boosted with interpolation
`halo_model`	Provides classes for computing the halo-model derivation of non-linear cosmological statistics.
`occupation_p`	Provides the occupation probability functions implementation.
	Python-only
`objects`	Defines the objects that can be stored in the class `catalogue` of the `scampy` package, namely `host_halo`, `halo` and `galaxy`.
`gadget_file`	Contains a class for reading the halo/sub-halo hierarchy from the outputs of the SUBFIND algorithm of GaDGET.
`catalogue`	It provides a class for organizing a collection of host-haloes into an hierarchy of central and satellite haloes. It also provides functionalities for authomatic reading of input files and to populate the Dark Matter haloes with objects of type `galaxy`.
`abundance_matching`	Contains routines used for running the SHAM algorithm.

- **OpenMP**: one of the most common APIs for multi-threading in shared memory architectures. It is already implemented in all the most common compilers, thus it does not burden on the user to include this dependency.

We are aware that a vast collection of libraries for cosmological calculations is already available to the community (Marulli et al. 2016; Astropy Collaboration 2013, 2018). The intent of our `cosmology` module is not to substitute any of these but to provide an optimized set of functions integrated in the API without adding a further dependence on external libraries. By using polymorphism (both static and dynamic) we tried to keep our API as much flexible as possible. We explicitly decided to not force the dependence to any specific Boltzmann-solver to obtain the linear power spectrum of matter perturbations (see Section 3.1.1), the choice is left to the user.

Furthermore, the choice of Python to build the user interface, allowed to easily implement functions that do not require any other specific library to work. An example is the `abundance_matching` module, which is almost completely independent to the rest of the API: the only other internal module needed is the `scampy.object` but all its functionalities can be obtained by using python lambdas and numpy arrays.

The only other python libraries used in ScamPy are:

- **CTypes** which is part of the Python standard and is used for connecting the C-style binaries to the Python interface.
- **Numpy** which, despite not being part of the standard, is possibly the most common python library on Earth and provides a large number of highly optimized functions and classes for array manipulation and numerical calculations.

4.1.3 Extensibility

Simplifying the addition of new features has been one of our objectives from the first phases of development. We wanted to be able to expand the functionalities of the API, both on the C++ side, in order to boost the performances, and on the Python side, in order to use the API for a wide range of cosmological applications.

This is easily achieved with the modular structure we have built up. Adding a new `c++` module reduces to including a new set of headers and source files in a dedicated sub-directory. Further details on the structure said sub-directory should have and on the way its build is integrated in the API will be provided in the library website.

Adding new modules to the Python interface is even simpler, as it only requires to add a new dedicated file in the `python/scampy` sub-directory. Eventually, it can be also appended to the `__all__` list in the `python/scampy/__init__.py` file of the package. In this case, it is not necessary to operate on the build system as it will automatically install the new module along with the already existing ones.

Table 4.2: Execution time in nanoseconds of the same function in different languages. For the python case, we also show the ratio with respect to the C++ execution time. The timings reported are the average of 10 runs on the 4 physical cores with hyper-threading disabled of a laptop with Intel® Core™i7-7700HQ 2.80GHz CPU.

Function	**C++**	**Python**	$\mathbf{t_{py}/t_{C++}}$
`cosmology` class			
c.tor	2.520e+05	8.892e+05	3.528
$d_C(z)$	2.083e+03	5.984e+03	2.873
$n(M,z)$	1.186e+08	1.197e+08	1.009
`halo_model` class			
c.tor	3.011e+09	2.961e+09	0.983
$n_g(z)$	1.917e+04	2.851e+04	1.488
$\xi(r,z)$	3.396e+06	1.784e+06	0.525

4.2 Performances & benchmarking

We have measured the performances of our API's main components and benchmarked the scaling and efficiencies of the computation at varying precision and work-load. We will show here a set of time measurements performed on the two main components of the library: the `cosmology` class and the `halo_model` class. These are the two classes that would most affect the performances in real-life applications of our API.

4.2.1 Wrapping benchmark

First of all, in Tab. 4.2, we show the execution time of the same function called from different languages. Since our hybrid implementation requires to bridge through C to wrap in python the optimisations obtained in C++, it is interesting to compare their respective execution time. Along with the python execution time, we also provide the ratio with respect to the reference C++ time, t_{C++}, for the same function. All the times are expressed in nanoseconds.

We are showing 3 typical member calls that are representative of the functionalities provided by the two classes. For both of them, we measured the constructor time (c.tor), the time for executing a function that returns a scalar ($d_C(z)$ and $n_g(z)$) and the execution time for a function returning an array ($n(M,z)$ and $\xi(r,z)$). It can be noticed that, especially for the `cosmology` class, by calling the same function in python, the execution time increases. The worst case is the `cosmology` class constructor time that looses a factor ~ 3.5 in python. It has to be noticed though, that the execution time is lower than a millisecond and, since the constructor is the member function that is called the less, this is not severely affecting the overall performance of the python interface.

Nonetheless, because of the larger number of function calls required by moving from

one language to another, loosing some performance is expected. What we did not expect is the gain in performance we are getting when moving to python, as it is shown in the last column of the `halo_model` class box of Tab. 4.2. This behaviour might be due to the different way memory is allocated, accessed and copied in python with respect to C++/C. Moreover, the timers used for measuring the execution in the different languages are different. Even by comparing measures taken with the same precision, it is not guaranteed to have the same accuracy.

4.2.2 Halo-model performances

We have then tested the execution time of the `halo_model` constructor and member functions at varying work-load. In our implementation, the `halo_model` class requires to define a set of interpolating functions at construction time, these functions can be defined using our `interpolator` class (see Table 4.1). The interpolation accuracy depends on the resolution of the interpolation grid. In the `halo_model` class, at fixed limits of the interpolation interval, this is controlled by the *thinness* input parameter, which takes typical values $50 \div 200$ in real-life applications.

In Fig. 4.1 we show how the constructor-time varies with varying thinness in the range $10 < thin < 10^3$. The plot is obtained by calling 10 times the constructor per each thinness value and then averaging (solid and dotted lines). The shaded region marks the best and worst execution time among the 10 runs. Instead of the actual execution time we show the percent distance with respect to perfect linear scaling (dashed line) for both the C++ case (blue) and the python case (red). We define the percent distance at given thinness as

$$\%\ \text{distance}(thin) \equiv 100 \cdot \frac{t(thin) - t_{\rm lin}(thin)}{t_{\rm lin}(thin)} \tag{4.1}$$

where $t_{\rm lin}(thin)$ is the execution time for given thinness in the linear scaling case, computed with respect to the C++ case. In the white text box of Fig. 4.1 we also show the C++ constructor time for $thin = 8$, as a reference. As the picture shows, the scaling is almost perfectly linear, with a maximum distance of the 0.06% in the C++ case.

Possibly the most crucial computational bottleneck of the whole API is the time taken by the computation of a full-model. With the term "full-model" we mean the execution of the two functions for computing the halo-model estimate of the 1- and 2-point statistics, namely $n_g(z)$ and $\xi(r, z)$. In an MCMC framework, while the constructor is called only once, these two functions are called tens of thousands of times. This is a necessary step to set the parameterisation of the SCAM algorithm.

As also shown in Tab. 4.2, the execution of the two single functions takes an amount of time which is in the order of the millisecond in the C++ case. We can also notice that the execution time of a full-model is dominated by the computation of the two point correlation function, $\xi(r, z)$. Since this function is operating on a vector and returning a vector, it is reasonable to expect that its execution time varies with the work-load, i.e. with the vector size.

In Fig. 4.2 we show the percent distance, defined as in Eq. (4.1), of the average full-model execution time at varying work-load (solid lines) with respect to the perfect linear

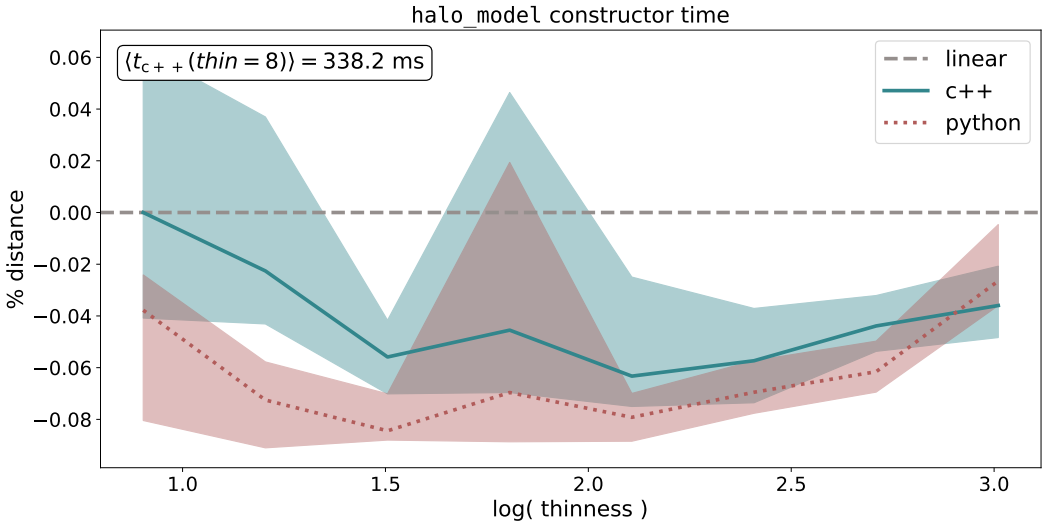

Figure 4.1: Percent distance between the constructor time scaling at varying thinness and the linear scaling case. For the python case, the percentage is computed with respect to the C++ time to ease the comparison. For reference, we also show in the white text-box the measured constructor time with thinness = 8.

scaling case (dashed line), in the range $2^3 \leq \text{load} \leq 2^{14}$. The measurements are obtained by averaging the results of 10 runs in both C++ (blue) and python (red). The shaded regions mark the best and worst performance among all the runs at varying workload. It can be noticed that, by increasing the work-load, the average execution time gets up to 15% worse than perfect linear scaling. This is due to some latency introduced by the necessity of Fourier transforming the power spectrum to model clustering. We have to point out though, that the typical work-load is in the range $5 \div 15$ for real-life applications and that the execution time in this cases is of the order of the millisecond.

Finally, we have measured how the constructor time scales with increasing number of multi-threading processors. We did not perform this measure for the full-model computation because, in the perspective of using it in a MCMC framework with parallel walkers, the full-model will be computed always serially.

We present measurements of both the constructor time *strong scaling* and *weak-scaling*. While the first measures the scaling with processor number at fixed thinness, the latter measures the scaling at thinness increasing proportionally with the processor number.

First of all, let us define the *speed-up*

$$S(p) = \frac{t(1)}{t(p)} \tag{4.2}$$

where p is the number of processors and $t(p)$ is the time elapsed running the code on p processors. This quantity measures the gain in performances one should expect when having access to larger parallel systems.

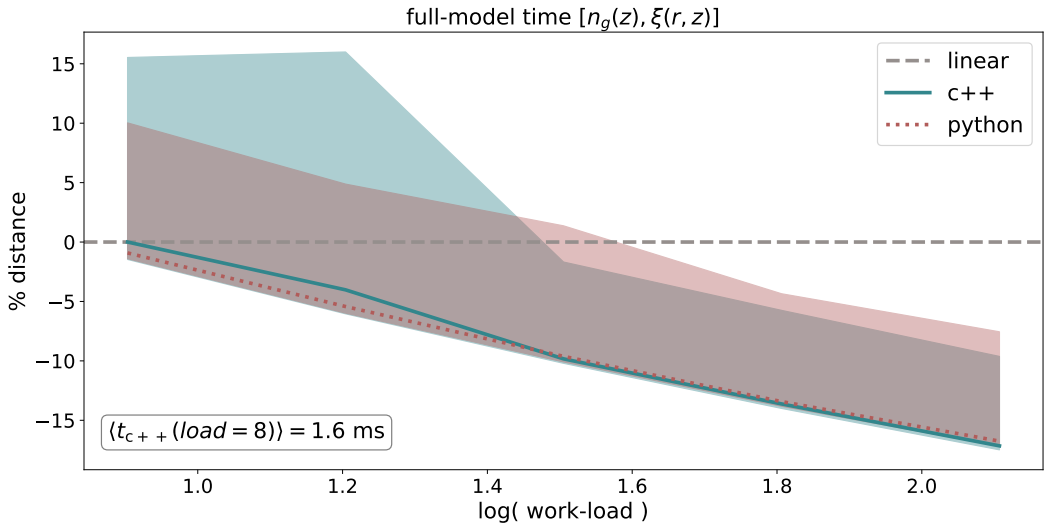

Figure 4.2: Percent distance between the full-model time scaling at varying work-load and the linear scaling case. For the python case, the percentage is computed with respect to the C++ time to ease the comparison. For reference, we also show in the white text-box the measured constructor time with work-load = 8.

We also define the efficiency for the strong and the weak scaling case:

$$\begin{aligned} E_{\text{strong}}(p) &= \frac{S(p)}{p} \\ E_{\text{weak}}(p) &= S(p) \end{aligned} \tag{4.3}$$

This quantity roughly measures the percentage of exploitation of the parallel system used. Thus, providing a hint of how much the serial part of the code is affecting the gain we can expect from spawning multiple threads.

We run these measures on a node from the `regular` partition of the SISSA Ulysses cluster.[1] Each of these nodes provide two shared memory sockets with 10 processors each. We measured the constructor time by averaging the results of 100 runs where the threads number has been controlled by setting

```
export OMP_NUM_THREADS=$ii
export OMP_PLACES=cores
export OMP_PROC_BIND=close
```

where `ii` varies in the set $\{1, 2, 4, 8, 16, 20\}$ and where the last two commands control the affinity of the processes spawned.

In Fig. 4.3 we show the speed-up (upper panel) and efficiency (lower panel) of the strong scaling. The dashed line marks perfect linear speed-up in the upper panel, and

[1] Please refer to the website for detailed informations.

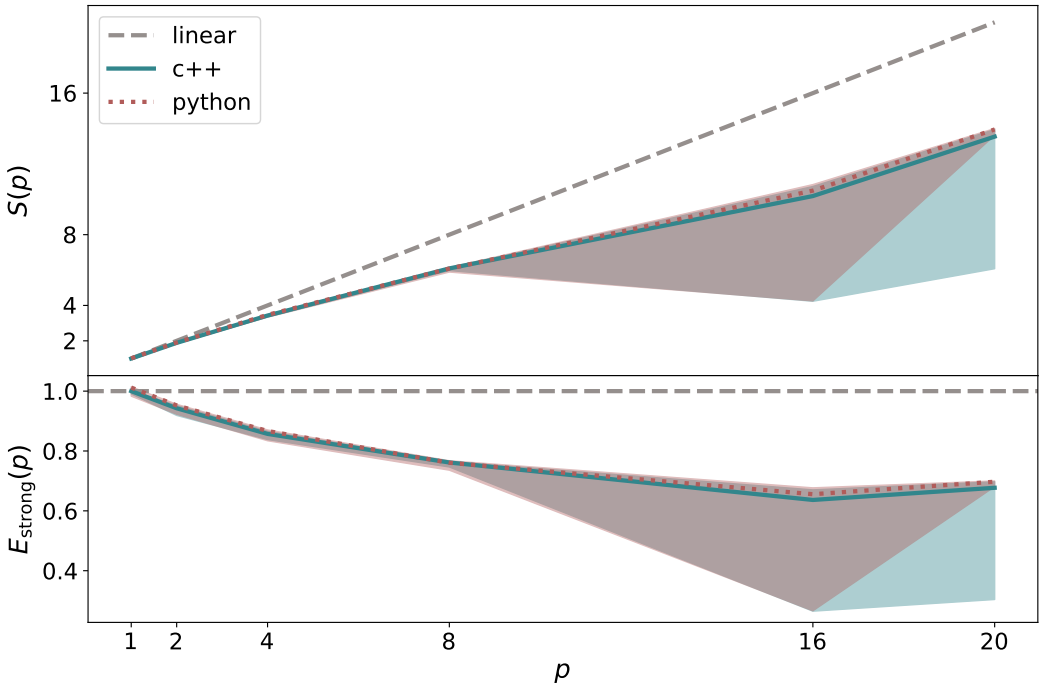

Figure 4.3: Strong-scaling speed-up (*upper panel*) and efficiency (*lower-panel*) of the constructor time at fixed thinness and varying number of multi-threading processors. The solid line marks the average of 100 runs while the shaded region marks the best and worst result area. We run the tests on a full computing-node of the SISSA Ulysses cluster.

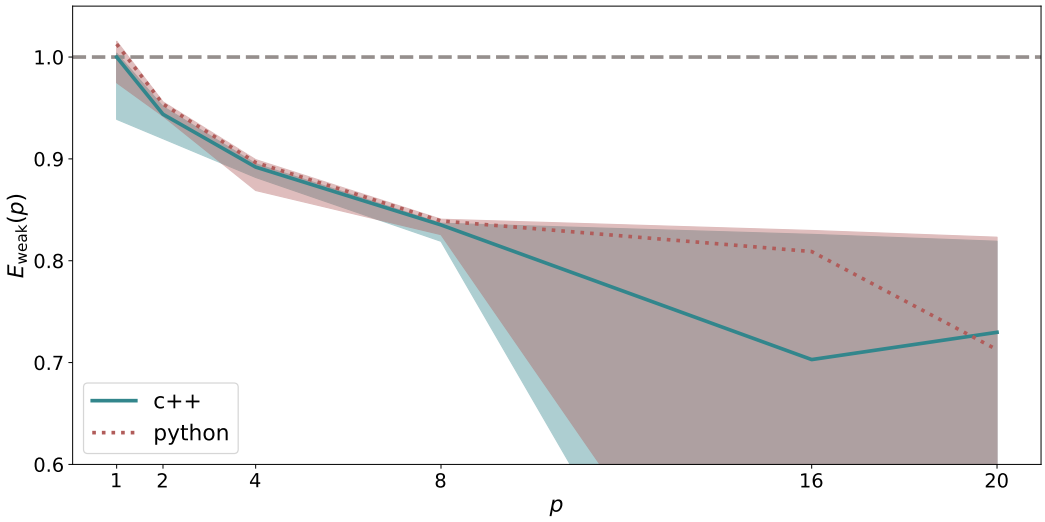

Figure 4.4: Weak-scaling efficiency of the constructor time at thinness growing proportionally to the number of multi-threading processors. The solid line marks the average of 100 runs while the shaded region marks the best and worst result area. We run the tests on a full computing-node of the SISSA Ulysses cluster.

100% efficiency in the lower panel. Even though it is far from being perfect, the speed-up shows a constantly increasing trend. The efficiency seams to get constant around the 60% for $p \geq 16$, but a larger parallel system would be necessary for getting a more precise measurement.

To conclude, in Fig. 4.4, we show the the weak scaling efficiency case. The thinness, at given processors number p, is set to $thin = 50 \cdot p$. As the picture shows, the efficiency seams to become almost constant at $p \gtrsim 8$ for both the C++ and python case, with a value between 70% and 80%.

Chapter 5

Reionizing radiation from ScamPy mock-galaxies

In the context of high redshift cosmology, one of the most compelling open problems is the process of Reionization, that brought the Universe from the optically thick state of the Dark Ages to the transparent state we observe today (see, e.g., Choudhury & Ferrara 2006; Wise 2019, for reviews). Modelling this phase of the Universe evolution is a tricky task, especially using methods tuned to reproduce the observations at low redshift, such as hydrodynamical simulations and semi-analytical models. Instead, if we trust the capability of N-body simulations to capture the evolution of DM haloes up to the highest redshifts, an empirical method such as ScamPy is more likely to correctly predict the distribution of sources.

We expect Reionization to occur as a non-homogeneous process in which patches of the Universe ionize and then merge, prompted by the formation of the first luminous sources (Barkana & Loeb 2001). In order to map the spatial distribution of these ionized bubbles to the underlying dark matter distribution we apply our method to reproduce the observations of eligible candidates for the production of the required ionizing photon budget. It is commonly accepted that the primary role in the production of ionizing photons at high redshift has been played by primordial, star forming galaxies (Shapiro & Giroux 1987; Miralda-Escude & Ostriker 1990; Barkana & Loeb 2001; Steidel et al. 2001). The best candidates for these objects are Lyman-break galaxies (LBGs) which are selected in surveys using their differing appearance in several imaging filters, due to the position of the Lyman limit (Steidel et al. 2001; Lapi et al. 2017; Steidel et al. 2018; Matthee et al. 2018).

The first galaxies that started to inject ionizing radiation in the intergalactic medium were hosted in small DM haloes with masses up to a minimum of $10^8\ M_\odot/h$. In order to paint a population of LBGs on top of a DM simulation, we need, first of all, a high resolution N-body simulation to provide the halo/sub-halo hierarchy required by ScamPy. We therefore run the FoF and SUBFIND algorithms on top of 25 snapshots in the redshift range $4 \leq z \leq 10$ with thinness $\Delta z = 0.25$. The DM snapshots have been obtained by running the (non-public) P-GADGET-3 N-body code (which is derived from the

GADGET-2 code, Springel 2005) on the two simulation dubbed `highres` and `midres` in Table 3.2.

Simulating the reionization process in more detail would required larger simulations at the same level of mass resolution as obtained for the `highres` and `midres` catalogues of our sample, the computational cost becoming out of reach for the test we want to perform here. Overcoming the computational cost of N-body simulations could be obtained by applying up-sampling techniques of sub-grid modelling. Such methods use a low resolution density field and build mock halo catalogues either by matching the theoretical predictions of the halo mass function de la Torre & Peacock (2013); Angulo et al. (2014) or the bias evolution in time (Nasirudin et al. 2019). This goes beyond the aim of the test-bench application we want to present here and we reserve to exploit it in future extensions of this work.

To set the occupation probabilities parameters with the likelihood in Eq. (3.1), we need the 1- and 2-point statistics of LBGs at high redshift. The observational constrains of these high redshift statistics, due to the high distances involved, are not sufficiently tight. We have therefore extrapolated the available measures as follows:

- in Bouwens et al. (2019) the luminosity function of LBGs is fitted up to redshift $z = 10$ and $M_{\rm UV} \approx -16$. We extrapolate this fit up to $M_{\rm UV} = -13$ and integrate to obtain an estimate of the number density of LBGs at high redshift.

- Harikane et al. (2016) provide measurements of the angular 2-point correlation function in the redshift range $4 \leq z \leq 7$. We assume the clustering at redshift $z \geq 7$ to be constant and equal to the measurement obtained for redshift $z = 7$.

Both the observables assumed above are simplistic approximations which are though sufficient for test-benching our model. We will investigate further the limits imposed by the lack of statistics at high redshift in future extensions of this work.

Once the parameters of the model have been set for each redshift, we run the algorithm described in Sec. 3.2.1 and obtain a set of LBG mock catalogues. As a first approximation, let us define a neutral hydrogen distribution on top of each of our snapshots and consider the ionized region that should form around each source of our mock catalogues. Each mock-LBG in our simulation is producing an amount of ionizing photons which is proportional to its UV luminosity, $M_{\rm UV}$. Namely, the rate of ionizing photons that escape from each UV source is

$$\dot{N}_{\rm ion}(M_{\rm UV}) \approx f_{\rm esc}\ k_{\rm ion}\ {\rm SFR}(M_{\rm UV}) \tag{5.1}$$

where $k_{\rm ion} \approx 4 \times 10^{53}$ is the number of ionizing photons ${\rm s}^{-1}(M_\odot/{\rm yr})^{-1}$, with the quoted value appropriate for a Chabrier initial mass function (IMF), $f_{\rm esc}$ is the average escape fraction for ionizing photons from the interstellar medium of high-redshift galaxies (see, e.g. Mao et al. 2007; Dunlop et al. 2013; Robertson et al. 2015; Lapi et al. 2017; Chisholm et al. 2018; Steidel et al. 2018; Matthee et al. 2018), and $\log({\rm SFR}(M_{\rm UV})) \approx -7.4 - 0.4\ M_{\rm UV}$ is the star formation rate of each source. The volume of the Strömgren sphere,

that forms around each mock-LBG, is then given by

$$V_{\rm S} \equiv \frac{\dot{N}_{\rm ion}(M_{\rm UV})}{\overline{n}_H(z)}\, t_{\rm rec}\, (1 - e^{-t/t_{\rm rec}}) \tag{5.2}$$

where $\overline{n}_H(z) \approx 2 \times 10^{-7}\ (\Omega_b(z)h^2/0.022)\ {\rm cm}^{-3}$ is the mean comoving hydrogen number density at given redshift while t is the cosmic time at given redshift and $t_{\rm rec}$ is the cosmic time at the epoch the source started producing a steady flux of ionizing photons.

We make the following simplistic assumptions:

- at each snapshot we do not provide any information about the previous reionization history: at each redshift sources have to completely ionize the medium and the value of $t_{\rm rec}$ is fixed at the cosmic time corresponding to $z = 20$.
- the escape fraction is set to $f_{\rm esc} = 0.1$, which is a conservative value with respect to what recent observations suggest.

With the aforementioned simplifications, we can build spheres around each source at each redshift, therefore producing an approximated map of the ionization state of our snapshots, without having to rely on radiative transfer. In Figure 5.1 we show the projection along one dimension of 4 snapshots at redshift $z = 10, 8, 6$ and 4 for the `highres` simulation. To get the point-by-point ionization fraction we divide our snapshots into voxels of fixed size. If a voxel is embedded within the Strömgren sphere belonging to some source, it is set as ionized, otherwise it is considered neutral. Voxels that lie in the overlapping of two or more Strömgren spheres are counted only once. This further approximation implies *loosing* an amount of ionizing photons which is proportional to the overlapping volume of the whole simulation box. This approximation mainly affects the lower redshifts, as shown in the next Section. In Figure 5.1 we set the voxel-side to 0.25 Mpc/h, resulting in 128^3 voxels in total.

In the remaining part of this Section we will show the results of some measurements that can be obtained from these mock ionization snapshots.

5.1 Ionized fraction measurement

At each redshift, we measure the ionization fraction resulting from our pipeline by counting the number of voxels marked as ionized over the total number of voxels in which the snapshot is divided. The results for both the `midres` and the `highres` simulations are marked with empty squares and red circles, respectively, in Figure 5.2. We compare our measurement with models of reionization history from recent literature.

The gray shaded region shows the tanh-model used in Planck Collaboration VI (2018) while the orange one delimits the prediction of the same model with a larger value of the parameter that regulates the steepness of the ionization fraction evolution ($\Delta_z = 1.5$ instead of $\Delta_z = 0.5$, as from Lewis 2008). The solid green line shows the model from Kulkarni et al. (2019) which is obtained by computing with the ATON code (Aubert & Teyssier 2008, 2010) the radiative transfer a-posteriori on top of a gas density distribution

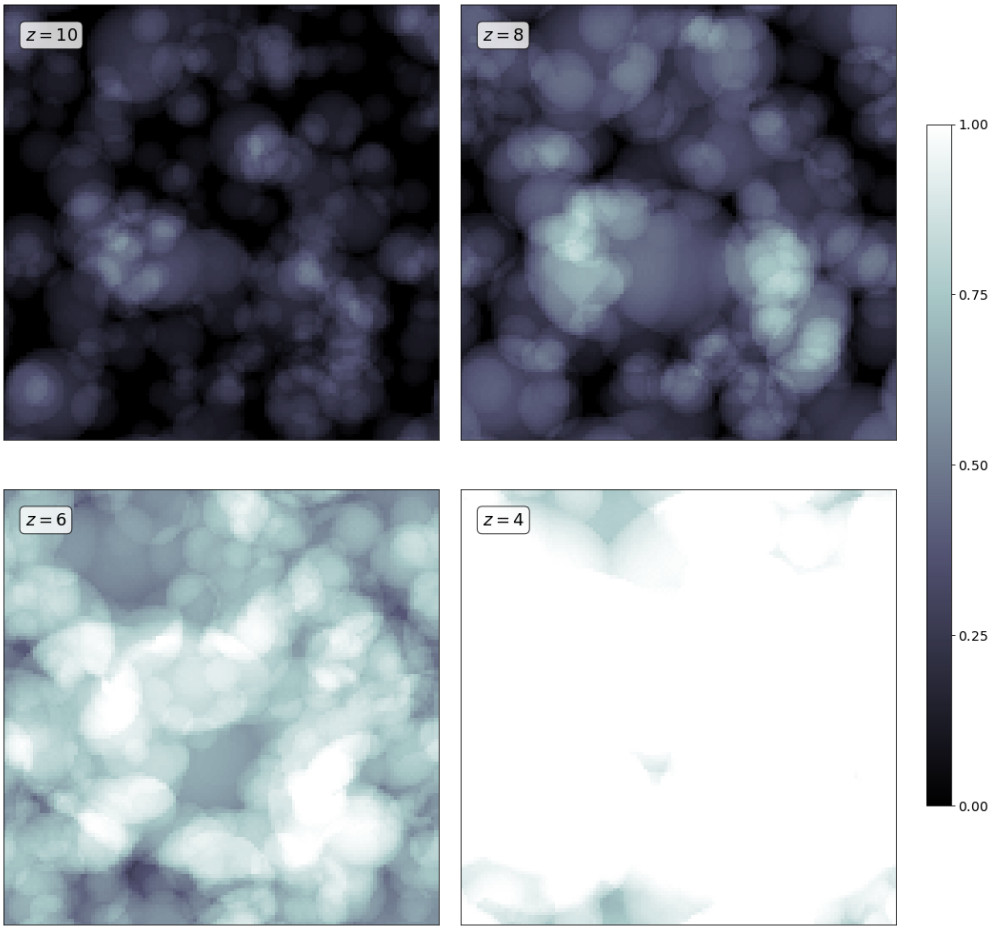

Figure 5.1: Four snapshots at different redshift (shown in the top left angle of each tile) of the ionization fraction obtained by projecting the values in each voxel along one dimension. The value of X_{re} increases from darker to lighter shades of gray.

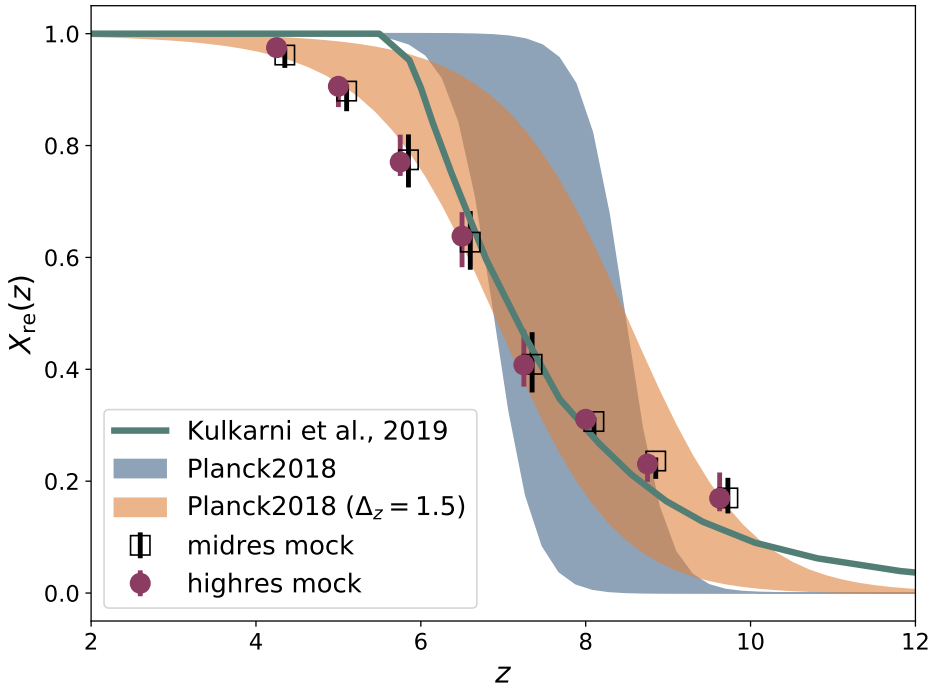

Figure 5.2: Evolution of the hydrogen ionization fraction $X_{\rm re}$ with redshift. Red circles and empty squares mark the measurements obtained with our method on the `midres` and `highres` simulation, respectively. The `midres` distribution has been shifted with an offset of $\delta z = 0.1$ along the x-axis direction to better distinguish it from the `highres` one. The shaded regions delimit the model used in Planck Collaboration VI (2018) and a modification for widening the reionization window. The solid line shows the prediction from Kulkarni et al. (2019).

obtained using the P-GADGET-3 code with the QUICK_LYALPHA approximation from Viel et al. (2004).

Our mock ionization boxes predict reionization to reach half-completion ($X_{\rm re} = 0.5$) at redshift $z = 6.88^{+0.12}_{-0.13}$, which is a lower value with respect to the Planck Collaboration VI (2018) prediction of $z = 7.68 \pm 0.79$, but still within the error bars. Comparing to the extremely steep model used in Planck Collaboration VI (2018), the evolution in our simulations is way shallower, closer to the lower limit of the modified tanh-model. Nonetheless, our measurements seem to agree fairly well with the measurements of Kulkarni et al. (2019) up to redshift $z \approx 6$. With respect to the other authors, our simulation reaches completeness (i.e. $X_{\rm re} = 1$) at redshift $z \approx 4$. As anticipated, this issue at the lowest redshifts is by some extent expected. The overall ionizing photon budget is indeed under-estimated in our approximation as the result of how we treat the overlapping region between different Strömgren spheres. We plan to address this point, by implementing a physically motivated treatment of these overlapping regions (as, e.g., in Zahn et al. 2007) in future extensions of our analysis.

Taking into account the strong approximations made in this proof-of-concept application, the measurement we obtain for the evolution of $X_{\rm re}$ is surprisingly consistent with equivalent measures in literature.

5.2 Ionized bubble size distribution

It is accepted that reionization results from the percolation of ionized HII bubbles as well as from their growth in radius (Miralda-Escude & Ostriker 1990; Furlanetto et al. 2004; Wang & Hu 2006) in the neutral intergalactic medium. A relevant statistics for cosmological studies is the size distribution of the individual bubbles forming around ionizing radiation sources. Obtaining precise measurements of this statistics could help constraining future experiments, such as CMB-S4 (Roy et al. 2018) and 21cm intensity mapping (e.g. Mesinger et al. 2011).

In our framework, getting estimates of the bubble size distribution is straightforward. In Figure 5.3 we present measurements of two different definitions for the bubble size probability.

On the left panel we plot the fraction of bubbles of given size over the total number of bubbles in the simulation box. The measurement has been obtained at redshift $4 \leq z \leq 10$, with bin size $\delta z = 1$, we plot results only for $z = 4, 6, 8$ and 10 for clarity. The distribution shown presents a log-normal shape that we fit with the model from Roy et al. (2018)

$$P(R) = \frac{1}{R} \frac{1}{\sqrt{2\pi\sigma^2_{\ln r}}} \exp\left\{ -\frac{\left[\ln(R/\overline{R})\right]^2}{2\sigma^2_{\ln r}} \right\}; \tag{5.3}$$

the model is regulated by two free parameters: the characteristic bubble size $\overline{R}$ (in Mpc/h) and the standard deviation $\sigma_{\ln r}$. We list the best-fitting values of these parameters in Table 5.1, for the different redshifts considered. While the value of the

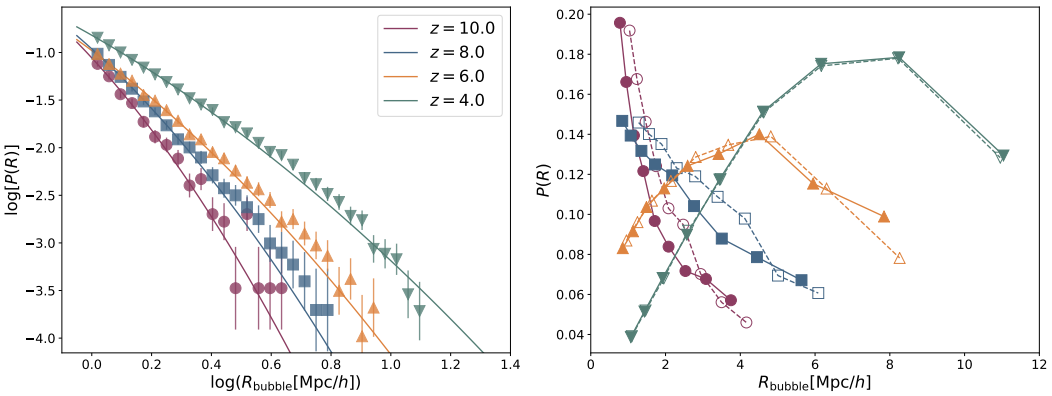

Figure 5.3: Bubble size probability distributions. *Left panel:* fraction of the bubbles with given size over the total number of bubbles, markers are measured from the `highres` simulation while the solid lines show the model of Eq. (5.3) fitted on these data (the best fitting parameters are listed in Table 5.1). *Right panel:* fraction of ionized voxels embedded in bubbles with given size over the total number of ionized voxels. Dashed lines mark the measurements obtained from the `midres` simulation, while solid lines have been obtained from the `highres` simulation.

Table 5.1: Best-fit parameters of the log-normal model defined in Eq. (5.3) obtained from our measures on the `highres` mock ionized bubble catalogue.

z	$\overline{R}$ [Mpc/h]	$\sigma^2_{\ln r}$
10	0.229 ± 0.006	0.614 ± 0.025
9	0.181 ± 0.006	0.786 ± 0.033
8	0.222 ± 0.004	0.810 ± 0.024
7	0.213 ± 0.003	0.974 ± 0.022
6	0.165 ± 0.003	1.340 ± 0.033
5	0.178 ± 0.003	1.616 ± 0.052
4	0.208 ± 0.003	1.983 ± 0.052

characteristic radius is almost constant in time with a value of $\overline{R} \approx 0.2$ Mpc/h, the standard deviation increases significantly from higher to lower redshift.

On the right panel of Figure 5.3, we show the fraction of ionized voxels as a function of the bubble radius over the total number of ionized voxels in the simulation box (normalized to 1). The solid lines show the measurements obtained from the `highres` box, while the dashed ones mark the distribution obtained from the `midres` box. The results on the two boxes are consistent between the two simulations, especially at lower redshifts. Compared to the left panel, the measurements obtained for the bubble size probability definition of the right panel are more consistent with what can be found in literature (e.g. Zahn et al. 2007). In particular, the characteristic radius seems to grow from higher to lower redshift, reaching values in the order of $1 \div 10$ Mpc/h. We could not fit the distributions on the right panel of Figure 5.3 with the same log-normal model of Eq. (5.3). This is probabily due to not having considered bubble overlapping in our measurements. We will investigate further on this topic in future work.

Chapter 6

Summary and discussion

I have here presented ScamPy, our application for painting observed populations of objects on top of DM-only N-body cosmological simulations. With the provided python interface, users can load and populate DM haloes and sub-haloes obtained by means of the FoF and SUBFIND algorithms applied to DM snapshots at any redshift. We foresee to extend this framework to the usage with DM halo and sub-halo catalogues obtained with alternative algorithms.

The main requirements that guided the design of ScamPy were to provide a flexible and optimized framework for approaching a wide variety of problems, while keeping the computation fast and efficient. To this end, we stick to the simple, yet physically robust, SCAM prescription for providing the recipe to populate DM haloes and sub-haloes. The development and testing of the application is treated in Chapter 3: in Section 3.1, I have presented an overview of the theoretical background of this methodology, while, in Section 3.2 I have provided a detailed description of the components and main algorithms implemented in ScamPy.

In Section 3.3 I have shown a set of measurements obtained from simulations populated with galaxies using ScamPy. We have demonstrated that the output mock-galaxies have the expected abundance and clustering properties. We have also proven that the same API could be used to "paint" multiple populations on top of the same DM simulation and that the cross-correlation between these populations is also well mimicked. Finally, we have tested the framework against a real galaxy catalogue in Section 3.4.

A discussion on the source code architecture and performances is provided in Chapter 4. In this part of the manuscript I have focused on the strictly computational aspects of our work. We first discuss the design choices made in the development of ScamPy (Section 4.1) and then provide some benchmark measurements of the application performances in its most critical sections (Section 4.2).

In the context of reionization, empirical modelling can be used to study the spatial distribution properties and evolution in time of the ionized bubbles that shall have developed around sources of ionizing radiation. In Chapter 5, we have performed a preliminary study, under simplistic assumptions, on the ionization properties of the high redshift Universe which result from the injection in the medium of ionizing photons from

Lyman Break Galaxies (LBG). We are able to measure locally on simulations the ionized hydrogen filling factor at different redshifts. This also allows to perform a tomographic measure of the ionization state of the medium at varying cosmic time. Furthermore, we can also directly measure the ionized bubble size distribution, which is a quantity that, up to now, has been either modelled indirectly (Roy et al. 2018) or measured assuming radiative transfer (Zahn et al. 2007).

While a specific problem prompted the development of the API, extensibility has been a crucial design choice. ScamPy features a modular structure exploiting Object-Oriented programming, both in C++ and in python. With a wise usage of polymorphism, we have obtained a flexible application that can be both used by itself as well as along with other libraries.

We are working on adding to the API further miscellaneous functionalities, such as the possibility to download and install it with both `pip` and `conda`. The online documentation is continuously updated, and more examples and tutorials are ready to be uploaded.

To conclude, we are planning to extend our work both on the scientific side, by exploring new directions, and on the computational side, by implementing an efficient k^d-tree algorithm for the optimisation of the neighbour search in both DM-halo catalogues and mock-galaxy catalogues. A possible list of the directions we are planning to pursue follows.

- *Application of the algorithm to multiple populations/different tracers of the LSS* - An application of the same pipeline to the other major players that are known to be involved on the reionization of hydrogen, e.g. AGNs, would be trivial, provided that suitable datasets are available. Another possibility would be the cross-correlation with intensity mapping like e.g. Spinelli et al. (2019).

- *Application to the reionization process* - Up to now we have mainly applied ScamPy functionalities to a proof-of-concept framework. Nonetheless, with the dataset at our disposal, an extensive study of the role played by LBG in reionization is possible, provided that larger N-body simulations with high resolution are available. This would require DM halo catalogues on large simulation boxes, with a complete sampling of the lower masses (down to $10^8 M_\odot$). Such catalogues could be obtained by running high resolution N-body simulations or, more realistically, by applying a sub-resolution scheme, such as the halo-bias models proposed by Nasirudin et al. (2019).

- *Extension to different cosmological models* - By now all our investigations have been performed assuming a ΛCDM cosmology. To extend these results to other cosmological models would only require modest modifications of the source code and to obtain the corresponding DM-only N-body simulations, similar to high redshift studies performed e.g. in warm dark matter scenarios or massive neutrino cosmologies Maio & Viel (2015); Fontanot et al. (2015).

- *Machine learning extension of the halo occupation model* - Using the halo occupation distribution (HOD) is straightforward and a lot of literature is available on

the topic. This approach, though, comes with the limit that all the properties of the observed population have to be inferred from the mass of the host halo. This is known to be a rough approximation. To overcome this limit, we plan to use, instead, a neural network (NN) model of the host halo occupation properties where the inputs of the NN are a set of known features of the halo/sub-halo hierarchy, such as the local environment around the halo and the dispersion velocity within the halo. Matter density based approaches for extending DM only N-body simulations have already been attempted. Recent efforts in this sense include work from He et al. (2019) which use NNs to predict the non-linear evolution of matter perturbations beyond the Zel'dovic Approximation. In another work, Yip et al. (2019), paint baryons on top of simulations only run with DM. We plan instead to investigate the possibility to post process the baryonic effects on top of DM-only simulations in a halo-based framework.

Appendix A

C++ implementation details

"**Polymorphism** — *providing a single interface to entities of different types. Virtual functions provide dynamic (run-time) polymorphism through an interface provided by a base class. Overloaded functions and templates provide static (compile-time) polymorphism.*"

(Bjarne Stroustrup, father of the C++ language)

We give here some details on the C++ implementation of the algorithm described above. We report on the advanced Object-Oriented Programming (OOP) strategies we adopted to boost performances and allow flexibility of the code. With an extensive use of both static and dynamic polymorphism we reduce code replication to a minimum and keep the code general without using conditional-statements, while preserving flexibility (Sections A.1 and A.2).

Based on these strategies we have also developed a powerful utility module for interpolation (Section A.3) which allowed us to boost dramatically the computation performances while keeping control on the precision of the computation (as we show in Section 4.2).

A.1 The `cosmology` class

We have implemented a templated class for cosmological computations. Since this class embeds all the kernel functions that are necessary for the scientific goal of the API, we extensively tested all its output results, both by comparing them with values found in literature and by collecting equivalent measures from another library developed for the same purpose (Marulli et al. 2016).[1]

There are indeed several public implementations of the cosmological functions needed in our work. We have implemented our version, for two main reasons

- performances: all the operations required are implemented in the most computationally efficient way, minimizing the ratios and calls to costly functions. Further accelaration is achieved by using our `interpolator` type.

[1] CosmoBolognaLib website

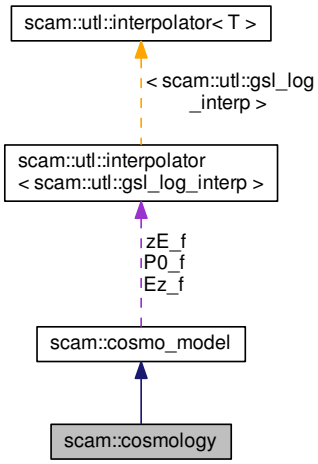

Figure A.1: Call-graph of the `cosmology` class in our default implementation. The solid blue arrow is used to visualize a public inheritance relation between two classes. A purple dashed arrow is used if a class is contained or used by another class. The arrow is labeled with the variables through which the pointed class or struct is accessible. A yellow dashed arrow denotes a relation between a template instance and the template class it was instantiated from. The arrow is labeled with the template parameters of the instance. *(Obtained with Doxygen)*

- for installation simplicity: having our own implementation provides an easy-to-install API which requires the minimum system setup possible for working.

Nevertheless, being aware of the wide range of public libraries serving the same purpose, we decided to keep our cosmology interface open for working with other implementations. The `cosmology` class is indeed derived from a base `cosmo_model` type, which constitutes a C++ interface to our implementation or third-party libraries providing the same functionalities.

In Fig. A.1 we show the call graph of our default implementation of the `cosmology` class. It inherits (solid blue arrow) from our `cosmo_model` structure which has access (dashed purple arrow) to the `interpolator` class through the private variables `Ez_f`, `P0_f` and `zE_f`. The dashed yellow arrow marks the relation between the template instance `interpolator< gsl_log_interp >` and the template class `interpolator< T >` from which it was instantiated.

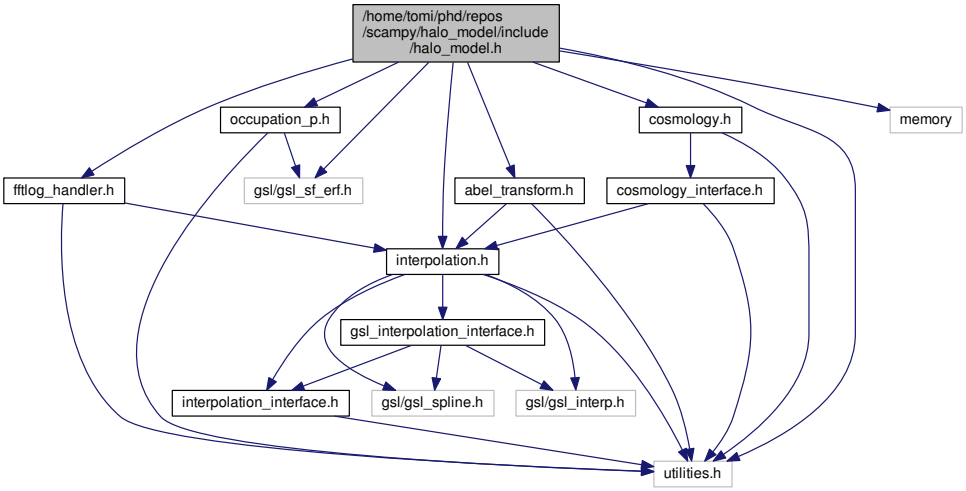

Figure A.2: Dependency graph of the `halo_model.h` header. The arrows mark calls to internal or external headers. *(Obtained with Doxygen)*

A.2 The `halo_model` class

This class provides the user with the full implementation of all the functions presented in Section 3.1.1. Since it provides methods and functions that have to be called tens of thousands of times, most of the C++ section of the library has been developed in order to make this class work with the best possible performances. Fig. A.2 shows the dependency tree of the `halo_model` header. It can be noticed that it is directly connected with almost all the other headers provided by our `c++` implementation. Furthermore, the only two external headers included that are not part of the C++ standard are actually required by the `interpolator` class which is defined in the `interpolation.h` header. We describe this core functionality of our API in Section A.3.

The constructor of our `halo_model` class takes two mandatory arguments:

- an object of type `cosmology` which provides the cosmological functions required by the halo model;
- an object of type `occupation_p`, which provides the shape of the occupation probabilities of central, $N_{\rm cen}$, and satellite, $N_{\rm sat}$, objects.

The latter is again a polymorphic custom type, an `occupation_p` base class provides a set of pure virtual functions which have to be overloaded in the derived classes. In ScamPy's default implementation there are two different derived classes, which define slightly different parameterisations for the HOD algorithm (Harikane et al. 2016; Tinker & Wetzel 2010). We show the inheritance graph of these custom types in Fig A.3.

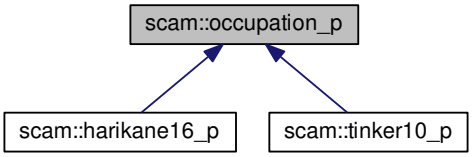

Figure A.3: Inheritance graph of the classes providing the occupation probability functions which are already defined in the default implementation of ScamPy. *(Obtained with Doxygen)*

Member functions of the `halo_model` type provide the models needed by the likelihood defined in Eq. (3.1). To infer the best-fitting parameterisation we have to run a MCMC algorithm in order to sample the parameter space. This means to compute the models tens of thousands of times. In order to speed these calculations we define at the time of construction of the `halo_model` object a large number of interpolated functions. This operation makes construction computationally expensive: the constructor time scales with the thinness required for the interpolation grid. It takes approximately 2 seconds on a Intel i7 laptop with four physical processor and hyper-threading disabled to compute on a grid with 200 cells. The advantage of this setup though, is that we have to call the constructor only one time per run. Anyways, to speed up the constructor, we are computing the values of the functions in the interpolation grid by spawning multiple threads using the OpenMP API.

On the other hand, the computation of the models needed by the likelihood of Eq. 3.1 becomes extremely fast: a full-model computation runs in approximately 3 milliseconds. This makes it also possible to run a full MCMC parameter space sampling on a regular laptop in a reasonable amount of time. Sampling a 4-dimensional parameter space with 8 walkers for 10^5 chain-steps, takes around 5 minutes. We have widely benchmarked the performances of this class, results are shown in Section 4.2.

A.3 The `interpolator` class

The `c++` interpolation module we have developed is probably the most complex piece of code in all the API. It is deeply soaked in C++ OOP and mixes a variety of techniques to work. This complexity though allowed us to reach outstanding optimization and speed-up. The effort spent in designing and implementing this functionality definitely paid off.

The basic idea driving our implementation is to have a tool for making computations faster while maintaining the instrument manageable. To achieve the speed-up necessary for running MCMC samplers we needed fast computations of complex functions and, possibly, a fast integration scheme. Using a polynomial interpolation algorithm, such as

a cubic spline, offers the following advantages:

- functions are defined by couples of arrays storing the x- and y-axis values of the function within some interval and with thinness, therefore precision, defined by the user;
- by using a polynomial interpolation, all the integrals reduce to an analytical form, thus not requiring any particular algorithm to be solved: the time to integrate a function inside a given interval becomes virtually zero;
- inversion of a continuous function also becomes trivial, since it only requires to switch the arrays storing the x- and y-axis and rebuild the interpolator. As a consequence, also zero finding becomes trivial since it requires to first invert the function and then evaluate it at zero.

The user interface to the functionalities of the module is provided by a class `interpolator`, defined in the header

`scampy/interpolation/include/interpolation.h`

accessible from the GitHub repository. This templated class provides, by using static-polymorphism, a set of functions and overloaded operators to make all the operations extremely manageable. The crucial member functions are

- overload of the function call operator (`operator()`);
- overload of the algebraic operators (`operator+`, `operator*` and the equivalent in-place operators);
- function `integrate( min, max )` for interpolated integration in the interval [`min`, `max`].

With this interface the possibilities are huge, consider for example an integrand such as one of those defined in Section 3.1.1. If we have interpolators defined for each of the functions composing the integrand, we can obtain the interpolator for the integrand by calling only algebraic operators and then compute the integral by simply calling the dedicated member function.

The only disadvantage of this custom type is that it requires to compute in advance functions on a grid of given thinness. Even though in certain cases this might not be efficient, this operation has to be performed only once. Therefore this method easily provides a huge speed-up when said functions have to be called over and over again, for example when infering the values of some function parameters in a MCMC scheme.

The `interpolator` class is templated on the `interpolation_interface` type (Fig. A.4, left panel). The latter is a pure virtual polymorphic type and is defined in the header

`scampy/interpolation/include/interpolation_interface.h`

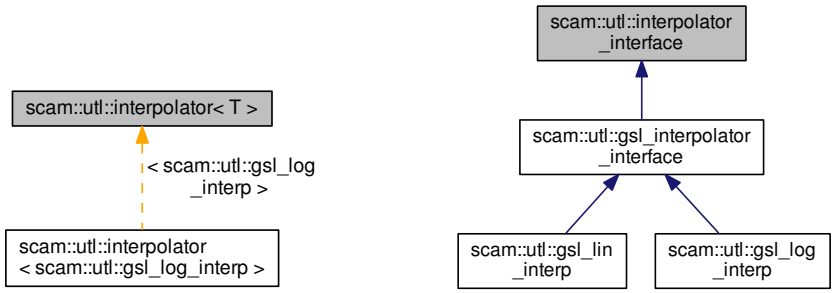

Figure A.4: *Left panel:* relation between the default template instance of the `interpolator` class and the template class it is instantiated from. *Right panel:* Inheritance graph of the classes derived from the `interpolator_interface` custom type. Color codes are the same of Fig. A.1. *(Obtained with Doxygen)*

accessible from the GitHub repository. This makes inheritance from `interpolator_interface` mandatory, defining a precise framework for the usage of our `interpolator` object. Yet, since the template instance of the `interpolator` type is resolved at compile time, we avoid the runtime overhead due to dynamic polymorphism.

Having a pure virtual base class introduces an additional pointer to the memory load. Nevertheless, even in the application that most severely relies on the instantiation of `interpolator` objects (i.e. the `halo_model` object) the number of instances is tipically in the order of 10^2. Considered these numbers and the gain in performance we obtain, using this set-up is worth the additional burden we introduce in memory. Furthermore, having a polymorphic template type makes the implementation extremely clear and easy to handle.

Types derived from `interpolation_interface` have to overload a set of mandatory functions to perform the construction of the interpolation scheme, the evaluation in a sub-grid position and integration. We provide our API with two different classes of this type:

- `gsl_lin_interp`: to perform interpolation on a linearly equispaced grid;
- `gsl_log_interp`: to perform interpolation on a logarithmically equispaced grid.

Both of these classes inherit from a bridging pure-virtual class called `gsl_interpolator_interface` whose role is to wrap methods from the `gsl_interpolation` library, defined in the headers:

```
gsl/gsl_interp.h
gsl/gsl_spline.h
```

This is one of the few external libraries used in this work and comes with accelerators and several possible interpolation schemes. Given its wide diffusion both among users from

the Cosmology and Astrophysics community and among HPC facilities, the inclusion of this external dependency should not affect the interoperability of our API.

Appendix B

Build system

Automation of the build process involves the development of scripts and the usage of software to compile source code into binary code and/or to install it in the chosen position of the file-system.

When we first started working on our API, the build system was based on a set of Makefiles dedicated to the compilation of each different module of the source code. While the software was growing larger though, we realised that it would be far more maintainable and user-friendly porting everything in a build-script generator.

For the automation of the build process we choose the Meson build system (Meson website 2014). Meson is free and open-source software written in Python, under the Apache License 2.0. It is intended for performance and for boosting the programmer productivity.

Much like `CMake` (CMake website 2003), it does not build directly, but it rather generates files to be used by a native build tool. Meson has been first developed for Linux but, being written in Python, it works on most of the Unix-like operative systems (OS), including macOS, as well as Microsoft Windows and others. This interoperability is also achieved by using different native build tools for different OS: it uses `ninja` (Ninja website 2020 - latest) in Linux, `MSbuild` (MS-Build website 2019 - latest) on Windows and `xcode` (XCode 11 website 2019 - stable) on macOS. It supports a wide range of compilers, including the GNU Compiler Collection, Intel compilers and Clang.

The build is controlled by a master `meson.build` script located in the root directory of the repository. In this file we first specify the basic setup of the build, setting

- the project type, `cpp` in our case;
- some default options to be used at compile time:
  - the C++ standard, i.e. `c++14`;
  - the build type, i.e. `plain`, `debug`, `debugoptimized` or `release`;
  - the directory structure of the compiled library, i.e. the installation prefix and all its eventual subdirectories (lib, include, share).

- the release version and license for the source code;
- the optional compile time flags.

We then list the external dependencies required to build the API, these will be either searched by meson using `pkgconfig` or downloaded from the specified mirrors and installed in the specified position.

Lastly, we go through each sub-directory hosting the different modules composing the API. As described in Section 4.1, each module composing our API is stored in a dedicated directory which comes with a `meson.build` file that provides the build directives proper to the module. All these build scripts have the very same structure, making them easily reproducible in case of extension of the API with new modules and functionalities.

```
###############################################################################
# Internal includes

inc_dir = 'include'
inc_utl = include_directories( inc_dir )

###############################################################################
# Utilities Lib

utl_lib = library( 'utilities',
                   [ 'src/utilities.cpp',
                     'src/bit_manipulator.cpp' ],
                   include_directories : inc_utl,
                   dependencies : [ gsl_dep ],
                   install : true  )

utl_dep = declare_dependency( link_with : utl_lib,
                              include_directories : inc_utl )

lib_headers += [ root + '/utilities/' + inc_dir + '/utilities.h',
                 root + '/utilities/' + inc_dir + '/error_handling.h',
                 root + '/utilities/' + inc_dir + '/bit_manipulator.h' ]

###############################################################################
```

Listing B.1: Example `meson.build` script file for the module `utilities` of ScamPy.

In List. B.1 we show the `meson.build` script file that builds the `utilities` module of our API. It is divided in two sections, the first one specifies the location of the header files and stores it in an object of type `include_directories`. The second one deals with building and installation directives. Firstly, it defines the `library` object, assigning a name, listing all the source files that have to be compiled, and specifing the internal and external dependencies.

It then declares the compiled module as an internal dependency and adds all the header files to the list of files that will be copied in the

`/path/to/install_prefix/include_dir`

sub-directory of the compiled library.

Installation using meson and the native build tool is trivial and as simple as calling the following commands from the repository's root directory:

- Linux and Unix-like systems:

```
$ meson /path/to/build_dir -Dprefix=/path/to/install_prefix
$ ninja -C /path/to/build_dir install
```

- Microsoft Visual Studio:

```
$ py meson /path/to/build_dir --backend vs2015/ninja
```

Note that we have added two configuration options to decide whether to build the API with testing and documentation enabled:

```
-DENABLE_TEST=true/false
-DENABLE_DOC=true/false
```

The default values are set to `false` to speed-up the build and to avoid the necessity of additional external dependencies.

Bibliography

Amendola, L., Appleby, S., Avgoustidis, A., et al. 2018, Living Reviews in Relativity, 21, 2

Angulo, R. E., Baugh, C. M., Frenk, C. S., & Lacey, C. G. 2014, *MNRAS*, 442, 3256

Angulo, R. E., Springel, V., White, S. D. M., et al. 2012, Monthly Notices of the Royal Astronomical Society, 426, 2046

Astropy Collaboration. 2013, *A&A*, 558, A33

Astropy Collaboration. 2018, *AJ*, 156, 123

Aubert, D. & Teyssier, R. 2008, *MNRAS*, 387, 295

Aubert, D. & Teyssier, R. 2010, *ApJ*, 724, 244

Baldi, M. 2012, *MNRAS*, 422, 1028

Baldi, M., Pettorino, V., Robbers, G., & Springel, V. 2010, *MNRAS*, 403, 1684

Baldry, I. K., Alpaslan, M., Bauer, A. E., et al. 2014, *MNRAS*, 441, 2440

Baldry, I. K., Robotham, A. S. G., Hill, D. T., et al. 2010, *MNRAS*, 404, 86

Barkana, R. & Loeb, A. 2001, *Phys. Rep.*, 349, 125

Behroozi, P., Wechsler, R. H., Hearin, A. P., & Conroy, C. 2019, *MNRAS*, 488, 3143

Behroozi, P. S., Conroy, C., & Wechsler, R. H. 2010, The Astrophysical Journal, 717, 379

Behroozi, P. S. & Silk, J. 2015, The Astrophysical Journal, 799, 32

Behroozi, P. S., Wechsler, R. H., & Conroy, C. 2012, The Astrophysical Journal, 762, L31

Behroozi, P. S., Wechsler, R. H., & Conroy, C. 2013a, *ApJ*, 770, 57

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

Beltz-Mohrmann, G. D., Berlind, A. A., & Szewciw, A. O. 2019, arXiv e-prints, arXiv:1908.11448

Benson, A. J., Hoyle, F., Torres, F., & Vogeley, M. S. 2003, *MNRAS*, 340, 160

Berlind, A. A. & Weinberg, D. H. 2002, *ApJ*, 575, 587

Bernardeau, F. 1994, *ApJ*, 427, 51

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

Bonavera, L., González-Nuevo, J., Cueli, M. M., et al. 2020, *A&A*, 639, A128

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

Bouwens, R. J., Stefanon, M., Oesch, P. A., et al. 2019, *ApJ*, 880, 25

Boylan-Kolchin, M., Springel, V., White, S. D. M., Jenkins, A., & Lemson, G. 2009, Monthly Notices of the Royal Astronomical Society, 398, 1150

Brown, M. J. I., Zheng, Z., White, M., et al. 2008, The Astrophysical Journal, 682, 937

Chisholm, J., Gazagnes, S., Schaerer, D., et al. 2018, *A&A*, 616, A30

Choudhury, T. R. & Ferrara, A. 2006, arXiv e-prints, astro

CMake website. 2003, CMake reference website, https://cmake.org/

Colberg, J. M., Pearce, F., Foster, C., et al. 2008, *MNRAS*, 387, 933

Colberg, J. M., Sheth, R. K., Diaferio, A., Gao, L., & Yoshida, N. 2005, *MNRAS*, 360, 216

Colless, M., Dalton, G., Maddox, S., et al. 2001, Monthly Notices of the Royal Astronomical Society, 328, 1039

Conroy, C. & Wechsler, R. H. 2009, *ApJ*, 696, 620

Conroy, C., Wechsler, R. H., & Kravtsov, A. V. 2006, The Astrophysical Journal, 647, 201

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

Cooray, A. & Sheth, R. 2002, *Phys. Rep.*, 372, 1

de la Torre, S. & Peacock, J. A. 2013, *MNRAS*, 435, 743

Diemer, B. 2017, The Astrophysical Journal Supplement Series, 231, 5

Dolag, K., Komatsu, E., & Sunyaev, R. 2016, *MNRAS*, 463, 1797

Driver, S. P., Hill, D. T., Kelvin, L. S., et al. 2011, Monthly Notices of the Royal Astronomical Society, 413, 971

Dunlop, J. S., Rogers, A. B., McLure, R. J., et al. 2013, *MNRAS*, 432, 3520

Elyiv, A., Marulli, F., Pollina, G., et al. 2015, *MNRAS*, 448, 642

Fontanot, F., Villaescusa-Navarro, F., Bianchi, D., & Viel, M. 2015, *MNRAS*, 447, 3361

Foreman-Mackey, D. et al. 2013, *PASP*, 125, 306

Furlanetto, S. R. & Piran, T. 2006, *MNRAS*, 366, 467

Furlanetto, S. R., Zaldarriaga, M., & Hernquist, L. 2004, *ApJ*, 613, 1

Galassi, M. et al. 2009, GNU Scientific Library Reference Manual - Third Edition

Gardner, J. P., Mather, J. C., Clampin, M., et al. 2006, *Space Sci. Rev.*, 123, 485

González-Nuevo, J., Lapi, A., Bonavera, L., et al. 2017, *J. Cosmology Astropart. Phys.*, 2017, 024

Grogin, N. A., Kocevski, D. D., Faber, S. M., et al. 2011, The Astrophysical Journal Supplement Series, 197, 35

Guo, H., Zheng, Z., Behroozi, P. S., et al. 2016, *MNRAS*, 459, 3040

Guo, Q., White, S., Li, C., & Boylan-Kolchin, M. 2010, Monthly Notices of the Royal Astronomical Society, 404, 1111

Hadzhiyska, B., Bose, S., Eisenstein, D., Hernquist, L., & Spergel, D. N. 2019, arXiv e-prints, arXiv:1911.02610

Hamaus, N., Pisani, A., Sutter, P. M., et al. 2016, Physical Review Letters, 117, 091302

Hamaus, N., Sutter, P. M., Lavaux, G., & Wandelt, B. D. 2015, *J. Cosmology Astropart. Phys.*, 11, 036

Hamaus, N., Sutter, P. M., & Wandelt, B. D. 2014, Physical Review Letters, 112, 251302

Hamilton, A. J. S. 2000, *MNRAS*, 312, 257

Harikane, Y., Ouchi, M., Ono, Y., et al. 2016, *ApJ*, 821, 123

He, S., Li, Y., Feng, Y., et al. 2019, Proceedings of the National Academy of Science, 116, 13825

Hearin, A. P., Campbell, D., Tollerud, E., et al. 2017, *AJ*, 154, 190

Icke, V. 1984, *MNRAS*, 206, 1P

Imara, N., Loeb, A., Johnson, B. D., Conroy, C., & Behroozi, P. 2018, The Astrophysical Journal, 854, 36

Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, *ApJ*, 873, 111

Jasche, J. & Lavaux, G. 2019, *A&A*, 625, A64

Jennings, E., Li, Y., & Hu, W. 2013, *MNRAS*, 434, 2167

Kitaura, F.-S., Ata, M., Rodriguez-Torres, S. A., et al. 2019, arXiv e-prints, arXiv:1911.00284

Klypin, A., Yepes, G., Gottlöber, S., Prada, F., & Heß, S. 2016, Monthly Notices of the Royal Astronomical Society, 457, 4340

Klypin, A. A., Trujillo-Gomez, S., & Primack, J. 2011, The Astrophysical Journal, 740, 102

Komatsu, E., Smith, K. M., Dunkley, J., et al. 2011, *ApJS*, 192, 18

Kreisch, C. D., Pisani, A., Carbone, C., et al. 2019, *MNRAS*, 1877

Kulkarni, G., Keating, L. C., Haehnelt, M. G., et al. 2019, *MNRAS*, 485, L24

Landy, S. D. & Szalay, A. S. 1993, *ApJ*, 412, 64

Lapi, A., Mancuso, C., Celotti, A., & Danese, L. 2017, *ApJ*, 835, 37

Lavaux, G. & Wandelt, B. D. 2012, *ApJ*, 754, 109

Leauthaud, A., George, M. R., Behroozi, P. S., et al. 2012, The Astrophysical Journal, 746, 95

Leclercq, F., Jasche, J., & Wandelt, B. 2015, *J. Cosmology Astropart. Phys.*, 2015, 015

Levi, M., Bebek, C., Beers, T., et al. 2013, arXiv e-prints, arXiv:1308.0847

Lewis, A. 2008, *Phys. Rev. D*, 78, 023002

Li, B., Zhao, G.-B., & Koyama, K. 2012, *MNRAS*, 421, 3481

Lilly, S. J., Fevre, O. L., Renzini, A., et al. 2007, The Astrophysical Journal Supplement Series, 172, 70

Limber, D. N. 1953, *ApJ*, 117, 134

Liske, J., Baldry, I. K., Driver, S. P., et al. 2015, *MNRAS*, 452, 2087

Maio, U. & Viel, M. 2015, *MNRAS*, 446, 2760

Mao, J., Lapi, A., Granato, G. L., de Zotti, G., & Danese, L. 2007, The Astrophysical Journal, 667, 655

Marulli, F., Veropalumbo, A., & Moresco, M. 2016, Astronomy and Computing, 14, 35

Massara, E., Villaescusa-Navarro, F., Viel, M., & Sutter, P. M. 2015, *J. Cosmology Astropart. Phys.*, 11, 018

Matthee, J., Sobral, D., Gronke, M., et al. 2018, *A&A*, 619, A136

McCracken, H. J., Milvang-Jensen, B., Dunlop, J., et al. 2012, A&A, 544, A156

Mesinger, A., Furlanetto, S., & Cen, R. 2011, *MNRAS*, 411, 955

Meson website. 2014, The Meson Build system, https://mesonbuild.com/index.html

Miralda-Escude, J. & Ostriker, J. P. 1990, *ApJ*, 350, 1

Mo, H. J. & White, S. D. M. 1996, Monthly Notices of the Royal Astronomical Society, 282, 347

Monaco, P. & Efstathiou, G. 1999, Monthly Notices of the Royal Astronomical Society, 308, 763

Monaco, P., Theuns, T., & Taffoni, G. 2002, *MNRAS*, 331, 587

Moster, B. P., Naab, T., & White, S. D. M. 2013, *MNRAS*, 428, 3121

Moster, B. P., Naab, T., & White, S. D. M. 2018, *MNRAS*, 477, 1822

Moster, B. P., Somerville, R. S., Maulbetsch, C., et al. 2010, The Astrophysical Journal, 710, 903

MS-Build website. 2019 - latest, Microsoft Build Engine, https://docs.microsoft.com/it-it/visualstudio/msbuild/msbuild?view=vs-2019

Naab, T. & Ostriker, J. P. 2017, Annual Review of Astronomy and Astrophysics, 55, 59

Nadathur, S. & Hotchkiss, S. 2015, *MNRAS*, 454, 2228

Nasirudin, A., Iliev, I. T., & Ahn, K. 2019, arXiv e-prints, arXiv:1910.12452

Nelson, D., Springel, V., Pillepich, A., et al. 2019, Computational Astrophysics and Cosmology, 6, 2

Neyrinck, M. C. 2008, *MNRAS*, 386, 2101

Ninja website. 2020 - latest, Ninja reference website, https://ninja-build.org/

Nuza, S. E., Kitaura, F.-S., Heß, S., Libeskind, N. I., & Müller, V. 2014, Monthly Notices of the Royal Astronomical Society, 445, 988

Paranjape, A., Lam, T. Y., & Sheth, R. K. 2012, *MNRAS*, 420, 1648

Peacock, J. A. & Smith, R. E. 2000, Monthly Notices of the Royal Astronomical Society, 318, 1144

Pisani, A., Sutter, P. M., Hamaus, N., et al. 2015, *Phys. Rev. D*, 92, 083531

Planck Collaboration VI. 2018, arXiv e-prints, arXiv:1807.06209

Pollina, G., Baldi, M., Marulli, F., & Moscardini, L. 2016, *MNRAS*, 455, 3075

Pollina, G., Hamaus, N., Dolag, K., et al. 2017, *MNRAS*, 469, 787

Pollina, G., Hamaus, N., Paech, K., et al. 2019, *MNRAS*, 487, 2836

Popping, G., Behroozi, P. S., & Peeples, M. S. 2015, Monthly Notices of the Royal Astronomical Society, 449, 477

Ricciardelli, E., Quilis, V., & Planelles, S. 2013, *MNRAS*, 434, 1192

Ricciardelli, E., Quilis, V., & Varela, J. 2014, *MNRAS*, 440, 601

Robertson, B. E., Ellis, R. S., Furlanetto, S. R., & Dunlop, J. S. 2015, *ApJL*, 802, L19

Rodríguez-Puebla, A., Primack, J. R., Avila-Reese, V., & Faber, S. M. 2017, *MNRAS*, 470, 651

Rodríguez-Puebla, A., Behroozi, P., Primack, J., et al. 2016, Monthly Notices of the Royal Astronomical Society, 462, 893

Ronconi, T., Contarini, S., Marulli, F., Baldi, M., & Moscardini, L. 2019, Monthly Notices of the Royal Astronomical Society, 488, 5075

Ronconi, T., Lapi, A., Viel, M., & Sartori, A. 2020a, *MNRAS*

Ronconi, T., Lapi, A., Viel, M., & Sartori, A. 2020b, ScamPy: Sub-halo Clustering and Abundance Matching Python interface

Ronconi, T. & Marulli, F. 2017, *A&A*, 607, A24

Roy, A., Lapi, A., Spergel, D., & Baccigalupi, C. 2018, *J. Cosmology Astropart. Phys.*, 2018, 014

Schaye, J., Crain, R. A., Bower, R. G., et al. 2015, *MNRAS*, 446, 521

Seljak, U. 2000, Monthly Notices of the Royal Astronomical Society, 318, 203

Shapiro, P. R. & Giroux, M. L. 1987, *ApJL*, 321, L107

Sheth, R. K. & van de Weygaert, R. 2004, *MNRAS*, 350, 517

Sinha, M., Berlind, A. A., McBride, C. K., et al. 2018, *MNRAS*, 478, 1042

Somerville, R. S., Behroozi, P., Pandya, V., et al. 2018, *MNRAS*, 473, 2714

Somerville, R. S. & Davé, R. 2015, Annual Review of Astronomy and Astrophysics, 53, 51

Spinelli, M., Zoldan, A., De Lucia, G., Xie, L., & Viel, M. 2019, arXiv e-prints, arXiv:1909.02242

Springel, V. 2005, *MNRAS*, 364, 1105

Springel, V., White, S. D. M., Jenkins, A., et al. 2005, *Nature*, 435, 629

Springel, V., White, S. D. M., Tormen, G., & Kauffmann, G. 2001, Monthly Notices of the Royal Astronomical Society, 328, 726

Steidel, C. C., Bogosavljević, M., Shapley, A. E., et al. 2018, *ApJ*, 869, 123

Steidel, C. C., Pettini, M., & Adelberger, K. L. 2001, *ApJ*, 546, 665

Sutter, P. M., Lavaux, G., Hamaus, N., et al. 2015, Astronomy and Computing, 9, 1

Sutter, P. M., Lavaux, G., Wandelt, B. D., & Weinberg, D. H. 2012, *ApJ*, 761, 44

Sutter, P. M., Pisani, A., Wandelt, B. D., & Weinberg, D. H. 2014, *MNRAS*, 443, 2983

Tassev, S., Zaldarriaga, M., & Eisenstein, D. J. 2013, *J. Cosmology Astropart. Phys.*, 2013, 036

Tinker, J. L., Weinberg, D. H., Zheng, Z., & Zehavi, I. 2005, The Astrophysical Journal, 631, 41

Tinker, J. L. & Wetzel, A. R. 2010, *ApJ*, 719, 88

Trujillo-Gomez, S., Klypin, A., Primack, J., & Romanowsky, A. J. 2011, The Astrophysical Journal, 742, 16

Vale, A. & Ostriker, J. P. 2004, Monthly Notices of the Royal Astronomical Society, 353, 189

Van de Weygaert, R. & Platen, E. 2011, International Journal of Modern Physics: Conference Series, 01, 41

Viel, M., Haehnelt, M. G., & Springel, V. 2004, Monthly Notices of the Royal Astronomical Society, 354, 684

Vogelsberger, M., Genel, S., Springel, V., et al. 2014a, *Nature*, 509, 177

Vogelsberger, M., Genel, S., Springel, V., et al. 2014b, *MNRAS*, 444, 1518

Wang, L., Li, C., Kauffmann, G., & De Lucia, G. 2006, Monthly Notices of the Royal Astronomical Society, 371, 537

Wang, L., Li, C., Kauffmann, G., & De Lucia, G. 2007, Monthly Notices of the Royal Astronomical Society, 377, 1419

Wang, X. & Hu, W. 2006, The Astrophysical Journal, 643, 585

Wechsler, R. H., Gross, M. A. K., Primack, J. R., Blumenthal, G. R., & Dekel, A. 1998, The Astrophysical Journal, 506, 19

Wechsler, R. H. & Tinker, J. L. 2018, *ARA&A*, 56, 435

White, M. 2001, Monthly Notices of the Royal Astronomical Society, 321, 1

White, S. D. M. & Rees, M. J. 1978, *MNRAS*, 183, 341

Wise, J. H. 2019, arXiv e-prints, arXiv:1907.06653

XCode 11 website. 2019 - stable, XCode 11 reference website, https://developer.apple.com/xcode/

Yang, X., Mo, H. J., & van den Bosch, F. C. 2003, *MNRAS*, 339, 1057

Yang, X., Mo, H. J., van den Bosch, F. C., Zhang, Y., & Han, J. 2012, *ApJ*, 752, 41

Yip, J. H. T., Zhang, X., Wang, Y., et al. 2019, arXiv e-prints, arXiv:1910.07813

York, D. G., Adelman, J., John E. Anderson, J., et al. 2000, The Astronomical Journal, 120, 1579

Zahn, O., Lidz, A., McQuinn, M., et al. 2007, *ApJ*, 654, 12

Zehavi, I., Weinberg, D. H., Zheng, Z., et al. 2004, The Astrophysical Journal, 608, 16

Zeldovich, Y. B. 1972, Monthly Notices of the Royal Astronomical Society, 160, 1P

Zentner, A. R. 2007, International Journal of Modern Physics D, 16, 763

Zheng, Z., Berlind, A. A., Weinberg, D. H., et al. 2005, The Astrophysical Journal, 633, 791

Zheng, Z., Coil, A. L., & Zehavi, I. 2007, The Astrophysical Journal, 667, 760

Zheng, Z., Zehavi, I., Eisenstein, D. J., Weinberg, D. H., & Jing, Y. P. 2009, The Astrophysical Journal, 707, 554

Zhu, H., Avestruz, C., & Gnedin, N. Y. 2020, arXiv e-prints, arXiv:2001.02233