Universität Zürich

Institute for Computational Science

# Creating Mock Galaxy Catalogues from Dark Matter Simulations

A thesis presented for the degree of Master of Science in Physics
August 2018

*Author:*
Mladen Ivkovic

*Supervisor:*
Prof. Dr. Romain Teyssier


Universität Zürich[UZH]

# Contents

## Abstract

The implementation of an algorithm to identify dark matter halo merger trees into the adaptive mesh refinement code `RAMSES` is presented. The algorithm is fully parallel using MPI and works on the fly. It tracks dark matter substructure individually through particle IDs, thus allowing to follow the formation history of dark matter clumps up to the point where they dissolve beyond the possibility of identification as substructure. Once a clump merges into another, it is still being tracked by marking the most strongly bound particle of that clump at the last snapshot where it was identified. This allows to check at later snapshots whether the identified merging event truly was one, or whether a misidentification by the density field gradient based clump finding algorithm might have occured. The influence of various definitions of substructure and the maximal number of particles tracked per clump have been tested.

With the known formation history of dark matter structure, galaxies can be introduced in a simulation containing only dark matter particles through use of a parametrised stellar-mass-to-halo-mass (SMHM) relation. In this work, the SMHM relation published by Behroozi, Wechsler, and Conroy 2013 was used. The obtained stellar mass functions of central galaxies from $z \sim 0 - 8$ and correlation functions at $z \sim 0$ are compared to observational data. Taking into account that the simulations used to obtain these mock galaxy catalogues didn't have particularly high resolutions with $512^3$ particles and that the main focus of this work is to demonstrate that a merger tree algorithm and the generation of mock galaxy catalogues can be done on the fly, the results show satisfactory agreement with observational data.

## 1. Motivation

Mock galaxy catalogues, artificial catalogues of galaxies created using numerical simulations, have become indispensable tools for astronomy and cosmology. Usually simulations can give real-space galaxy data, while observational data is measured in redshift-space. By following a light cone through the past, redshift-space catalogues may be generated from simulated real-space data, enabling direct comparisons to observations, possibly aiding in the interpretation thereof. Observational effects and uncertainties can be included in results of simulations more easily than taken out from observations, so by comparing the mocks with observed catalogues one can test theories and assumptions and estimate systematic and statistical errors. Furthermore, mock galaxy catalogues can be used to plan and forecast future surveys and to develop analysis codes for anticipated observational data.

The current concordance model of cosmology, the ΛCDM model, states that the Universe is made up from $\sim 5\%$ baryonic matter, which is what galaxies are made of, $\sim 25\%$ dark matter and $\sim 70\%$ dark energy. Dark matter has fundamentally different properties from baryonic matter: It is collisionless, the only significant interaction it experiences is via gravity. This property makes dark matter easier and cheaper to simulate than baryonic matter, where many physical and hydrodynamical effects such as viscosity, radiation, pressure, heating, cooling and many more need to be taken into account as well. For efficiency, cosmological simulations often neglect baryonic effects in the Universe and replace the baryonic matter with dark matter in order to preserve the total matter content. Such simulations are commonly referred to as '*dark matter only*' (DMO) simulations. With growing processing power, improved algorithms and the use of parallel computing tools and architectures, larger and better resolved DMO simulations are becoming possible. The current state-of-the-art cosmological simulation (Potter, Stadel, and

Teyssier 2017) contained 2 trillion particles. However, in order to obtain mock galaxy catalogues, galaxies somehow need to be re-introduced into DMO simulations. Various concepts to achieve that goal have been developed and used, some of which will be introduced later. Most of them however have in common that they place galaxies in condensates of dark matter, called '*haloes*', where the galaxy's properties depend on the properties of its host halo's properties and formation history. In the hierarchical bottom-up structure formation picture, large haloes are thought to form mainly through consecutive merging events of smaller haloes, which is schematically shown in figure 5. The merger histories can be followed by means of a tree structure, which are commonly referred to as '*merger trees*'. Merger trees are essential to obtain accurate mock galaxy catalogues. Why that is the case will be further elaborated and illustrated at a later point.

While bigger simulations can yield data of unprecedented size and resolution, they also produce large amounts of data which needs to be stored and post-processed effectively. This creates a variety of issues. On one hand there is a possibility that not all produced simulation data can be stored because it is simply too large. Another issue is that most modern astrophysical simulations are executed on large supercomputers which offer large distributed memory. Post-processing the data they produce may also require just as much memory, so that the analysis will also have to be executed on the distributed memory infrastructures. The reading and writing of a vast amount of data to a permanent storage remains a considerable bottleneck, particularly so if the data need to be read and written multiple times. One way to reduce the computational cost is to include analysis tools like halo-finding and the generation of merger trees in the simulations and run them "*on the fly*", i.e. run them during the simulation.

In this work, a new implementation of a merger tree algorithm into the simulation code `RAMSES` (Teyssier, R. 2002), designed to work on the fly and on parallel architectures, as well as mock galaxy catalogues obtained using these merger trees on the fly, are presented and tested. This work is structured as follows: Chapter 2 (partially adapted from Ivkovic 2017) gives a short introduction amongst other topics to cosmological simulations of dark matter, halo finding, merger trees, and obtaining galaxies from DMO simulations. The description of the merger tree algorithm and the results of tests of the resulting from merger trees are in chapter 3. Mock galaxy catalogues obtained through the presented merger tree algorithm are shown and tested in chapter 4. Finally, this work is concluded in chapter 5.

# 2. Introduction

## 2.1. Simulating Dark Matter with RAMSES

A system that contains many particles is called an "*N-body system*". A collisionless N-body system is described by the following equations for the particles with positions $\mathbf{x}_p$ and velocities $\mathbf{v}_p$:

$$\frac{\mathrm{d}\mathbf{x}_p}{\mathrm{d}t} = \mathbf{v}_p \qquad \frac{\mathrm{d}\mathbf{v}_p}{\mathrm{d}t} = -\nabla_x \phi \tag{1}$$

with

$$\Delta_x \phi = 4\pi G \rho \tag{2}$$

Where $\phi$ is the potential, $G$ the gravitational constant and $\rho$ the mass density field. For a collisionless system, the only interaction and therefore the only source of acceleration is gravity.

RAMSES (Teyssier, R. 2002) is a N-body and hydrodynamical adaptive mesh refinement (AMR) code which uses the "*Fully Threaded Tree*" data structure of Khokhlov (Khokhlov 1998). Aside from many other uses, it can simulate the time evolution of particles by numerically integrating equations (1). In the case of dark matter only (DMO) simulations, the matter content of the Universe is taken to consist only of collisionless dark matter, thus neglecting any baryonic physics like star or galaxy formation. The simulation usually consists of a (physically) large box, called the domain, which is assumed to represent a typical chunk of the Universe. According to the cosmological principle, the Universe is homogeneous and isotropic on large enough scales, so if the physical domain is sufficiently large, one may pretend to simulate the entire Universe at once by introducing periodical boundary conditions: If a particle would leave a boundary surface of the domain, it is reintroduced into the system at the opposite surface. This way, one simulates the entire Universe by assuming it consists of an infinite number of identical adjacent boxes.

Particles in the domain represent groups of dark matter particles and usually are all identical. Since the density of the Universe is known, the mass of each particle is determined by the total number of particles in the domain and the physical size of the domain.

The domain is covered by a Cartesian grid, called the mesh. Simulations require numerical integration, whose accuracy increases as the size of a grid cell decreases, but smaller grid cells naturally need more resources to cover a domain of the same size. AMR is a method that "*tries to attain a fixed accuracy for a minimum cost*" (Berger and Colella 1989) by refining the mesh only where and when necessary. "Refining" in this case means that cells of smaller and smaller sizes are introduced until the desired accuracy is achieved. Grid cells that contain lots of particles will be refined many times, while grid cells that contain no particles won't be refined at all.

To compute the spatial movement of the particles, first the density field $\rho$ is computed using a "*Cloud-In-Cell*" (CIC) interpolation scheme. The CIC scheme considers all particles to be cubes ("clouds") of one cell size and of uniform density. The mass of a particle is deposited in cells based on what fraction of its "cloud" overlaps with the cell, thus determining the density field. Once the density field is known, the Poisson equation (2) can be solved numerically and the potential $\phi$ is computed. RAMSES utilises a Multigrid solver. Now the acceleration for each cell can be obtained, from which the particle acceleration $\frac{\mathrm{d}\mathbf{v}_p}{\mathrm{d}t}$ is computed using the inverse CIC interpolation. The acceleration is then integrated over time to compute the particle velocity $\mathbf{v}_p$ and position $\mathbf{x}_p$ using a second-order midpoint scheme.

## 2.2. Halo Finding

For simulations of collisionless dark matter in particular, an important tool for problems concerning cosmic structure and its formation is the identification of *haloes*, i.e. gravitationally bound objects made of particles as well as their internal structure and bound objects nested within them, called *subhaloes*. Codes that perform this task are called "*halo finders*" or "*clump finders*".

Over the last decades, a multitude of halo finding tools has been introduced. The Halo-Finder Comparison Project (Knebe et al. 2011) lists 29 different codes in the year 2010 and roughly divides them into two distinct groups of codes:

1. Particle collector codes, where particles are linked together, usually by linking particles that are closer to each other then some specified linking length, a method referred to as "*friends-of-friends*" (Davis et al. 1985). This implicitly determines some minimal density for the haloes found this way.

2. Density peak locator codes, that first find density peaks and then collect particles around those. One frequently used method to identify haloes in such manner is the "*Spherical Overdensity*" method (Press and Schechter 1974). The basic idea is to find groups of particles by growing spherical shells around density peaks until the mean overdensity of the sphere falls below some threshold.

`PHEW` (Bleuler et al. 2015), the clump finder implemented in `RAMSES`, is also based on first identifying density peaks in the density field $\rho$, but then assigns cells (not particles) to density peaks following the steepest density gradient. This assignment gives rise to patches of cells around density peaks, which will separate the mass density field along minima. Such a method is frequently referred to as '*watershed segmentation*'. Unlike the spherical overdensity method, this allows to identify haloes without the assumption of spherical symmetry. The algorithm can be divided in four main steps:

1. Segmentation
   Cells containing a sufficiently high density are either marked as density peaks, if there are no denser neighbouring cells around them, or assigned to the same peak their most dense neighbour is also assigned to, thus dividing the density field into patches around density peaks.
2. Connectivity Establishment
   Every peak patch needs knowledge of all its neighbouring peak patches, i.e. patches that share the surface along density minima with it. The maximal surface density between two particular peak patches is considered as the "*saddle*" between these two. Naturally, a peak patch can have multiple neighbouring peak patches. Out of all the saddles of all the neighbouring peak patches, the one with the highest density is called the "*key saddle*" and the neighbour it connects to is referred to as the "*key neighbour*".
3. Noise Removal
   Each peak patch is assigned a value representing the contrast to the background called "*relevance*", defined as the ratio of the peak's density to its key saddle. A peak patch is considered noise if its relevance is lower than a user-defined relevance threshold. An irrelevant peak patch is then merged to its key neighbour. Expressed explicitly, merging a peak patch $i$ into a patch $j$ means that all cells of $i$ inherit the peak label of $j$.
4. Substructure Merging
   Once the noise removal step is completed, the remaining structure consists only of peak patches, essentially clumps of particles, which satisfy the relevance condition. These clumps represent the structure on the lowest scale. A large halo for example, which can very roughly be described as "a large clump" in a first approximation, would be decomposed into many small clumps. In order to identify such a halo as a single object, one more step is necessary: The identified clumps at the lowest scale need to be merged further into composite clumps. All peak patches whose key saddle density is higher than some user-defined saddle threshold are merged into their key neighbour. The saddle threshold defines which clumps should be considered as separate structures and which should be merged and considered as composite structures. This merging is done iteratively and in doing so the hierarchy of substructure is established.

## 2.3. Unbinding Particles

A further requirement for substructure finding is the removal of energetically unbound particles, i.e. assigning a particle originally located within a substructure to the parent structure based on an energy criterion. This applies recursively to any level of substructure within substructure. Such a hierarchical clustering of matter is expected due to self gravity. It is customary to treat

all particles assigned to a halo as bound to it, even though from a strict energetic perspective they are not, thus particle unbinding may not be necessary for haloes, but it is vital for subhaloes. Subhaloes are by definition located within a host halo and are therefore expected to be contaminated by the host's particles. Considering that substructure often contains far fewer particles than its host, blindly assigning particles to it without an unbinding procedure can influence its physical properties significantly.

Various techniques to identify unbound particles within the found structures are used as well. Because removing particles from a structure changes said structures properties, often iterative approaches are used. `AHF` (Knollmann and Knebe 2009), `ASOHF` (Planelles, S. and Quilis, V. 2010) and `SUBFIND` (Springel et al. 2001) for example remove all unbound particles from a halo, recompute the potential and repeat the procedure until no more particles are removed. Unbound particles are then passed from subhaloes on to host haloes (or host subhaloes) for examination. `SKID` (Stadel 2001) however only removes the particle with the highest energy per iteration.

By considering an isolated clump in a time-independent scenario, where energy is conserved, a particle $i$ is considered to be bound if its total energy $E_i$

$$E_i = \frac{1}{2} m_i \cdot v_i^2 + m_i \phi(\mathbf{r}_i) \tag{3}$$

is negative:

$$E_i < 0 \qquad \Leftrightarrow \text{ particle is bound} \tag{4}$$

$$\Rightarrow \quad v_i < \sqrt{-2 \cdot \phi(\mathbf{r}_i)} \qquad \Leftrightarrow \text{ particle is bound} \tag{5}$$

where $\mathbf{r}_i$ is the particle's position, $v_i = ||\mathbf{v}_i||$ is the magnitude of the particle's velocity, both given in the centre of mass frame of the clump, and $m_i$ is the mass of every particle $i$. Approximating the clump as a spherically symmetric object, the Poisson equation (eq. (2)) reduces from a three-dimensional problem to a one-dimensional one, as the potential $\phi(\mathbf{r})$ will only depend on the absolute distance from the origin, $r_i \equiv ||\mathbf{r_i}||$, not any angular direction. For this case, the Poisson equation for the gravitational potential $\phi$ can be solved analytically under the assumption that $\phi(r \to \infty) \to 0$:

$$\phi(r_i) = -G \int_{r_i}^{r_{max}} \frac{M(< \tilde{r})}{\tilde{r}^2} \mathrm{d}\tilde{r} - G \frac{M_{tot}}{r_{max}} \tag{6}$$

$$M(< r) \equiv \int_0^r 4\pi \rho(\tilde{r}) \tilde{r}^2 \mathrm{d}\tilde{r} \tag{7}$$

Where $M(< r)$ is the mass enclosed by a sphere of radius $r$ such that the clump's total mass is enclosed by the radius $r_{max}$: $M_{tot} = M(< r_{max})$.

One problem with this condition is that it assumes that clumps will be isolated, which by construction of the clump finder `PHEW` subhaloes will never be. The issues that arise from this fact can be understood by considering the boundaries of a particle's trajectory. These boundaries in a given potential $\phi$ can be estimated using the conservation of energy:

$$E/m_p = \frac{1}{2} v^2 + \phi = const. \tag{8}$$

A gravitational potential is qualitatively shown in figure 1 as well as a particle $\alpha$ with $\frac{1}{2}v^2 < -\phi$. The total energy per particle mass $E/m_p$ of the particle on the graph is then exactly the difference

between the negative potential and the kinetic energy on the $y$-axis. In order to conserve energy, the curve of possible kinetic energies (dotted line in figure 1) that the particle may take on will always have the same distance $E/m_p$ from the negative potential curve. Because $v^2 \geq 0$, the spatial boundaries of a particle's trajectory can be found by following the curve of possible kinetic energies of the particle to the points where $v^2 = 0$.

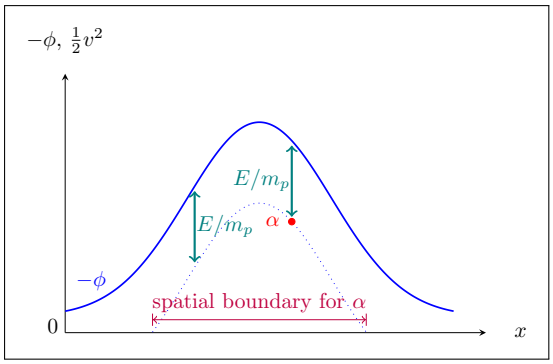


**Figure 1:** A qualitative plot of a potential $-\phi$ and a particle $\alpha$. The boundaries of the particle's trajectory can be found using energy conservation $E/m_p = \frac{1}{2}v^2 + \phi = const$ by following the curve of the particle's kinetic energies (dotted line) to the points where $v^2 = 0$.

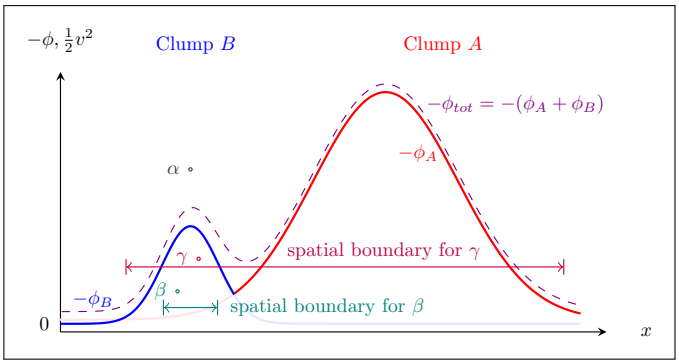


**Figure 2:** Qualitative potential of a halo containing two clumps $A$ and $B$. Three particles assigned to $B$ are shown: $\alpha$ is not bound to $B$, $\beta$ is bound, $\gamma$ satisfies the energy condition to be bound, but can wander off into clump $A$ and shouldn't be considered as such.

This fact changes the situation significantly for the interpretation of what particles should be considered bound as follows. Consider now an isolated halo that consists of two clumps, $A$ and $B$, where $B$ is a smaller clump nested within clump $A$. Their potentials are qualitatively depicted in figure 2. Three particles assigned to $B$ with different kinetic energies are marked, representing three different cases:

- Particle $\alpha$ has a kinetic energy higher than the potential, it is clearly not bound to the clump $B$.
- Particle $\beta$ has a kinetic energy lower than the potential at that distance from the centre of mass, so it will remain bound on an elliptic trajectory around the centre of mass.
- Particle $\gamma$ is considered energetically bound to the clump just like $\beta$, i.e. it satisfies condition (5), but it won't necessarily remain on an elliptic trajectory around clump $B$'s centre of mass: Because of clump $A$'s neighbouring potential, the particle can leave the boundaries of clump $B$ and wander off deep into clump $A$.

These considerations show that due to the fact that subhaloes will always have neighbouring structure by definition, there will be particles like $\gamma$ that can wander off into neighbouring clumps even though they satisfy condition (5). It is obvious that particles like $\gamma$ shouldn't be considered as bound and that therefore the condition for a particle to be bound needs to be modified appropriately. Since the reason $\gamma$ can wander off is because its boundary extends past the interface that connects the two clumps, the condition for a particle to be bound must be that its trajectory must never reach that interface. Defining $\phi_S$ to be the potential of clump $B$ at the interface to the neighbouring structure that is closest to $B$'s centre of mass, the condition for a particle to be bound *exclusively* to a particular clump can be written as

$$E/m_p = \frac{1}{2}v^2 + \phi - \phi_S < 0 \tag{9}$$

or equivalently:

$$v < \sqrt{-2(\phi - \phi_S)} \tag{10}$$

According to the argumentation above and figures 1 and 2, it is to be expected that demanding particles to be exclusively bound will find more unbound particles than not doing so, where particles close to the centre of mass should be more likely to be exclusively bound than the particles closer to the edge of the subhalo.

## 2.4. Parallel implementation

`RAMSES` is a parallel code which makes use of the MPI standard. The use of MPI (message passing interface) allows a process on a distributed memory architecture to be executed in parallel by multiple tasks and defines various types of communications between them.

The fundamental parallelisation strategy used in `RAMSES` is domain decomposition, where a part of the total spatial computational domain is assigned to each processing unit, or "*MPI task*". It makes use of the fact that most calculations on grids do not require the knowledge of the entire computational domain, but only the cells in their vicinity. The basic idea of the domain decomposition is illustrated in figure 3, where a 2D-grid is split between two processors. The partial domains do not overlap and a thin layer of cells, called the "*virtual boundary*", is introduced where the domain was cut. Only necessary information is then communicated across MPI tasks from the other tasks' "real domain" into the virtual boundary, so that the virtual boundary copies what happens in the domains of other tasks, allowing the execution of the code as if the domain wasn't split. Two tasks working on the same problem allows a much faster execution time, but it also enables the solution of bigger problems that only one task couldn't do on its own, e.g. because of memory restrictions.

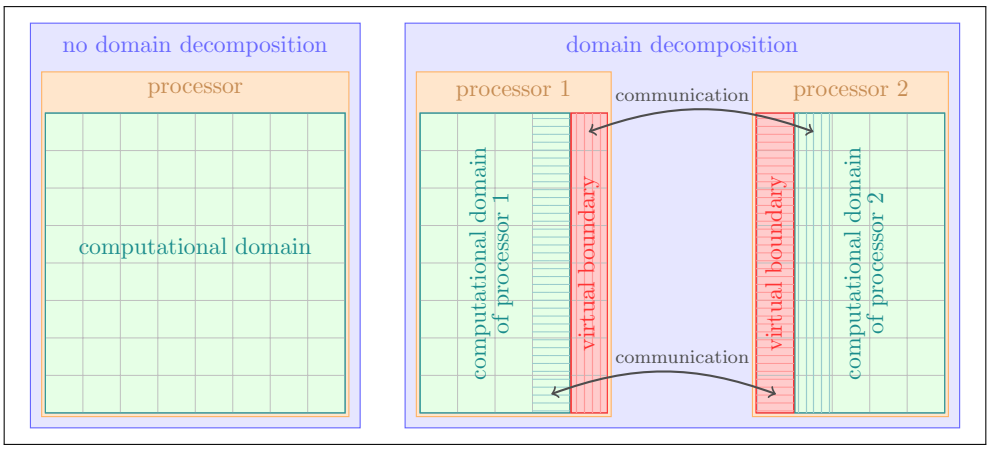


**Figure 3:** The basic idea of domain decomposition. Here a 2D-grid is split between two processors (right) instead of only one (left). The partial domains do not overlap and where the domain was cut, a "virtual boundary" is introduced. Necessary information is then communicated between processors from the other tasks' "real domain" into the virtual boundary, so that the virtual boundary copies what happens in the domains of other tasks and allows the execution of the code as if the domain wasn't split.

`PHEW` uses the virtual mesh boundary as well, since every cell on each domain must have the information of all its neighbours. Similarly to the virtual mesh boundary, a virtual peak boundary is necessary. For the peak patch merging step, each peak patch on each task's domain needs the information of the peak patches that surround it. If, for example, a peak patch is split in two by the domain boundaries between two tasks, both tasks need to know all the peak patch's neighbours on the other task's domain. Unlike the mesh boundary however, the virtual peak boundary is not a fixed region in space. Because of the merging, peaks gain neighbours they hadn't had before. This requires that new peaks are introduced to the virtual peak boundary

during the merging procedure. Once introduced, all other peak properties, e.g. its relevance and saddle points, can be transferred by means of MPI communication.

The virtual peak boundary requires two types of communications. One type is the collection (sum, minimum or maximum) of a value for a peak from all tasks which have that particular peak patch in their virtual boundary to the owner of the peak. The "*owner*" of the peak is the task where the density peak is in the "real" domain, as opposed to the virtual mesh boundary. Imagine for example the calculation of a peak patch's total volume on multiple tasks: Each task would compute the volume of that peak patch on its own domain and then all these partial results would be sent to the peak's owner and summed up.
In that scenario only the peak's owner has the total volume of the patch, but the ones that have it in their virtual boundary still only have their partial values. This brings us to the second required type of communication: A scatter of data from the owner of the peak to all tasks with that particular peak in their virtual boundary. Following the example where the total volume of a peak patch is computed, the owner of the peak would send the computed total volume of the peak patch to all the tasks which require that information. In order to perform these communications, a communication structure (called the "peak communicator") must be built first. The purpose of the peak communicator is to establish how many peaks of every task are owned by any other task, or in other words: "what needs to be sent (and received from) where". Once that is known, the communications between the processes and thus the parallel merging can be performed.

## 2.5. Dark Matter Halo Merger Trees

### 2.5.1. Terminology

Before having a closer look at dark matter halo merger trees and how to make them, some clear definitions and terminology on the subject are necessary. In this work, the terminology as set by the Merger Tree Comparison Project (Srisawat et al. 2013) is adapted:

- A *halo* is a dark-matter condensation as returned by a halo-finder.
- Haloes may be spatially nested: in that case the outer halo is the *main halo* and the other haloes are *subhaloes*. Where no distinction between subhaloes and main haloes is made, they will be collectively referred to as *clumps*.
- Haloes are defined at distinct *s*napshots. Snapshots correspond to particular values of cosmic time and contain the particle IDs, mass, location & velocity for each dark matter particle in the simulation.
- For two snapshots at different times to the older one (i.e. higher redshift) is referred to as $A$ and the younger one (i.e. lower redshift) as $B$.
- A *graph* is a set of ordered halo pairs, $(H_A, H_B)$, where $H_A$ is older than $H_B$. It is the purpose of the merger-tree codes to produce a graph that best represents the growth of structure over cosmic time. $H_A$ and $H_B$ are usually taken from adjacent snapshots, but this is not a requirement as there are occasions where haloes lose their identity and then reappear at a later time.
- Recursively, $H_A$ itself and progenitors of $H_A$ are *progenitors* of $H_B$. Where it is necessary to distinguish $H_A$ from earlier progenitors, the term *direct progenitor* will be used.
- Recursively, $H_B$ itself and descendants of $H_B$ are *descendants* of $H_A$. Where it is necessary to distinguish $H_B$ from later descendants, the term *direct descendant* will be used.
- This work is primarily concerned with *merger trees* for which there is *precisely one direct descendant for every halo*. Note that it is possible for haloes near the minimum mass limit to have zero descendants: such haloes are omitted from the analysis.

- In the case that there are multiple direct progenitors, it is required that precisely one of these is labelled the *main progenitor*.
- The *main branch* of a halo is a complete list of main progenitors tracing back along its cosmic history.

### 2.5.2. Aim

Merger trees follow the growth and merges of dark matter haloes over cosmic history. In the hierarchical structure formation scenario, starting with a nearly uniform density field of the Universe, large, massive haloes are thought to form mainly through a series of merging events of smaller haloes over cosmic time. Figure 5 shows schematically how a massive halo may form through consecutive merging events. Because galaxies form inside the potential well dark matter haloes, knowledge of how many merging events a halo underwent during its lifetime is crucial for accurate mock galaxy catalogues. After a merging event, the galaxy of the smaller halo that has been "swallowed up" by a bigger one has no reason to simply vanish without a trace. The "swallowed up" halo might become a subhalo, or, if it is small enough or after some time, it might not be detectable as substructure in the simulation any more. Galaxies of haloes that dissolve in this manner are referred to as "*orphan galaxies*". Their importance can be clearly seen in figure 4, which shows the most massive haloes resulting from two cosmological simulations described in section 4.2. If orphan galaxies are taken into account, the number of galaxies in these haloes grows from less than 100 galaxies based on subhaloes with a minimal number of bound particles to a few thousand.

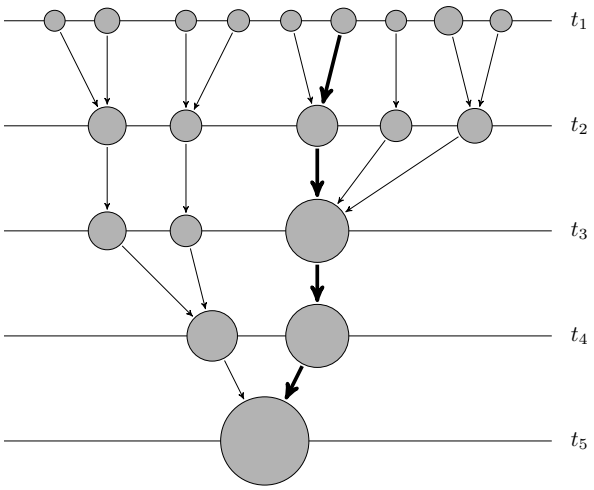


**Figure 5:** Illustration of a merger tree, showing the formation history of some halo over cosmic time through a series of merging events with $t_1 < t_2 < t_3 < t_4 < t_5$. The $x$-axis has no physical meaning. The main branch of this merger tree is signified by the thicker arrows. The size of the circles represents the haloes' masses.

In dark matter simulations haloes however are only defined at distinct snapshots. The aim of a merger tree code is to link haloes from an earlier snapshot to the haloes of the consecutive snapshot, i.e. to find the descendants of the haloes of the earlier snapshot within the consecutive snapshot, thus enabling the tracking of growth and merges of haloes in a simulation.

A straightforward method to link progenitors with descendants in adjacent snapshots is to trace particles by their unique ID. All but one merger tree algorithm tested in Srisawat et al. 2013 also rely on particle IDs to connect progenitor clumps with descendant ones. Essentially this would mean to check in which clumps of a later snapshot $B$ did particles that were found to

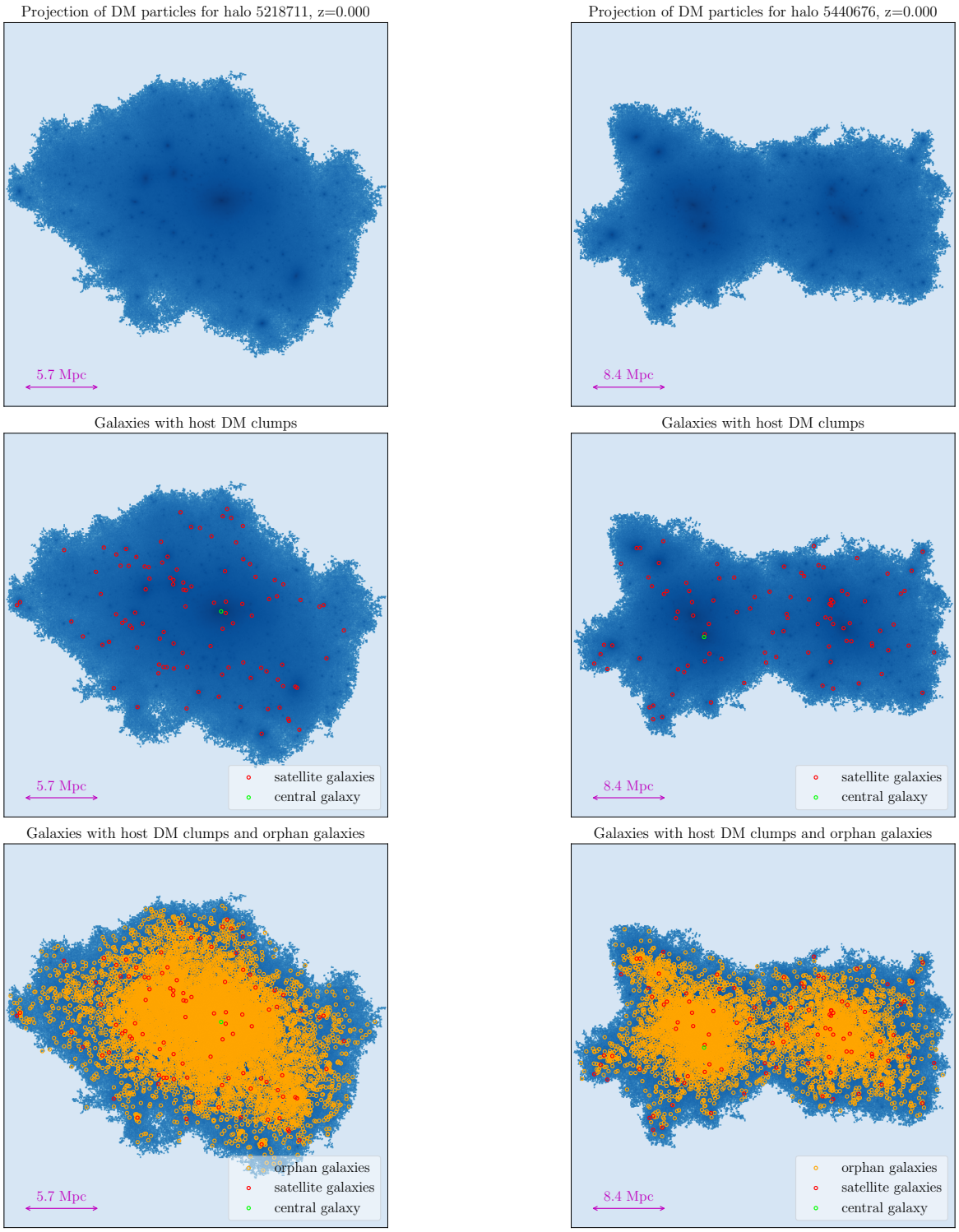


**Figure 4:** Projection along the $z$-axis of the most massive dark matter haloes from the G69 (left) and the G100 (right) simulations with marked galaxy positions at $z = 0$. The simulations used are described in section 4.2. The circles representing galaxies do not indicate their physical size. The arrows indicating the physical lenght of the image correspond to one fifth of the plotted region in each direction.
Noticeably, some dark matter condensations appear to have no satellite galaxy assigned to them. There are two reasons for this: Firstly, because the visualisation technique used is a projection, no information on how stretched along the $z$-axis these clumps are is shown. Secondly, the stricter unbinding criterion was applied for these simulations, where a particle is only considered bound if it can't escape the spatial boundaries of the clump (see section 2.3 for details). This tends to remove more particles from a clump then when not applied, and possibly remove enough particles from a clump so that it doesn't satisfy a lower mass threshold condition. If that is the case, such clumps are removed from the (sub)halo catalogue.

be in a clump in an earlier snaphot $A$ end up in. Other methods are also conceivable: `JMERGE` (Srisawat et al. 2013) for example uses the properties of haloes at both snapshot $A$ and $B$ to estimate where each halo should be at the midpoint in time between the two snapshots, and then links the haloes based on these calculated positions.

Finally, the presented merger tree algorithm is required to run on the fly and in parallel using the MPI library.

### 2.5.3. Restrictions, Complications and Solutions

To obtain a merger tree, as opposed to a merger graph, each progenitor may have exactly one direct descendant halo. Descendants however may have multiple direct progenitors. In this case, exactly one of the direct progenitors must be labelled as the main progenitor. The other direct progenitors of this descendant are then assumed to have merged into the main direct progenitor to form the descendant.

Descendant candidates for any progenitor are identified by tracing particles of that progenitor across snapshots. Naturally, those particles may end up in multiple clumps, giving multiple descendant candidates for a progenitor. In such cases, the most promising descendant candidate will be called the *main descendant*. To find a main progenitor and a main descendant, usually some merit function $\mathcal{M}$ is defined, which is to be maximised or minimised, depending on its definition.

Let $\mathcal{M}_{pd,adj}(A, B_i)$ be the merit function to be maximised for a number of descendants $B_i$ to be a main descendant of a progenitor $A$, and let $n_{mb}$ be the total number of particles of progenitor $A$ that are being traced to a later adjacent snapshot, where the descendants $B_i$ are found. $n_{mb}$ may or may not be the total number of particles of $A$. Then a straightforward ansatz for $\mathcal{M}_{pd,adj}(A, B_i)$ would be:

$$\mathcal{M}_{pd,adj}(A, B) \propto \frac{n_{A\cap B_i}}{n_{mb}} \tag{11}$$

where $n_{A\cap B_i}$ is the number of traced particles of $A$ found in $B$. Similarly, if $\mathcal{M}_{dp,adj}(A_i, B)$ is the merit function to be maximised for a number of progenitors $A_i$ to be the main progenitor of a descendant $B$ in an adjacent snapshot, then a straightforward ansatz would be:

$$\mathcal{M}_{dp,adj}(A_i, B) \propto \frac{n_{A_i\cap B}}{N_B} \tag{12}$$

where $N_B$ is the total number of particles in clump $B$. In these two merit functions, $n_{mb}$ and $N_B$ constitute a norm.

These two merit functions can be united into one by considering that when evaluating the value of $\mathcal{M}_{adj}$, in both cases the denominator is independent of the candidates for which it is evaluated: The number of particles traced, $n_{mb}$, won't depend on what or how many descendant candidates have been identified. The same goes for total the number of particles of clump $B$, which won't change by choosing some progenitor candidate or the other as the main progenitor. So the merit function for adjacent snapshots can be reduced to

$$\mathcal{M}_{adj}(A, B) \propto n_{A\cap B} \tag{13}$$

A complication arises from the fact that the clump-finder in `RAMSES` defines the main halo as the one with the highest density peak. If for example a halo consists of two similar clumps with

similar height of their respective density peaks, then it is possible that over time small variations in the density peak will lead to oscillations in the identification of the main halo between these two clumps. The particle unbinding algorithm will then look for unbound particles in what was found to be the subhalo and pass them on to the main halo, increasing its mass and decreasing the subhalo's mass. This is amplified when a particle is defined to be bound if and only if it mustn't cross the spatial boundaries of the subhalo. Therefore, if between snapshots the identification of which clump is the main halo varies, then strong mass oscillations can be expected. To counter this behaviour, the merit function can be extended to prefer candidates with similar masses:

$$\mathcal{M}_{adj}(A,B) = \frac{n_{A\cap B}}{\frac{m_>}{m_<} - 1} \tag{14}$$

The factor $(m_>/m_< - 1)^{-1}$ increases as $m_> \to m_<$, where $m_<$ and $m_>$ are the smaller and larger mass of the descendant-progenitor pair $(A, B)$, respectively. An overview of what merit functions were chosen in other merger tree algorithms is given in table 1 in Srisawat et al. 2013.

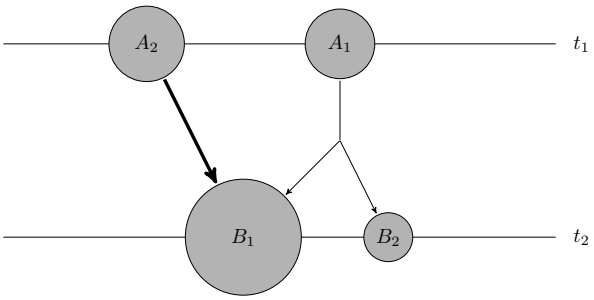


**Figure 6:** Illustration of a progenitor $A_1$ at time $t_1$ which is partially merged into a descendant $B_1$ at time $t_2 > t_1$, but some other part $B_2$ isn't. Because $A_1$ is not the main progenitor of $B_1$, by assigning its descendant only according to the merit function (14) would not pass on its formation history to $B_2$, but treat it as newly formed. The size of the circles represents the haloes' masses, the $x$-axis has no physical meaning.

Since a descendant may have multiple progenitors, but each progenitor may have only one descendant, a question that needs to be adressed is how to deal with situations where multiple descendant candidates, i.e. descendant clumps that contain tracked particles of some progenitor, are found. Problems arise for example when some progenitor $A_1$ is not the main progenitor of its main descendant $B_1$, but also has fractured into another descendant candidate $B_2$. This situation is schematically shown in figure 6. Relying only on the merit function (14), progenitor $A_1$ will seem to have merged with $A_2$, the direct progenitor of $B_1$, in order to form $B_1$. The fractured remainder, $B_2$, will be treated as newly formed, provided it has no other progenitor candidates. In this case the entire formation history of $B_2$ would be lost. In order to preserve history, instead of merging progenitor $A_1$ into $B_1$, the link to $B_2$ should be preferred. This is simpler to implement into the algorithm than to express via the merit function. If $A_1$ is not the main progenitor of its main descendant $B_2$, then don't merge it into $B_2$ until all of $A_1$'s descendant candidates have found their own main progenitor, and give priority to $A_1$ for being a progenitor to some descendant which is not its main descendant over merging it into its main descendant.

Finally, in some cases a subhalo passing close to the core of its main halo may not be identified as an individual subhalo and appear to be "swallowed up" by the main halo, but will re-emerge at a later snapshot. Such a scenario is shown in figure 7. When this occurs, the merger tree code will deem the subhalo to have merged into the main halo, and will likely find no progenitor for the re-emerged subhalo, thus treating it as newly formed. This is a problem because this will essentially erase the growth history of the subhalo, regardless of its size. This way, massive

clumps may be found to just appear out of nowhere in the simulation. For this reason, it is necessary to check for links between progenitors and descendants in non-consecutive snapshots as well. As the presented merger tree code works on the fly, future snapshots will not be available at the time of the tree making, so it will be necessary to check for progenitors of a descendant in multiple past snapshots. This can be achieved by keeping track of the most strongly bound particle of each clump when it is merged into some other clump. (These particles are also used to follow orphan galaxies.)

Priority is given to progenitor candidates in adjacent snapshots. Only if no progenitor candidates have been found for some descendant, then progenitor candidates from non-adjacent snapshots will be searched for. Because these progenitors from non-adjacent snapshots are only tracked by one single particle, the previous merit function can't be applied. Instead, a straightforward choice would be to find the orphan galaxy particle within the descendant clump which is most tightly bound, i.e. minimises $E$ in condition (10).

The `Consistent Trees` algorithm (Behroozi et al. 2013) for example addresses this problem differently. The positions and velocities of descendants are evolved back in time and compared to progenitor candidate's properties. If they differ too much, the links between them are cut, and for descendants without likely progenitors, new haloes at previous snapshots are created. Such introduced haloes are being kept track of for several snapshots, allowing to link identified clumps across non-adjacent snapshots. Others, like `LHaloTree` (Springel et al. 2005) and `D-Trees` (Jiang et al. 2014), allow the search for a descendant in later snapshots in the same manner as if the snapshots were adjacent. Intuitively this might seem like a more reliable method to create merger trees, however it is considerably more computationally expensive and therefore not suited for on the fly merger tree creation.

## 2.6. Creating Mock Galaxy Catalogues from Dark Matter Simulations

Once merger trees from DMO simulations are available, the only missing link to obtain mock galaxy catalogues is a galaxy-halo connection. Various approaches have been used to establish such a connection. Wechsler and Tinker 2018 distinguish between "*two basic approaches to modeling the galaxy-halo connection,* empirical modeling*, which uses data to constrain a specific set of parameters describing the connection at a given epoch or as a function of time, and* physical modeling*, which either directly simulates or parametrizes the physics of a galaxy formation such as gas cooling, star formation, and feedback.*" The models are not mutually exclusive: Starting from a hydrodynamical simulation as an example of a very physical model, where dark matter, gas, and star formation processes are directly simulated (e.g. Chaves-Montero et al. 2016), some assumptions may be relaxed and constrained by data instead. Semi-analytic models (e.g. White and Frenk 1991, Bower et al. 2006, Somerville and Primack 1999, Kauffmann, White, and Guiderdoni 1993) for example approximate some processes with analytical prescriptions, however parameters of these prescriptions need to be constrained empirically with observational data. Physical models are in general computationally more expensive, making them less suitable to be used on the fly.

Broadly speaking, empirical models of the galaxy-halo connection make no effort to explain the physical processes governing galaxy formation, but are mainly concerned with constraining a prescription of galaxy properties given a halo catalogue. The Halo Occupation Density (HOD) model (e.g. Seljak 2000, Berlind and Weinberg 2002) for example specifies the probability distribution of the number of galaxies that meet some criteria like a luminosity or stellar mass threshold in a halo, typically depending on its mass. Conditional Luminosity Functions (e.g. Vale and Ostriker 2006) and Conditional Stellar Mass Functions (e.g. Yang, Mo, and van den

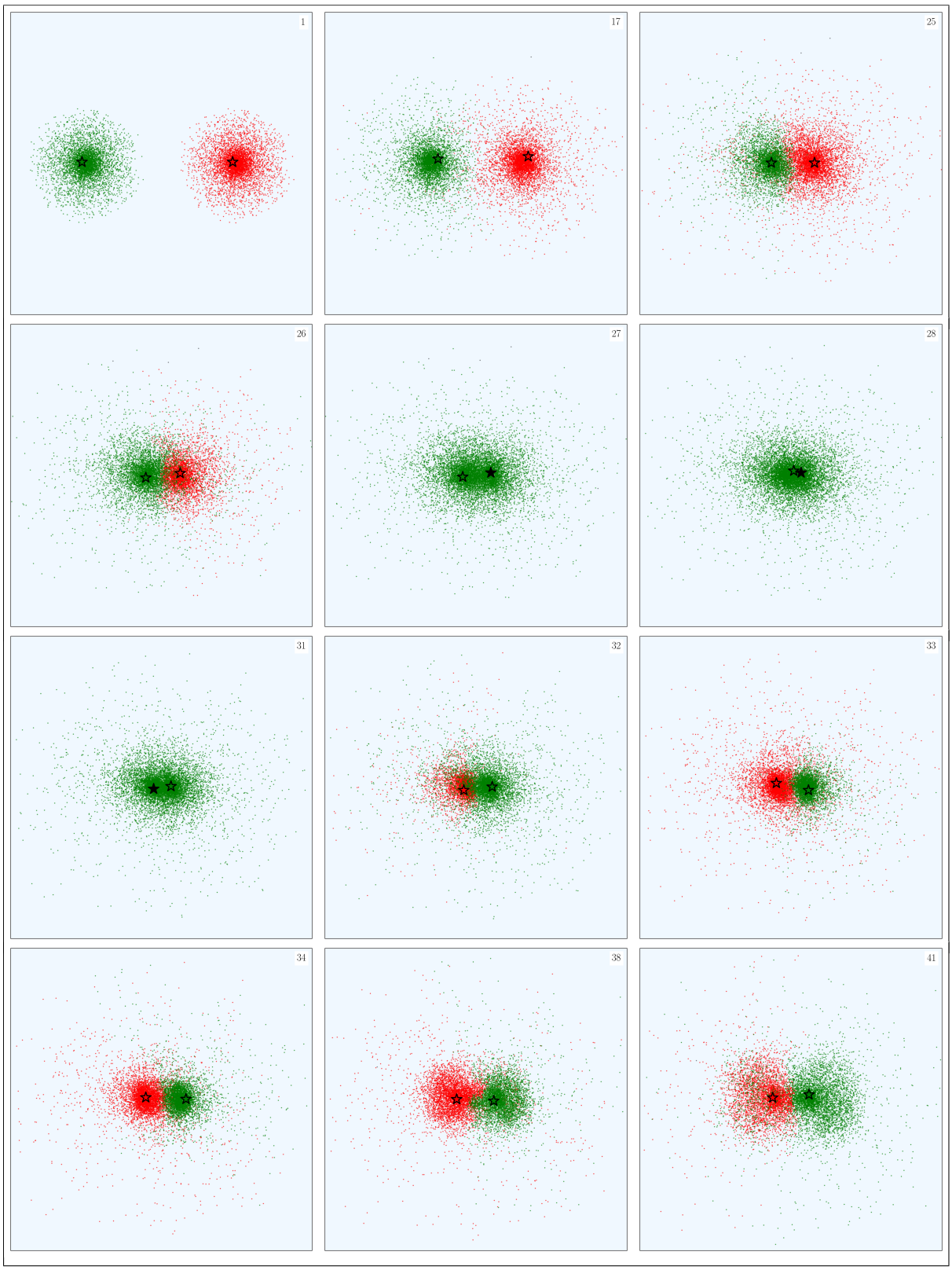


**Figure 7:** Illustration of how haloes can seemingly merge into another one and re-appear a few snapshots later. The green and red particles are two initially distinct haloes that pass through each other. The galaxies assigned to them are marked by a star with the same colour as the particles. Black stars mark orphan galaxies, which have lost their unique host halo. The number in the upper right corner of each plot is the snapshot number that is depicted. In snapshots 27-31, the halo-finding algorithm didn't identify both haloes as distinct objects. However by tracking the red halo's orphan galaxy, it was possible to link the halo in snapshot 32 all the way back to snapshot 26.
The simulation was created using `DICE` (Perret 2016). Both haloes are identical with mass of $5 \cdot 10^{10} M_{\odot}$, each containing 5000 particles and following a NFW mass profile. The plotted region corresponds to $400$ kpc on each side.

Bosch 2009) go one step further and describe the full distribution of galaxy luminosities or masses for a given halo mass.

Other empirical models make the assumption that the most massive galaxies live in the most massive haloes and then rank-order galaxies from observations by mass (or some other property) with dark matter (sub)haloes from simulations. These techniques are commonly called 'Halo Abundance Matching' (HAM), or 'Subhalo Abundance Matching' (SHAM) in case one assumes that subhaloes host a galaxy on their own (e.g. Kravtsov et al. 2004, Vale and Ostriker 2004). A further commonly used assumption is based on the fact that subhaloes, once accreted by their respective host halo, quickly loose their mass as their outer regions are stripped away due to tidal forces. Nagai and Kravtsov 2005 have shown that the galaxies hosted by subhaloes however, because they're located close to the centre of the subhalo, are stripped of their mass only much later. This leads to the approximation that the stellar mass of subhaloes' galaxies isn't directly determined by the current mass of the subhalo, but to either the mass of the subhalo at the time it was accreted by the main halo or the subhalo's peak mass during its formation. This is yet another reason why merger trees are essential for mock galaxy catalogues.

Using either abundance matching or by constraining a parametrisation with observational data, the typical galaxy stellar-mass-to-halo-mass (SMHM) relation can be determined, which essentially gives the expected stellar mass for any given halo mass at different epochs such that the resulting galaxy catalogues coincide with observations. Usually a one-to-one monotonic relation between stellar and halo mass assumed. In this work, such a SMHM relation as found by Behroozi, Wechsler, and Conroy 2013 is used to determine galaxy stellar masses from merger trees. This SMHM relation was chosen for two reasons: Firstly, Behroozi, Wechsler, and Conroy 2013 find that the commonly used double power law for the SMHM relation, like the one used in Moster, Naab, and White 2013, cannot accurately fit the unique shape of the SMF. Secondly, Behroozi, Wechsler, and Conroy 2013 fits their data up to $z \sim 8$, while others like Moster, Naab, and White 2013 and Yang et al. 2012 "only" go up to $z \sim 4$.

The parametrisation is as follows:

$$\log_{10}(M_*(M_h)) = \log_{10}(\epsilon M_1) + f\left(\log_{10}\left(\frac{M_h}{M_1}\right)\right) - f(0) \tag{15}$$

$$f(x) = -\log_{10}(10^{\alpha x}+1) + \delta\frac{[\log_{10}(1+\exp(x))]^\gamma}{1+\exp(10^{-x})} \tag{16}$$

Here $M_*$ is the stellar mass and $M_h$ is the halo mass. Because the stellar mass is thought to depend not explicitly on the mass, but on the depth of the potential well where the baryonic matter is located, the mass used to obtain stellar masses within the code is always inclusive, meaning that any parent clump will be considered to contain its substructure's mass, independently of which mass definition of substructure is used to link clumps together between snapshots for the generation of merger trees. For central haloes, $M_h$ is its current mass, while for satellites, $M_h$ is the peak progenitor mass in its entire formation history. The galaxy is placed at the position of the most tightly bound particle of each dark matter clump.

The other parameters from equations (15) and (16) and their best fits as found by Behroozi, Wechsler, and Conroy 2013 are:

$$\nu(a) = \exp(-4a^2) \tag{17}$$

$$\begin{aligned}\log_{10}(M_1) &= M_{1,0} + (M_{1,a}(a-1) + M_{1,z}z)\nu \\ &= 11.514 + (-1.793(a-1) + (-0.251)z)\nu\end{aligned} \tag{18}$$

$$\log_{10}(\epsilon) = \epsilon_0 + (\epsilon_a(a-1) + \epsilon_z z)\nu + \epsilon_{a,2}(a-1) \tag{19}$$

$$= -1.777 + (0.006(a-1) + 0.000z)\nu - 0.119(a-1)$$

$$\alpha = \alpha_0 + (\alpha_a(a-1))\nu \tag{20}$$
$$= -1.412 + (0.731(a-1))\nu$$

$$\delta = \delta_0 + (\delta_a(a-1) + \delta_z z)\nu \tag{21}$$
$$= 3.508 + (2.608(a-1) + ((-0.043)z)\nu$$

$$\gamma = \gamma_0 + (\gamma_a(a-1) + \gamma_z z)\nu \tag{22}$$
$$= 0.316 + (1.319(a-1) + 0.279z)\nu$$

with $z$ bein the redshift and $a$ being the cosmological scale factor. Additionally, one would not expect two haloes of same mass $M_h$ to also each host a galaxy of exactly the same mass. Haloes may have different formation histories, spins, and concentrations even when having exactly the same mass. For this reason, a lognormal scatter in the halo mass is introduced, which scales with redshift via a two-parameter scaling:

$$\xi = \xi_0 + \xi_a(a-1) = 0.218 + (-0.023)(a-1) \tag{23}$$

Lastly, Behroozi, Wechsler, and Conroy 2013 also introduce parameters to account for observational systematics, which haven't been used in scope of this work.

# 3. Making and Testing Merger Trees

## 3.1. Making Merger Trees

The first step for any merger tree code is to identify plausible progenitor candidates for descendant clumps as well as descendant candidates for progenitor clumps. In this algorithm, this is done by tracking up to a maximal number, labelled $n_{mb}$, of particles per progenitor clump. The minimal number of tracker particles is given by the mass threshold for haloes. The tracker particles of any halo are chosen to be the $n_{mb}$ particles with the lowest energy $E$:

$$E = m\mathbf{v}^2 + \phi(\mathbf{r})$$

where $m$ is the particle's mass, $\mathbf{v}$ is the particle's velocity relative to the halo's bulk velocity, $\phi$ is the gravitational potential of the halo and $\mathbf{r}$ is the position of the particle. If $E < 0$, a particle is considered to be energetically bound to the halo. Selecting the $n_{mb}$ particles with lowest energy $E$ thus corresponds to choosing the "$n_{mb}$ **m**ost (tightly) **b**ound particles of the halo". This choice is made because the most strongly bound particles are expected to more likely remain within the clump between two snapshots.

For every clump in the current snapshot, the $n_{mb}$ tracker particles are found and written to file. In the following output step, those files will be read in and sorted out: The clumps of the previous snapshot will be the progenitors of this snapshot. Based on in which descendant clump each progenitor's particles ended up in, progenitors and descendants are linked, i.e. possible candidates are identified this way.

Next, the main progenitor of each descendant and the main descendant of each progenitor need to be found. This search is performed iteratively. A main progenitor-descendant pair is established when the main progenitor of a descendant is the main descendant of said progenitor. At every iteration, all descendant candidates of all progenitors that haven't found their match

yet are checked; The descendants without a matching progenitor however only move on to the next best progenitor candidate. For both descendants and progenitors, all candidates are ranked from "best" to "worst" based on the merit function (14). The iteration is repeated until every descendant has checked all of its progenitor candidates or found its match. Progenitors that haven't found a main descendant that isn't taken yet will be considered to have merged into their best fitting descendant candidate.

After the iteration, any progenitor that is considered as merged into its descendant will be recorded as a "past merged progenitor". Only one, the most strongly bound, particle and the time of merging will be stored for past merged progenitors. This particle is referred to as the "galaxy particle" of the merged progenitor. Storing this data will allow to check in later, non-consecutive snapshots whether the progenitor has truly merged into its main descendant and to track orphan galaxies.

Then descendants that still haven't got a progenitor at this point will try to find one in non-consecutive past snapshots: The particles that the descendant consists of are checked for being a galaxy particle of a past merged progenitor. The most strongly bound galaxy particle will be considered the main progenitor of the descendant under consideration.

Descendants that still haven't found a progenitor at this point are deemed to be newly formed. This concludes the tree-making and the results are written to file.

Because every processing unit in the current implementation reads in all progenitor data, which unlike the current clump data doesn't change any more, no intricate and flexible communication structures like a peak communicator for the clump finder are necessary. Simple collective MPI communications suffice.

Lastly, there is an option to remove past merged progenitors from the list once they merged into their main descendants too many snapshots ago. By default, the algorithm will store them until the end of the simulation. For the interested reader, a detailed description of the merger tree algorithm is given in appendix A.

## 3.2. Testing Parameters of the Merger Tree Algorithm

### 3.2.1. Methods

The current implementation of the merger tree algorithm allows for multiple free parameters for the user to choose from, which will be introduced and tested further below in this section. Testing these parameters is not a straightforward matter, mainly because there is no "correct solution" which would enable a comparison and error quantification. Nevertheless, one could define a set of quantities one deems a priori favourable for a merger tree and cross-compare these quantities obtained for varying parameters on an identical cosmological simulation. This method was also used in the "Sussing Merger Trees Comparison Project" (Srisawat et al. 2013, Wang et al. 2016, Avila et al. 2014). Some or similar quantifications from the "Sussing Merger Tree" paper series are adapted in this work, as they seem sensible and even allow for a rough cross-comparison with other merger tree codes.

The following properties will be used to quantify the merger trees:

- **Length of the Main Branch**

The length of the main branch of $z = 0$ haloes gives a most basic measure of how far back in time the formation history of these haloes can be tracked. Naturally, longer main branches should be considered a favourable feature for a merger tree code.
In this work, the length of the main branch is defined as the number of snapshots a halo and its progenitors appear in. A newly formed halo at the $z = 0$ snapshot, which doesn't have any progenitors, will by definition have the main branch length of 1. If a halo appears to merge into another, but re-emerges at a later snapshot and is identified to do so, then the snapshots where it is missing from the halo catalogue will still be counted towards the length of the main branch as if it weren't missing.

- **Branching Ratio**
  A further simple tree quantity is the number of branches of the tree. The main branch is included in this count, thus the minimal number of branches for each clump at $z = 0$ will be 1.
  As long as the halo catalogue remains unchanged, lower branching ratios mean less merging events and therefore should be accompanied by longer main branches.
  In the picture of bottom-up structure formation, where larger object form through repeated mergers of smaller ones, one would expect more massive clumps to have longer main branches and a higher branching ratio.

- **Misidentifications, Quantified by Displacements**
  Since no unique correct solution exists, the same displacement statistic $\Delta_r$ as is done in Wang et al. 2016 to quantify misidentifications is used:

$$\Delta_r = \frac{|\mathbf{r}_{k+1} - \mathbf{r}_k - 0.5(\mathbf{v}_{k+1} + \mathbf{v}_k)(t_{k+1} - t_k)|}{0.5(R_{200,k} + R_{200,k+1} + |\mathbf{v}_{k+1} + \mathbf{v}_k|(t_{k+1} - t_k)} \tag{24}$$

  where $\mathbf{r}_{k+1}$, $\mathbf{v}_{k+1}$ and $\mathbf{r}_k$, $\mathbf{v}_k$ are the position and velocity of a clump at snapshot $k + 1$ and its progenitor at snapshot $k$, respectively; $t_{k+1}$ and $t_k$ are the cosmic times at which the two clumps were defined, and $R_{200}$ is the radius that encloses an overdensity of 200 times the critical density $\rho_c = \frac{3H^2}{8\pi G}$.
  It can be interpreted as the difference between the actual displacement $(\mathbf{r}_{k+1} - \mathbf{r}_k)$ and the expected one $(0.5(\mathbf{v}_{k+1} + \mathbf{v}_k)(t_{k+1} - t_k))$, normalized by an estimate of how large the displacement is allowed to be to rule out a clear misidentification. This estimate is the sum of the average halo size, $0.5(R_{200,k} + R_{200,k+1})$, allowing the exact clump position to vary within its own boundaries, and an estimate for the distance travelled based on the time interval and average velocities, $0.5|\mathbf{v}_{k+1} + \mathbf{v}_k|(t_{k+1} - t_k)$.
  Values of $\Delta_r > 1$ would indicate a misidentification, so the parameters minimising $\Delta_r$ should be preferred, provided the acceleration is approximately uniform. This is often not the case for subhaloes, which will shift the distribution of $\Delta_r$ towards higher values. Nevertheless, it is used for subhaloes as well, since the goal is a simple cross-comparison of varying parameters, hoping to get some indication towards which parameters produce the most reliable merger trees.

- **Logarithmic Mass Growth**
  The logarithmic mass growth rate of haloes is approximated discretely by

$$\frac{\mathrm{d}\log M}{\mathrm{d}\log t} \approx \frac{(t_{k+1} + t_k)(M_{k+1} - M_k)}{(t_{k+1} - t_k)(M_{k+1} + M_k)} \equiv \alpha_M(k, k+1) \tag{25}$$

  where $k$ and $k + 1$ are a clump and its descendant, with masses $M_k$ and $M_{k+1}$ at times $t_k$ and $t_{k+1}$, respectively.

To reduce the range of possible values to the interval $(-1, 1)$, Wang et al. 2016 define

$$\beta_M = \frac{2}{\pi} \arctan(\alpha_M) \tag{26}$$

Within the hierarchical structure formation scenario, one would expect haloes to grow over time, thus a distribution of $\beta_M$ should be skewed towards $\beta_M > 0$. $\beta_M \to \pm 1$ imply $\alpha_M \to \pm\infty$, indicating extreme mass growth or losses.

- **Mass Growth Fluctuations**
  Mass growth fluctuations can be quantified by using

$$\xi_M = \frac{\beta_M(k, k+1) - \beta_M(k-1, k)}{2} \tag{27}$$

where $k-1$, $k$, $k+1$ represent consecutive snapshots. When far from zero, it implies an extreme growth behaviour. For $\xi_M \to \pm 1$, $\beta_M(k, k+1) \to \pm 1$ and $\beta_M(k-1, k) \to \mp 1$, indicating extreme mass loss followed by extreme mass growth for the upper sign, and the opposite behaviour for the lower sign. Within the hierarchical structure formation scenario this behaviour shouldn't occur and such an occurrence might indicate either a misidentification by the tree code or an error in the mass assignment of the halo finder.

In the evaluation, no distinction between main haloes and subhaloes is made. Distinguishing between those two cases gives no information on which parameters are preferable that can't already be seen when no distinction is made, so for clarity's sake, the evaluations for main haloes and subhaloes individually are omitted from the main body of this work, but the figures for the displacement statistics, logarithmic mass growth and mass growth fluctuations for haloes and subhaloes separately can be found in appendix B.

### 3.2.2. Parameters Influencing the Halo Catalogue

In the current implementation, there are two parameters which influence the halo catalogue aside from mass and density thresholds. The first one concerns the mass definition of a subhalo. By construction, the mass of a halo contains all its substructure's mass. This isn't necessarily the case for subhaloes though. In a hierarchical structure formation scenario, substructures are expected to contain substructures on their own. Whether to recursively include substructure mass to their respective parent structure is a matter of choice and application. The influence of this choice on the merger trees is shown in figures 9, 10 and 11. When subhaloes' masses are defined to include their respective substructure masses, the results will be labelled as `inclusive`, or `exclusive` otherwise.

A second matter of definition is in which case a particle is to be considered as bound to a clump. The concept of "exclusively bound" particles, which aren't allowed to leave the spatial boundaries of their host clump, was introduced in section 2.3. It is to be expected that demanding particles to be exclusively bound will find more unbound particles than not doing so, thus changing the subhalo catalogue.

The influence of this choice on the merger trees is also shown in figures 9, 10, and 11, along with the influence of the previously described `inclusive` and `exclusive` mass definitions. When bound particles are allowed to leave the clump's boundaries, the results will be labelled as `no saddle`, or `saddle` otherwise.

### 3.2.3. Dataset Used for Testing

All tests are performed on the same dark matter only simulation which contains $256^3 \approx 1.7 \cdot 10^7$ particles of identical mass $m_p = 1.55 \cdot 10^9 M_\odot$. The Hubble constant $H_0 = 70.4$ km s$^{-1}$Mpc$^{-1}$ and density parameters $\Omega_m = 0.272$ and $\Omega_\Lambda = 0.728$ were used. The density threshold for clump finding was chosen to be $80\rho_c$ and the saddle threshold for halos was set to $200\rho_c$, where $\rho_c = \frac{3H^2}{8\pi G}$ is the cosmological critical density. Only clumps with at least 10 particles were kept.

The output strategy was chosen as follows: As virtually no haloes were found before $a \leq 0.1$, only few snapshots were stored up to $a = 0.1$ in steps of $\Delta a \approx 0.02$. From this point on, snapshots were created every $\Delta t \approx 0.3$ Gyrs up until $a = \frac{1}{3}$, after which a smaller time interval of $\Delta_t \approx 0.2$ Gyrs were chosen. This choice resulted in 67 snapshots to get to $z = 0$. The simulation was then continued for 3 further snapshots with $\Delta t \approx 0.2$ Gyrs to ensure that the merging events at $z = 0$ are actually mergers and not clumps that will re-emerge later.

A visualisation of the merger tree of the most massive main halo of this simulation is shown in figure 8. Stunningly, even for a relatively low resolution simulation like the one used, incredibly complex formation histories can be uncovered and followed back to the first snapshot with identifiable haloes. Note that this is only the tree of the central halo, not containing any subhaloes that are still identifiable as such.

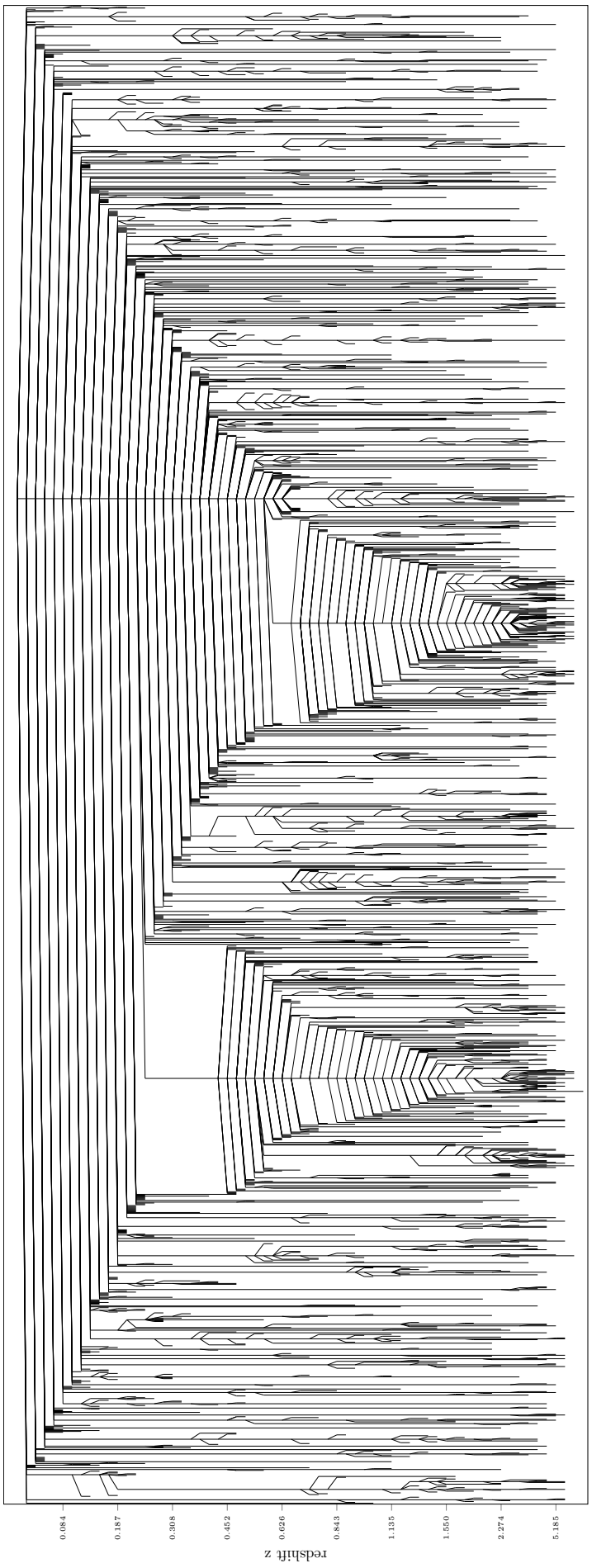


**Figure 8:** The merger tree of the most massive central halo in the simulation, obtained with the parameters `exclusive` and $n_{mb} = 1000$. The redshift at the time of the snapshot is given on the $x$-axis, the $y$-axis has no physical meaning.

### 3.2.4. Influence of the Definition of Subhalo Mass

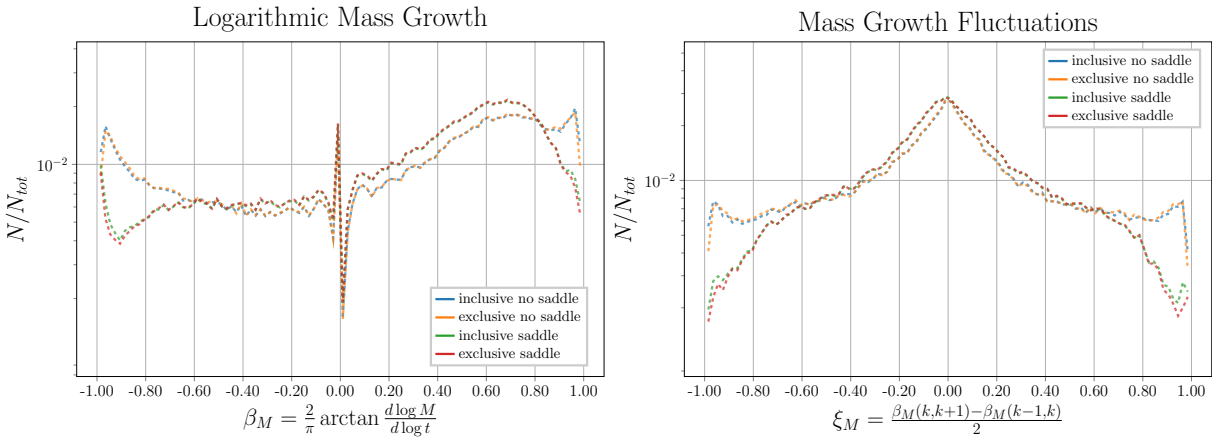


**Figure 9:** Logarithmic mass growth and mass growth fluctuation distributions for progenitor - descendant pairs or three consecutive nodes in a branch, respectively, with masses above $5 \cdot 10^{11} M_\odot$ throughout the entire simulation for the halo catalogue influencing parameters: whether subhalo particles are included (`inclusive`) or excluded (`exclusive`) in the clump mass of satellite haloes, and whether to consider particles which might wander off into another clump as bound (`no saddle`) or not (`saddle`). The distribution is computed as a histogram which is normalised by the total number of events found.

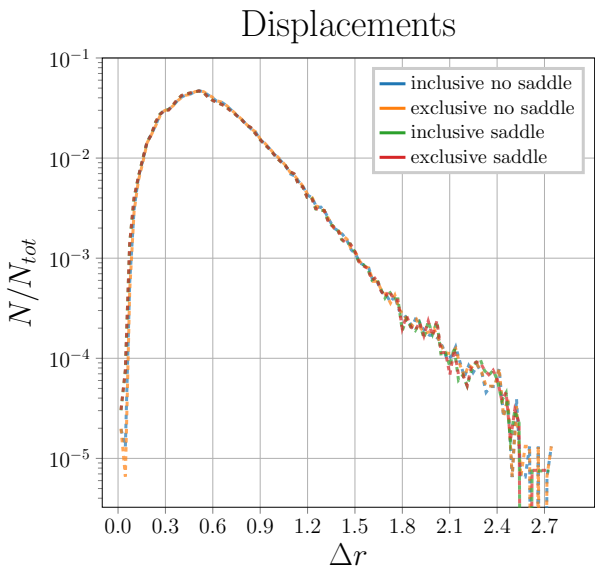


**Figure 11:** Distribution of the displacements for progenitor - descendant pairs with masses above $5 \cdot 10^{11} M_\odot$ throughout the entire simulation for the halo catalogue influencing parameters: whether subhalo particles are included (`inclusive`) or excluded (`exclusive`) in the clump mass of satellite haloes, and whether to consider particles which might wander off into another clump as bound (`no saddle`) or not (`saddle`). The distribution is computed as a histogram which is normalised by the total number of events found.

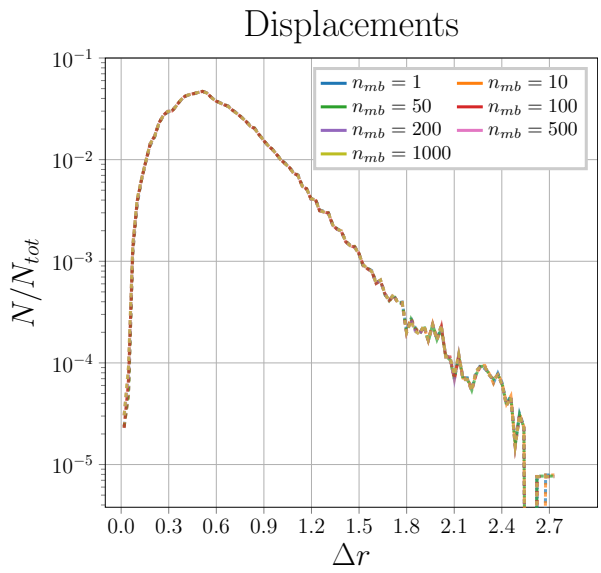


**Figure 12:** Distribution of the displacements for progenitor - descendant pairs with masses above $5 \cdot 10^{11} M_\odot$ throughout the entire simulation for varying numbers of clump tracer particles $n_{mb}$. The distribution is computed as a histogram which is normalised by the total number of events found.

In accordance to the hierarchic structure formation picture, more massive clumps tend to have longer main branches and a higher branching ratio in all cases. This is clearly visible from the

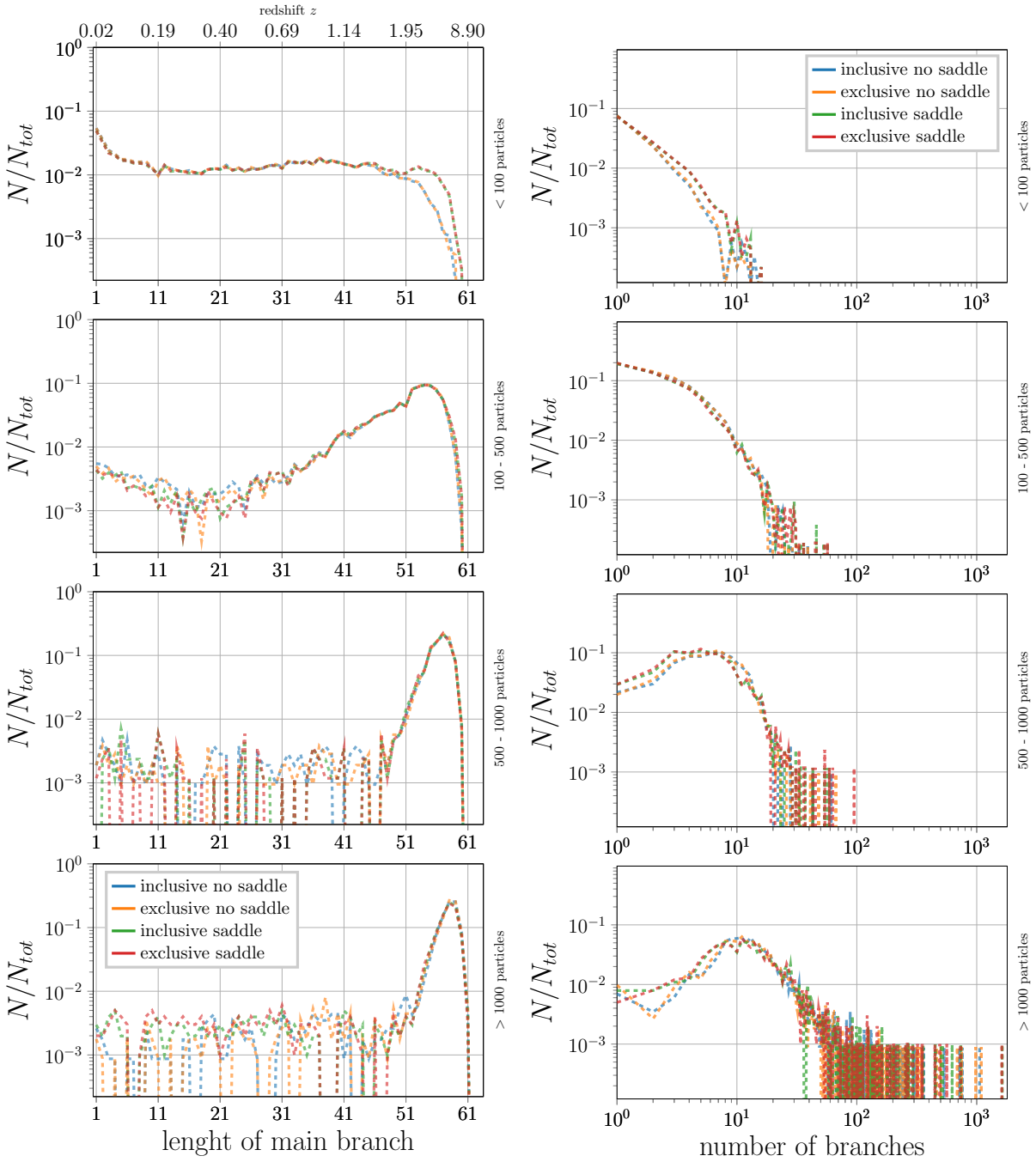


**Figure 10:** Length of main branches, defined as the number of snapshots this clump appears in, and the number of branches including the main branch, for the halo catalogue influencing parameters of $z = 0$ clumps: whether subhalo particles are included (`inclusive`) or excluded (`exclusive`) in the clump mass of satellite haloes, and whether to consider particles which might wander off into another clump as bound (`no saddle`) or not (`saddle`). Four distributions are shown, for four different ranges of numbers of particles at $z = 0$ exclusively assigned to the clump: less then 100 (top), 100-500, 500-1000 and more than 1000 (bottom), where each particle has mass $m_p = 1.55 \cdot 10^9 M_\odot$. The distribution is computed as a histogram which is normalised by the total number of events found per particle count group.

| | `inclusive saddle` | `exclusive saddle` | `inclusive no saddle` | `exclusive no saddle` |
|---|---|---|---|---|
| total clumps | 16262 | 16262 | 17242 | 17242 |
| max number of particles in a clump | 414570 | 414570 | 271438 | 271438 |
| median number of particles in a clump | 84 | 83 | 93 | 93 |
| average main branch length group I | 22.980 | 23.153 | 20.426 | 20.527 |
| average main branch length group II | 48.653 | 48.835 | 48.132 | 48.642 |
| average main branch length group III | 54.358 | 54.715 | 54.305 | 54.904 |
| average main branch length group IV | 54.546 | 53.958 | 56.110 | 56.265 |
| average number of branches group I | 1.318 | 1.327 | 1.230 | 1.230 |
| average number of branches group II | 3.480 | 3.448 | 3.632 | 3.603 |
| average number of branches group III | 8.466 | 8.670 | 8.673 | 8.661 |
| average number of branches group IV | 29.771 | 29.457 | 28.783 | 28.607 |

**Table 1:** Average data for all clumps at $z = 0$ for the four halo catalogue modifying parameter pairs: whether subhalo particles are included (`inclusive`) or excluded (`exclusive`) in the clump mass of satellite haloes, and whether to consider particles which might wander off into another clump as bound (`no saddle`) or not (`saddle`). The groups I, II, III and IV are defined as clumps that contain less then 100, 100-500, 500-1000 or more than 1000 particles, respectively.

average length of the main branch and the average branching ratio of clumps at $z = 0$, binned in four groups by their mass, which are given in table 1. The average main branch length for clumps with more that 500 particles is $\sim 55$, meaning that on average, haloes with mass above $\sim 7.75 \cdot 10^{11} M_\odot$ can be traced back to redshift $\sim 3$.

The length of the main branches for the same four mass bins of clumps are shown in figure 10. Interestingly, small clumps with less than 100 particles seem to have a somewhat constant formation rate from $z \sim 2$ until $z \sim 0.2$. Furthermore they too can be traced back to high redshifts, indicating good performance of the merger tree algorithm.

Whether subhalo masses are computed in an `exclusive` or `inclusive` manner seems to have negligible effect on the branching ratio and the length of main branch.

The `saddle` definition of bound particles however tends to result in longer main branch lengths for small clumps with less than 100 particles. This might be explained by the fact that in general, when `saddle` is applied, subhaloes which are at the bottom of the clump hierarchy will tend to contain less bound particles compared to when `no saddle` is used, and have shorter lifetimes because they are found to have merged into their hosts earlier. Therefore clumps with more than 100 particles with the `no saddle` condition might be moved to the lower mass bin of $\leq 100$ particles when `saddle` is used, thus increasing the fraction of clumps with high main branch lengths (lengths of 50-60), as well as the number of branches. Evidence of the earlier merging can be seen in the overall higher number of branches in the right column of figure 10, as well as the average values, total clump numbers and the median particle numbers in a clump given in table 1.

The displacement statistic used to quantify misidentifications in figure 11 indicates that no parameter choice results in a obviously "wronger" progenitor-descendant pairing. Even though there are some $\Delta_r > 1$, indicating that there might be misidentifications present, it can't be determined easily whether they truly are. When the displacement statistic is calculated for haloes and subhaloes separately, which is shown in appendix B, no halo descendant-progenitor pair shows a displacement above 1, indicating no obvious misidentifications, yet a significant fraction of subhalo progenitor-descendant pairs have a high displacement value $\Delta_r > 1$. However, the displacement statistic is calculated assuming a uniform acceleration, which is not valid for

subhaloes. On the other hand, the fact that any parameter pair used found at least some clumps with more than 1000 particles at $z = 0$ with main branch length of unity, which is the case when it doesn't have any progenitor, and thus essentially "appearing out of nowhere", is strongly suggesting present misidentifications.

The logarithmic mass growth in figure 9 shows that as expected, the distribution is indeed skewed towards $\beta_M > 0$. When the `inclusive` parameter is used, the distribution of mass growth contains a few more extreme mass growths and losses ($\beta_M \to \pm 1$), as well as some high mass growth fluctuations ($\xi_M \to \pm 1$) (see figure 9). When the `no saddle` parameter is used, noticeably more extreme mass growth ($\beta_M \to \pm 1$) and mass growth fluctuations ($\xi_M \to \pm 1$) occur.

In conclusion, whether to use `inclusive` or `exclusive` mass definitions for subhaloes shows very little effect on the merger trees. Based on the fewest extreme mass growth fluctuations, the `saddle` parameter is clearly preferable.

### 3.3. Influence of the Number of Tracer Particles Used

| $n_{mb} =$ | 1 | 10 | 50 | 100 | 200 | 500 | 1000 |
|---|---|---|---|---|---|---|---|
| average MBL group I | 24.188 | 24.330 | 23.567 | 23.353 | 23.153 | 22.876 | 22.656 |
| average MBL group II | 50.399 | 50.116 | 49.472 | 49.122 | 48.835 | 48.777 | 48.762 |
| average MBL group III | 55.233 | 54.863 | 53.264 | 54.059 | 54.715 | 54.327 | 54.165 |
| average MBL group IV | 56.690 | 54.884 | 52.345 | 52.900 | 53.958 | 55.761 | 56.448 |
| average NoB group I | 1.228 | 1.305 | 1.296 | 1.305 | 1.327 | 1.357 | 1.367 |
| average NoB group II | 2.699 | 3.062 | 3.265 | 3.337 | 3.448 | 3.586 | 3.596 |
| average NoB group III | 6.625 | 7.229 | 8.051 | 8.206 | 8.670 | 8.914 | 9.121 |
| average NoB group IV | 20.407 | 25.237 | 27.288 | 28.554 | 29.457 | 30.443 | 31.420 |

**Table 2:** Average data for all clumps at $z = 0$ for varying numbers of clump tracer particles $n_{mb}$. The groups I, II, III and IV are defined as clumps that contain less then 100, 100-500, 500-1000 or more than 1000 particles, respectively. "MBL" is an abbreviation for "main branch length", "NoB" stands for "number of branches".

| $n_{mb} =$ | 1 | 10 | 50 | 100 | 200 | 500 | 1000 |
|---|---|---|---|---|---|---|---|
| trees pruned from tree catalogue | 33924 | 23091 | 22146 | 22131 | 22130 | 22129 | 22129 |
| highest particle number of a LIDIT | 1369 | 236 | 236 | 236 | 236 | 157 | 157 |
| median particle number of a LIDIT | 19 | 20 | 20 | 20 | 20 | 20 | 20 |
| LIDITs with >100 particles pruned | 513 | 42 | 32 | 26 | 25 | 24 | 24 |
| total number of jumpers | 20176 | 20905 | 22074 | 22041 | 20970 | 19307 | 18249 |

**Table 3:** Trees pruned from the merger tree catalogue for varying numbers of clump tracer particles $n_{mb}$ throughout all snapshots. "LIDIT" is an abbreviation of "last identifiable descendant in tree". For LIDITs no further descendants could be identified throughout the simulation and consequently their tree was pruned from the merger tree catalogue. A "jumper" refers to a clump that has been merged into another clump at some snapshot, but re-emerged at a later snapshot, like shown in figure 7.

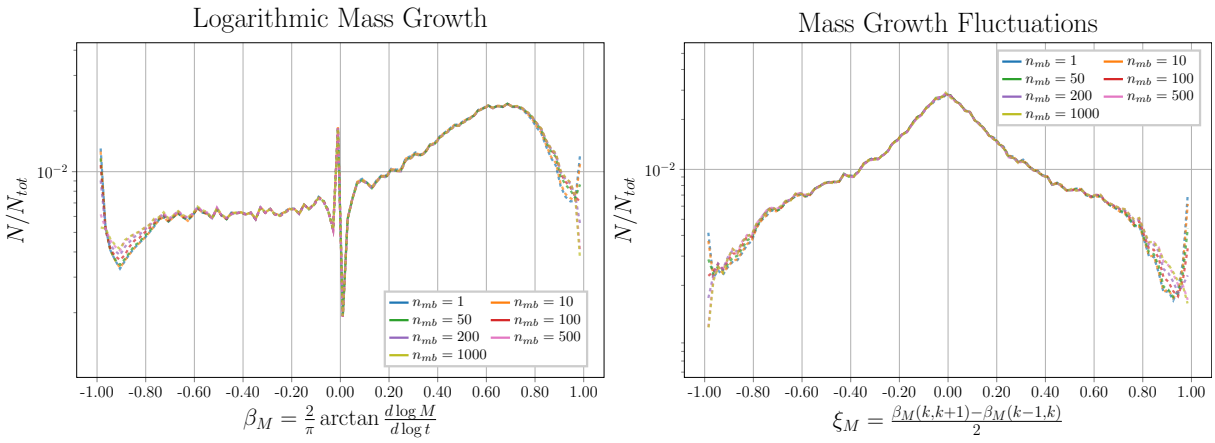


**Figure 13:** Logarithmic mass growth and mass growth fluctuation distributions for progenitor - descendant pairs or three consecutive nodes in a branch, respectively, with masses above $5 \cdot 10^{11} M_\odot$ throughout the entire simulation for varying numbers of clump tracer particles $n_{mb}$. The distribution is computed as a histogram which is normalised by the total number of events found.

The average number of branches and average main branch lengths are shown in table 2. The average number of branches increases with the number of tracers used, the case for $n_{mb} = 10$ for clumps with less than 100 particles being the only exception. The average main branch length decreases for the two lower mass clump bins (less than 500 particles). This can also be seen in the top two rows of figure 14, where the length of the main branches and the number of branches are plotted. This indicates that more mergers were detected. Counter-intuitively, this can be seen as a sign that more reliable trees are created with increasing $n_{mb}$. Recall that progenitors from adjacent snapshots are given priority over non-adjacent "jumpers". By tracking more particles per clump, more candidates can be expected to be found, which is supported by the fact that the number of jumpers in the simulations decreasing with increasing $n_{mb}$ (see table 3). Considering that also being a main progenitor to any descendant is given priority over being merged into the main descendant of the progenitor, it should be safe to say that it should be true merging events that have been misidentified by tracking fewer $n_{mb}$ particles.

When a clump has no descendant candidates at all, its tree is removed from the list of trees. How many of these trees have been pruned throughout the simulation is shown in table 3, as well as the particle number of the most massive pruned clump, the median particle number of pruned clumps and the number of clumps containing more than 100 particles that have been pruned. With increasing $n_{mb}$, the number of pruned clumps, the highest particle number of a pruned clump, and the number of clumps with more than 100 particles decreases. Notice that for $n_{mb} = 1$, there is a drastic increase in all these three quantities. In particular, clumps with more than 1000 particles are pruned, meaning that haloes with mass above $1.5 \cdot 10^{12} M_\odot$ simply vanished between two snapshots. These statistics indicate furthermore that with increasing number of tracing particles, more merging events are detected.

The displacements in figure 12 show virtually no differences. The only noticeable difference is towards the high end of $\Delta_r$, where $n_{mb} = 1$ and $n_{mb} = 10$ have a small peak further out than the other choices for $n_{mb}$.

The logarithmic mass growth and mass growth fluctuations in figure 13 show that these distributions mostly overlap, but extreme growths ($\beta_M \to \pm 1$) and fluctuations ($\xi_M \to \pm 1$) decrease with increasing $n_{mb}$.

It seems that $n_{mb} = 100 - 200$ is a good compromise between computational efficiency and good results. Note that for this simulation, the median number of particles in $z = 0$ clumps was 83,

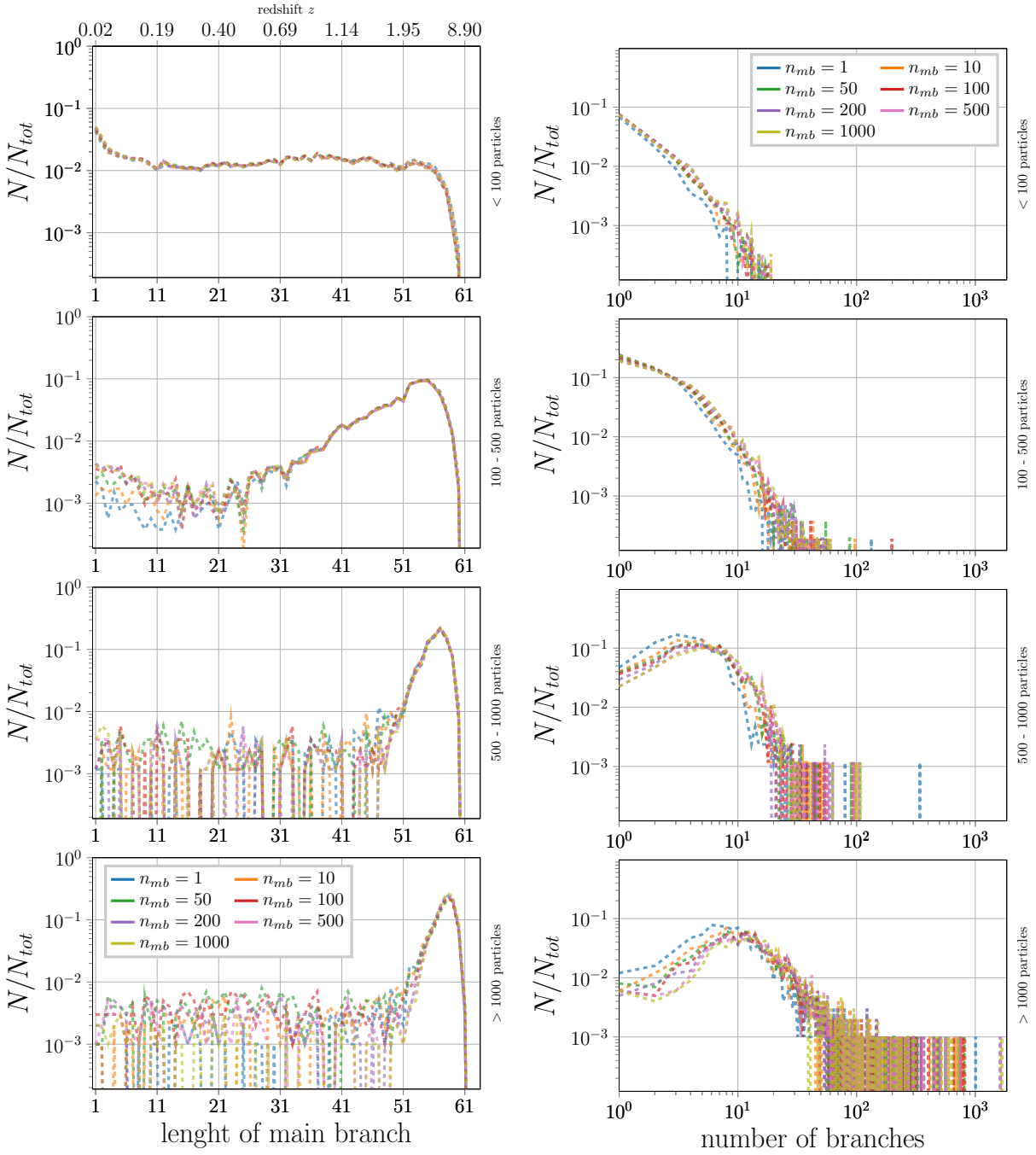


**Figure 14:** Length of main branches, defined as the number of snapshots this clump appears in, and the number of branches including the main branch of $z = 0$ clumps for varying numbers of clump tracer particles $n_{mb}$. Four distributions are shown, for four different ranges of numbers of particles at $z = 0$ exclusively assigned to the clump: less then 100 (top), 100-500, 500-1000 and more than 1000 (bottom), where each particle has mass $m_p = 1.55 \cdot 10^9 M_\odot$. The distribution is computed as a histogram which is normalised by the total number of events found per particle count group.

meaning that with $n_{mb} = 100$, more than half of identified clumps were being tracked by every particle they contain.

### 3.4. Outlook

Based on the previously shown results, the current implementation of the merger tree algorithm seems to perform well. The shapes of the logarithmic mass growths and mass growth fluctuations in figures 9 and 13 as well as the distributions of lengths of main branches and number of branches are in good agreement with the results from other merger tree codes, which have been compared in Avila et al. 2014. However, there are still some unanswered conceptual questions and possible algorithm optimisations to be discussed.

On the conceptual side, when linking progenitors and descendants across multiple snapshots, one must ask: How far in the future or in the past does one need to look for a descendant or progenitor, respectively? At what point should one assume that the tracked progenitor is really dissolved and definitely won't reappear at later times? The current implementation only contains the option to forget past merged progenitors after a user defined number of snapshots has passed, but by default, it will track them until the simulation ends. By not removing orphans at all and using them to link descendants with progenitors across multiple snapshots, misassignments are enabled, leading to wrong formation histories.

Two possible solutions would be the following:

1. Estimate the time a clump would require to completely merge into its parent structure, after which the progenitor shouldn't be tracked anymore. This is for example done in Moster, Naab, and White 2013, where they compute the dynamical friction time $t_{df}$ of a merged subhalo based on the orbital parameters found at the last snapshot where this subhalo was identified:
$$t_{df} = \alpha_{df} \frac{V_{vir} r_{sat}^2}{G M_{sat} \ln \Lambda} \tag{28}$$
where $r_{sat}$ is the distance between the centres of the main halo and of the subhalo, $M_{sat}$ is the mass of the subhalo, $\ln \Lambda = (1 + M_{vir}/M_{sat})$ is the Coulomb logarithm, $M_{vir}$ is the virial mass of the main halo, $V_{vir}$ is the circular velocity of the main halo at the virial radius and $\alpha_{df} = 2.34$. A smaller subhalo inside a main halo experiences dynamical friction because of its gravitational attraction: At any given moment, it attracts the particles of the host towards the point in space where it currently resides, but because the subhalo itself is in orbit, it will move away from that point, thus leaving a slightly denser trail along the path it moves. The gravitational attraction from this trail on the other hand will eventually slow it down and cause it to fall into the main halo's centre.
Another possibility would be to use the fitting formula for the merger timescale of galaxies in cold dark matter models by Jiang et al. 2008.
2. The particle used to track a past merged progenitor is also the same particle that an orphan galaxy is assigned to. In principle, it should be possible to define some galaxy merging cross-sections such that the probability of a collision between an orphan galaxy and a non-orphan galaxy which will result in a galaxy merger can be computed. Unknown parameters of these cross-sections should be able to be calibrated using N-body simulations. After a collision, one could remove the orphan from future snapshots.

From a technical viewpoint, one clear bottleneck in the current merger tree algorithm is the requirement to write progenitor particles and data to file and read them back in and sort them out at a later snapshot. An elegant solution would be to permanently store the clump IDs of

particles in memory, however this would require an extra integer per particle in the simulation, which becomes prohibitively expensive for large simulations not only because it would need a lot of memory, but also because more data needs to be communicated between MPI tasks.

An option would be to track which particles left each task's domain and which particles entered between two snapshots. The clump IDs of particles would still be read and written to and from files, but it would minimise the sorting part of the algorithm where each MPI task figures out which tracker particles it contains. The necessary data of particles that left or entered new domains between snapshots could then be communicated with one collective MPI communication, provided they've been tracked in a clever manner.

Another option would be to change the amount of data each MPI task needs to read in. Currently, every MPI task reads in and writes to one shared file using MPI reading and writing routines in order to maximally make use of the parallel architecture. Instead, each task could write its own file. Meanwhile, between snapshots, the maximal velocity of any particle should be traced. This way, once the simulation advances to the next snapshot, it would be possible to estimate the maximal distance any particle could've travelled. Provided every MPI task has knowledge on how the entire computational domain is split between tasks, it could skip reading in data written by tasks where no particle currently in this task's domain could have come from. This would however probably require a more sophisticated communication for progenitor data such as their mass or descendant candidates. (Currently, because every MPI task reads in all the progenitor data, this communication are simple collective scatter and gather operations.) Furthermore, the situation will get more complicated if the domain decomposition changes its shape between snapshots to e.g. load balance.

# 4. Testing Mock Galaxy Catalogues

## 4.1. Methods

The primary quantity a mock galaxy catalogue must reproduce are stellar mass functions $\Phi(M_*)$, which give the number density of central galaxies with stellar mass $M_*$. The stellar mass functions obtained using the merger tree algorithm and the SMHM relation (15) are showed and discussed in section 4.3

The second test of the mock galaxy catalogues is whether the galaxy clustering of the Universe is reproduced. A commonly used measure of clustering is the two-point correlation function (2PCF) $\xi(r)$, which according to the cosmological principle should be isotropic and thus a function of distance $r$ as opposed to position $\mathbf{r}$. The two-point correlation function can be interpreted as the excess probability of finding a galaxy in a volume element at a separation $r$ from another galaxy, compared to what is expected for a uniform random distribution. It can be computed via inverse Fourier transform of the power spectrum $P(k)$ (Mo, Van den Bosch, and White 2011), which itself can be obtained from the Fourier transform of the density contrast field $\delta(\mathbf{r})$:

$$\delta_{\mathbf{k}} = \frac{1}{V} \int e^{i\mathbf{kr}} \delta(\mathbf{r}) \mathrm{d}^3\mathbf{r} \tag{29}$$

with

$$\delta(\mathbf{r}) = \frac{\rho(\mathbf{r})}{\langle \rho(\mathbf{r}) \rangle} - 1 \tag{30}$$

Where $\rho(\mathbf{r})$ is the galaxy density field and $\langle\rho(\mathbf{r})\rangle$ is the mean density, $V = L^3$ is the volume of a large box on which the density field is assumed periodic, and $k = \frac{2\pi}{L}(i_x, i_y, i_z)$, where $i_x, i_y, i_z$ are integers.

The power spectrum $P(k)$ and the 2PCF $\xi(r)$ are given by

$$P(k) = V\langle|\delta_{\mathbf{k}}|^2\rangle \tag{31}$$

$$\xi(r) = \frac{1}{(2\pi)^3}\int e^{-i\mathbf{kr}}P(k)\mathrm{d}^3\mathbf{k} \tag{32}$$

$$= \frac{1}{2\pi^2}\int_0^\infty P(k)\frac{\sin(kr)}{kr}k^2\mathrm{d}k \tag{33}$$

The simulation box is divided in a uniform grid of $1024^3$ cells and the mass is distributed using a cloud-in-cell interpolation scheme to obtain the density field. The Fourier transforms are performed using the FFTW library (Frigo and Johnson 2005). Instead of first averaging the tree-dimensional power spectrum $P(\mathbf{k})$ to obtain a one-dimensional power spectrum $P(k)$ and then integrating it following eq. (33), first the three-dimensional correlation function $\xi(\mathbf{r})$ is computed by computing the inverse Fourier transform on the three-dimensional power spectrum $P(\mathbf{k})$ and then $\xi(\mathbf{r})$ is averaged over all angular directions to obtain $\xi(r)$. This has the advantage of not having to perform an integral to infinity while only a finite sample is present.

Once the real space correlation function is known, the projected correlation function $w_p(r_p)$ can be derived by integrating $\xi(r)$ along the line of sight (Moster et al. 2010):

$$w_p(r_p) = 2\int_0^\infty \mathrm{d}r_{||}\xi\left(\sqrt{r_{||}^2 + r_p^2}\right) = 2\int_{r_p}^\infty \mathrm{d}r\frac{r\xi(r)}{\sqrt{r^2 - r_p^2}} \tag{34}$$

where the comoving distance $r$ has been decomposed into components parallel ($r_{||}$) and perpendicular ($r_p$) along the line of sight. The integration is truncated at half the length of the simulation box.

The obtained correlations are showed and discussed in section 4.4.

## 4.2. Dataset Used for Testing

Mock galaxy catalogues from two simulations were created, each containing $512^3 \approx 1.3 \cdot 10^8$ particles. They differ in the volume they simulate: `G69` covers 69 comoving Mpc, while the second simulation, `G100`, contains 100 comoving Mpc.

With different box sizes come different mass resolutions: The particle mass for `G69` is $m_p = 9.59 \cdot 10^7 M_\odot$, for `G100` it is $m_p = 3.09 \cdot 10^8 M_\odot$.

The cosmological parameters are taken from the 2015 Plack Collaboration results (Planck Collaboration et al. 2016): The Hubble constant $H_0 = 67.74$ km s$^{-1}$Mpc$^{-1}$, density parameters $\Omega_m = 0.309$, $\Omega_\Lambda = 0.691$, scalar spectral index $n_s = 0.967$, and fluctuation amplitude $\sigma_8 = 0.816$ were used. The initial conditions were created using the `MUSIC` code (Hahn and Abel 2011).

As before, the density threshold for clump finding was chosen to be $80\rho_c$ and the saddle threshold for halos was set to $200\rho_c$, where $\rho_c = \frac{3H^2}{8\pi G}$ is the cosmological critical density. Only clumps with at least 10 particles were kept.

**Table 4:** Redshift interval of observed stellar mass functions that are used for comparison and the abbreviation used in this work as reference.

| redshift interval | Reference | Abbreviation |
|---|---|---|
| $z \sim 0 - 1$ | Moustakas et al. 2013 | MOU |
| $z \sim 0 - 4$ | Pérez-González et al. 2008 | PG |
| $z \sim 1 - 3.5$ | Mortlock et al. 2011 | MOR |
| $z \sim 1.3 - 4.0$ | Marchesini et al. 2009 | MAR |
| $z \sim 4$, $z \sim 5$ | Lee et al. 2012 | KSL |
| $z \sim 4 - 6$ | Stark et al. 2009 | ST |
| $z \sim 7$, $z \sim 8$ | Bouwens et al. 2011 | BOU |

### 4.3. Stellar Mass Functions

The obtained stellar mass functions $\Phi(M_*)$ of central galaxies for the two simulations, `G69` and `G100`, compared to observed stellar mass functions are shown in figure 15. The abbreviations used for observational data are listed in table 4. For clarity's sake, only stellar mass functions from snapshots at redshifts which are closest to the mean redshift of the observational data are plotted. Averaging the stellar mass function over the redshift interval made very little difference compared to choosing only one closest to the mean of the interval, which is why the average stellar mass functions were omitted from the main body of this work. As an example, the plot of both average and a single stellar mass functions for the `G69` simulation are shown in appendix C.

The `G69` simulation gives better results at the low mass end at $z \sim 0$, and starts to deviate noticeably around $M_* \sim 10^8 M_\odot$. Using a crude estimate that the SMHM ratio $M_*/M_h \sim 10^{-1} - 10^{-2}$, together with a lower mass threshold of $10m_p \sim 10^9 M_\odot$ for clumps, one can see that $M_* \sim 10^8 M_\odot$ should be the lower mass threshold for stellar mass that is accurately resolved. Furthermore, the "shoulder" of the SMF around $\log_{10} M \sim 10 - 12$ appears flatter. This flattening seems to produce results closer to observations in most redshift intervals. Also, at $z \sim 0$, the high mass end of the SMF is underestimated. Because high mass central galaxies are hosted by high mas haloes, the simulation volume just might be too small to accurately represent the statistics of the high mass halo abundance. This is supported by the fact that the `G100` simulation gives slightly better results at the high mass end.

The results seem quite good and within the uncertainties of the observed data up until $z \sim 1.65$, where the deviations look like they're often outside the error bars. It gets worse with increasing redshift. However, seeing how in almost every case the `G69` yields better results than `G100`, the high redshift SMFs should improve with increased resolution.

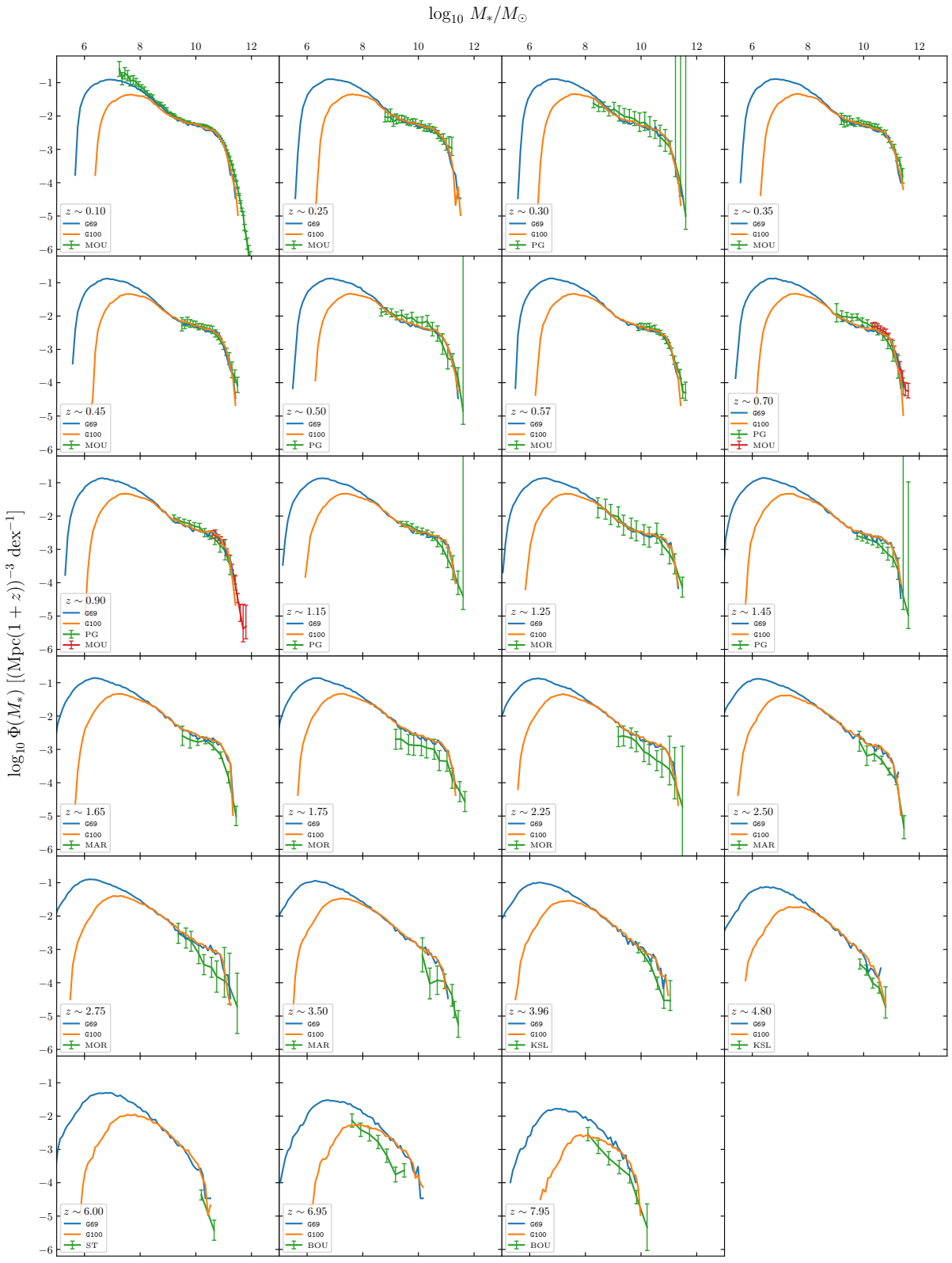


**Figure 15:** Obtained stellar mass functions $\Phi(M_*)$ of central galaxies for the two simulation datasets G69 and G100, described in 4.2 with boxsize of 69 and 100 Mpc, respectively, compared to observed stellar mass function. The abbreviations used for observational data are listed in table 4.

## 4.4. Correlation Functions

The obtained 2PCF $\xi(r)$ and the projected correlation function $w_p(r_p)$ at $z \sim 0$ are shown in figure 16 for both the G69 and G100 simulations, and they are compared to observational results from Li and White 2009 and Zehavi et al. 2004.

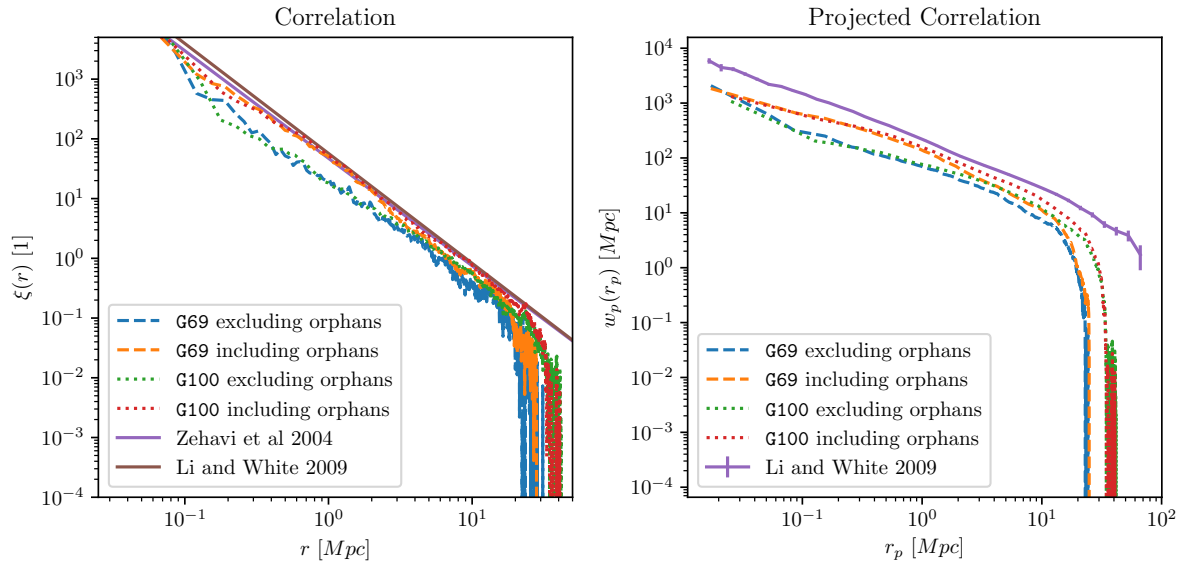


**Figure 16:** The obtained 2PCF $\xi(r)$ and projected correlation function $w_p(r_p)$ for the G69 (dashed lines) and G100 (dotted lines) simulations, both including and excluding orphan galaxies, compared to best power law fits of the 2PCF from Li and White 2009 and Zehavi et al. 2004 and the projected correlation function from Li and White 2009 (solid lines).

In all cases, including orphan galaxies produced correlation functions closer to observations. Campbell et al. 2018 also found that the inclusion of orphan galaxies for mass-based SHAM models may improve the clustering statistics of mock galaxy catalogues, particularly so at small scales. The 2PCF obtained from the G100 simulation even can reproduce the best power law fit from observations quite well for about two orders of magnitude of $r \sim 0.2 - 20$ Mpc. Noticeably for both simulations the correlation functions start with very similar values regardless of whether orphan galaxies are included or not for small $r$. As the distance $r$ increases, so does the difference between the two cases, and starts decreasing around $r \sim 1 - 2$ Mpc. After $r \sim 10$ Mpc, the difference becomes very small. This behaviour may be explained by considering that one would expect orphan galaxies to tend to be located within host halos, not somewhere in a void all by themselves, thus contributing to the correlations at small distances stronger than at large distances.

Any case underestimates the projected correlation function, but as for the 2PCF, the G100 simulation with orphan galaxies included in the computation of the correlation comes closest. Because $w_p$ is computed by numerically integrating the previously obtained $\xi(r)$, part of the reason why $w_p$ might be underestimated is the propagation of errors. $w(r_p)$ is computed by integrating $\xi(r)$ from $r = r_p$ to $r \to \infty$, meaning that the underestimated $\xi(r)$ at large scales $r \gtrsim 20$ Mpc is included in the integration for every $r_p$. Another reason might be that while technically the integration should be performed to infinity, it was truncated at half of the length of a simulation box.

The G100 simulation gives better correlation functions, which might be due to the fact that a

bigger volume was simulated. A volume of 69 Mpc might just be too small to properly represent a homogeneous, isotropic chunk of the Universe.

# 5. Conclusion

An algorithm to identify dark matter halo merger trees, designed to work on the fly on parallel computing systems with the adaptive mesh refinement code `RAMSES`, was presented. Clumps of dark matter are tracked across snapshots through up to some user-defined number, $n_{mb}$, of particles they consist of. The best choice for $n_{mb}$ seems to be around 100-200, where the trade-off between computational cost and performance appears optimal. Furthermore, the influence of various definitions of substructure properties on the resulting merger tree were tested. Whether substructures contain their respective substructures' masses or not had negligible effect on the merger trees. However defining particles of substructure to be gravitationally bound to that substructure if and only if the particles can't leave the spatial boundaries of that substructure leads to much less extreme mass growths and extreme mass growth fluctuations of dark matter clumps, suggesting that it should be the preferred definition for accurate merger trees.

In agreement with the bottoms-up hierarchical structure formation picture for dark matter haloes, the merger trees of massive haloes at $z = 0$ were found to tend to have more branches and their formation history can often be traced to very high redshifts. Even clumps on the lower mass end were successfully tracked back to high redshifts.

With the known formation history of dark matter clumps, using a stellar-mass-halo-mass relation (eqns. (19)-(23)) galaxies can be placed in a dark matter only simulation to obtain mock galaxy catalogues. The galaxies are placed at the position of the most tightly bound particle of any dark matter clump. Once a clump merges into another and dissolves beyond the possibility of identification, its last assigned galaxy is kept track of. Such an "orphan" galaxy is used two reasons. Firstly, just because a clump can't be identified any more due to the environment it currently resides in, it doesn't mean that the galaxy that it hosted is also dissipated. On the contrary: Nagai and Kravtsov 2005 showed that tidal stripping of galaxies inside a dark matter halo sets in much later than for the subhalo they reside in. Secondly, it is possible for a subhalo to re-emerge from its host halo at later snapshots because it wasn't detected by the clump finder in the density field of the host halo, but still existed. Such a situation is illustrated in figure 7. In these cases, orphan galaxies are used to establish a link between progenitor and descendant clumps across multiple snapshots.

However, the current implementation only contains the option to forget past merged progenitors after a user defined number of snapshots has passed, but by default, it will track them until the simulation ends. This might lead to misidentifications of progenitor-descendant pairs and therefore wrong formation histories. Solutions for this problem would be either to remove the orphans after the estimated time for them to merge into the parent structure has passed, which could be e.g. the dynamical friction time (eq. (28)), or introduce some form of galaxy-galaxy merging cross sections to compute the probability of a collision between galaxies that will result in a galaxy merger.

From a technical viewpoint, one clear bottleneck in the current merger tree algorithm is the requirement to write progenitor particles and data to file and read them back in and sort them out at a later snapshot. A possible improvement would be to track which particles left each task's domain and which particles entered in between two snapshots. The progenitor particles would still be read and written to and from files, but it would minimise the sorting part of the

algorithm where each MPI task figures out which tracker particles it contains. Another option would be to change the amount of data each MPI task needs to read in. The maximal velocity of any particle in the time interval between two snapshots should be traced. This way, once the simulation advances to the next snapshot, it would be possible to estimate the maximal distance any particle could've travelled. Provided every MPI task has knowledge on how the entire computational domain is split between tasks, it could skip reading in data written by tasks where no particle currently in this task's domain could have come from.

Given that the mock galaxy catalogues in this work were created using simulations with relatively low spatial and mass resolution of $512^3$ particles in boxes of 69 and 100 comoving Mpc each, the obtained correlation functions (shown in figure 16) and stellar mass functions (figure 15) show good agreement with observed stellar mass functions. By comparing the results of the two simulations it can be expected that a higher spatial resolution should improve the clustering statistics, and together with a higher mass resolution the stellar mass functions of central galaxies should also improve.

# References


[1] S. Avila et al. "SUSSING MERGER TREES: the influence of the halo finder". In: *Monthly Notices of the RAS* 441 (July 2014), pp. 3488–3501. DOI: 10.1093/mnras/stu799. arXiv: 1402.2381.

[2] P. S. Behroozi, R. H. Wechsler, and C. Conroy. "The Average Star Formation Histories of Galaxies in Dark Matter Halos from z = 0-8". In: *The Astrophysical Journal* 770, 57 (June 2013), p. 57. DOI: 10.1088/0004-637X/770/1/57. arXiv: 1207.6105 [astro-ph.CO].

[3] P. S. Behroozi et al. "Gravitationally Consistent Halo Catalogs and Merger Trees for Precision Cosmology". In: *The Astrophysical Journal* 763, 18 (Jan. 2013), p. 18. DOI: 10.1088/0004-637X/763/1/18. arXiv: 1110.4370 [astro-ph.CO].

[4] M. J. Berger and P. Colella. "Local adaptive mesh refinement for shock hydrodynamics". In: *Journal of Computational Physics* 82 (May 1989), pp. 64–84. DOI: 10.1016/0021-9991(89)90035-1.

[5] A. A. Berlind and D. H. Weinberg. "The Halo Occupation Distribution: Toward an Empirical Determination of the Relation between Galaxies and Mass". In: *The Astrophysical Journal* 575 (Aug. 2002), pp. 587–616. DOI: 10.1086/341469. eprint: astro-ph/0109001.

[6] Andreas Bleuler et al. "PHEW: a parallel segmentation algorithm for three-dimensional AMR datasets". In: *Computational Astrophysics and Cosmology* 2.1 (2015), pp. 1–16. ISSN: 2197-7909. DOI: 10.1186/s40668-015-0009-7. URL: http://dx.doi.org/10.1186/s40668-015-0009-7.

[7] R. J. Bouwens et al. "Ultraviolet Luminosity Functions from 132 $z \sim 7$ and $z \sim 8$ Lyman-break Galaxies in the Ultra-deep HUDF09 and Wide-area Early Release Science WFC3/IR Observations". In: *The Astrophysical Journal* 737.2 (2011), p. 90. URL: http://stacks.iop.org/0004-637X/737/i=2/a=90.

[8] R. G. Bower et al. "Breaking the hierarchy of galaxy formation". In: *Monthly Notices of the RAS* 370 (Aug. 2006), pp. 645–655. DOI: 10.1111/j.1365-2966.2006.10519.x. eprint: astro-ph/0511338.

[9] D. Campbell et al. "The galaxy clustering crisis in abundance matching". In: *Monthly Notices of the RAS* 477 (June 2018), pp. 359–383. DOI: 10.1093/mnras/sty495. arXiv: 1705.06347.

[10] J. Chaves-Montero et al. "Subhalo abundance matching and assembly bias in the EAGLE simulation". In: *Monthly Notices of the RAS* 460 (Aug. 2016), pp. 3100–3118. DOI: 10.1093/mnras/stw1225. arXiv: 1507.01948.

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

[12] Matteo Frigo and Steven G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005). Special issue on "Program Generation, Optimization, and Platform Adaptation", pp. 216–231.

[13] O. Hahn and T. Abel. "Multi-scale initial conditions for cosmological simulations". In: *Monthly Notices of the RAS* 415 (Aug. 2011), pp. 2101–2121. DOI: 10.1111/j.1365-2966.2011.18820.x. arXiv: 1103.6031.

[14] Mladen Ivkovic. "Halo and Sub-Halo Finding in Cosmological N-body Simulations". Unpublished bachelor thesis. 2017.

[15] C. Y. Jiang et al. "A Fitting Formula for the Merger Timescale of Galaxies in Hierarchical Clustering". In: *The Astrophysical Journal* 675 (Mar. 2008), pp. 1095–1105. DOI: 10.1086/526412. arXiv: 0707.2628.

[16] L. Jiang et al. "N-body dark matter haloes with simple hierarchical histories". In: *Monthly Notices of the RAS* 440 (May 2014), pp. 2115–2135. DOI: 10.1093/mnras/stu390. arXiv: 1311.6649.

[17] G. Kauffmann, S. D. M. White, and B. Guiderdoni. "The Formation and Evolution of Galaxies Within Merging Dark Matter Haloes". In: *Monthly Notices of the RAS* 264 (Sept. 1993), p. 201. DOI: 10.1093/mnras/264.1.201.

[18] A.M Khokhlov. "Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations". In: *J. Comput. Phys.* 143.2 (July 1998), pp. 519–543. ISSN: 0021-9991. DOI: 10.1006/jcph.1998.9998. URL: http://dx.doi.org/10.1006/jcph.1998.9998.

[19] A. Knebe et al. "Haloes gone MAD: The Halo-Finder Comparison Project". In: *Monthly Notices of the RAS* 415 (Aug. 2011), pp. 2293–2318. DOI: 10.1111/j.1365-2966.2011.18858.x. arXiv: 1104.0949.

[20] S. R. Knollmann and A. Knebe. "AHF: Amiga's Halo Finder". In: *The Astrophysical Journal* 182 (June 2009), pp. 608–624. DOI: 10.1088/0067-0049/182/2/608. arXiv: 0904.3662.

[21] A. V. Kravtsov et al. "The Dark Side of the Halo Occupation Distribution". In: *The Astrophysical Journal* 609 (July 2004), pp. 35–49. DOI: 10.1086/420959. eprint: astro-ph/0308519.

[22] Kyoung-Soo Lee et al. "How Do Star-forming Galaxies at z > 3 Assemble Their Masses?" In: *The Astrophysical Journal* 752.1 (2012), p. 66. URL: http://stacks.iop.org/0004-637X/752/i=1/a=66.

[23] C. Li and S. D. M. White. "The distribution of stellar mass in the low-redshift Universe". In: *Monthly Notices of the RAS* 398 (Oct. 2009), pp. 2177–2187. DOI: 10.1111/j.1365-2966.2009.15268.x. arXiv: 0901.0706.

[24] D. Marchesini et al. "The Evolution of the Stellar Mass Function of Galaxies from z = 4.0 and the First Comprehensive Analysis of its Uncertainties: Evidence for Mass-Dependent Evolution". In: *The Astrophysical Journal* 701 (Aug. 2009), pp. 1765–1796. DOI: 10.1088/0004-637X/701/2/1765. arXiv: 0811.1773.

[25] Houjun Mo, Frank Van den Bosch, and Simon White. *Galaxy formation and evolution*. Cambridge University Press, 2011.

[26] Alice Mortlock et al. "A deep probe of the galaxy stellar mass functions at $z \sim 1-3$ with the GOODS NICMOS Survey". In: *Monthly Notices of the Royal Astronomical Society* 413.4 (2011), pp. 2845–2859. DOI: 10.1111/j.1365-2966.2011.18357.x. eprint: /oup/backfile/content_public/journal/mnras/413/4/10.1111/j.1365-2966.2011.18357.x/2/mnras0413-2845.pdf. URL: http://dx.doi.org/10.1111/j.1365-2966.2011.18357.x.

[27] B. P. Moster, T. Naab, and S. D. M. White. "Galactic star formation and accretion histories from matching galaxies to dark matter haloes". In: *Monthly Notices of the RAS* 428 (Feb. 2013), pp. 3121–3138. DOI: 10.1093/mnras/sts261. arXiv: 1205.5807.

[28] B. P. Moster et al. "Constraints on the Relationship between Stellar Mass and Halo Mass at Low and High Redshift". In: *The Astrophysical Journal* 710 (Feb. 2010), pp. 903–923. DOI: 10.1088/0004-637X/710/2/903. arXiv: 0903.4682 [astro-ph.CO].

[29] John Moustakas et al. “PRIMUS: Constraints on Star Formation Quenching and Galaxy Merging, and the Evolution of the Stellar Mass Function from z = 0-1”. In: *The Astrophysical Journal* 767.1 (2013), p. 50. URL: http://stacks.iop.org/0004-637X/767/i=1/a=50.

[30] D. Nagai and A. V. Kravtsov. “The Radial Distribution of Galaxies in Λ Cold Dark Matter Clusters”. In: *The Astrophysical Journal* 618 (Jan. 2005), pp. 557–568. DOI: 10.1086/426016. eprint: astro-ph/0408273.

[31] V. Perret. *DICE: Disk Initial Conditions Environment*. Astrophysics Source Code Library. July 2016. ascl: 1607.002.

[32] Planck Collaboration et al. “Planck 2015 results. XIII. Cosmological parameters”. In: *Astronomy and Astrophysics* 594, A13 (Sept. 2016), A13. DOI: 10.1051/0004-6361/201525830. arXiv: 1502.01589.

[33] Planelles, S. and Quilis, V. “ASOHF: a new adaptive spherical overdensity halo finder”. In: *Astronomy and Astrophysics* 519 (2010), A94. DOI: 10.1051/0004-6361/201014214. URL: https://doi.org/10.1051/0004-6361/201014214.

[34] D. Potter, J. Stadel, and R. Teyssier. “PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys”. In: *Computational Astrophysics and Cosmology* 4, 2 (May 2017), p. 2. DOI: 10.1186/s40668-017-0021-1. arXiv: 1609.08621 [astro-ph.IM].

[35] W. H. Press and P. Schechter. “Formation of Galaxies and Clusters of Galaxies by Self-Similar Gravitational Condensation”. In: *The Astrophysical Journal* 187 (Feb. 1974), pp. 425–438. DOI: 10.1086/152650.

[36] Pablo G. Pérez-González et al. “The Stellar Mass Assembly of Galaxies from z = 0 to z = 4: Analysis of a Sample Selected in the Rest-Frame Near-Infrared with Spitzer”. In: *The Astrophysical Journal* 675.1 (2008), p. 234. URL: http://stacks.iop.org/0004-637X/675/i=1/a=234.

[37] U. Seljak. “Analytic model for galaxy and dark matter clustering”. In: *Monthly Notices of the RAS* 318 (Oct. 2000), pp. 203–213. DOI: 10.1046/j.1365-8711.2000.03715.x. eprint: astro-ph/0001493.

[38] R. S. Somerville and J. R. Primack. “Semi-analytic modelling of galaxy formation: the local Universe”. In: *Monthly Notices of the RAS* 310 (Dec. 1999), pp. 1087–1110. DOI: 10.1046/j.1365-8711.1999.03032.x. eprint: astro-ph/9802268.

[39] V. Springel et al. “Populating a cluster of galaxies - I. Results at [formmu2]z=0”. In: *Monthly Notices of the RAS* 328 (Dec. 2001), pp. 726–750. DOI: 10.1046/j.1365-8711.2001.04912.x. eprint: astro-ph/0012055.

[40] V. Springel et al. “Simulations of the formation, evolution and clustering of galaxies and quasars”. In: *Nature* 435 (June 2005), pp. 629–636. DOI: 10.1038/nature03597. eprint: astro-ph/0504097.

[41] C. Srisawat et al. “Sussing Merger Trees: The Merger Trees Comparison Project”. In: *Monthly Notices of the RAS* 436 (Nov. 2013), pp. 150–162. DOI: 10.1093/mnras/stt1545. arXiv: 1307.3577 [astro-ph.CO].

[42] J. G. Stadel. “Cosmological N-body simulations and their analysis”. PhD thesis. UNIVERSITY OF WASHINGTON, 2001.

[43] Daniel P. Stark et al. “The Evolutionary History of Lyman Break Galaxies Between Redshift 4 and 6: Observing Successive Generations of Massive Galaxies in Formation”. In: *The Astrophysical Journal* 697.2 (2009), p. 1493. URL: http://stacks.iop.org/0004-637X/697/i=2/a=1493.

[44] Teyssier, R. "Cosmological hydrodynamics with adaptive mesh refinement. A new high resolution code called RAMSES". In: *Astronomy and Astrophysics* 385 (Apr. 2002), pp. 337–364. DOI: 10.1051/0004-6361:20011817. eprint: astro-ph/0111367.

[45] A. Vale and J. P. Ostriker. "Linking halo mass to galaxy luminosity". In: *Monthly Notices of the RAS* 353 (Sept. 2004), pp. 189–200. DOI: 10.1111/j.1365-2966.2004.08059.x. eprint: astro-ph/0402500.

[46] A. Vale and J. P. Ostriker. "The non-parametric model for linking galaxy luminosity with halo/subhalo mass". In: *Monthly Notices of the RAS* 371 (Sept. 2006), pp. 1173–1187. DOI: 10.1111/j.1365-2966.2006.10605.x. eprint: astro-ph/0511816.

[47] Y. Wang et al. "Sussing merger trees: stability and convergence". In: *Monthly Notices of the RAS* 459 (June 2016), pp. 1554–1568. DOI: 10.1093/mnras/stw726. arXiv: 1604.01463.

[48] R. H. Wechsler and J. L. Tinker. "The Connection between Galaxies and their Dark Matter Halos". In: *ArXiv e-prints* (Apr. 2018). arXiv: 1804.03097.

[49] S. D. M. White and C. S. Frenk. "Galaxy formation through hierarchical clustering". In: *The Astrophysical Journal* 379 (Sept. 1991), pp. 52–79. DOI: 10.1086/170483.

[50] X. Yang, H. J. Mo, and F. C. van den Bosch. "The Subhalo-Satellite Connection and the Fate of Disrupted Satellite Galaxies". In: *The Astrophysical Journal* 693 (Mar. 2009), pp. 830–838. DOI: 10.1088/0004-637X/693/1/830. arXiv: 0808.2526.

[51] X. Yang et al. "Evolution of the Galaxy-Dark Matter Connection and the Assembly of Galaxies in Dark Matter Halos". In: *The Astrophysical Journal* 752, 41 (June 2012), p. 41. DOI: 10.1088/0004-637X/752/1/41. arXiv: 1110.1420.

[52] I. Zehavi et al. "On Departures from a Power Law in the Galaxy Correlation Function". In: *The Astrophysical Journal* 608 (June 2004), pp. 16–24. DOI: 10.1086/386535. eprint: astro-ph/0301280.

# Appendix A Detailed Description of the Merger Tree algorithm

The merger tree code essentially consists of two steps:

1. Create trees using progenitor data that was previously written to file and descendant data which is currently in memory. The clumps identified in the snapshot where the simulation currently is are treated as descendants, while the clumps from past snapshots are considered to be progenitors.
2. Prepare and write data of current clumps to file. This data will be the progenitor data in the following snapshot.

Suppose the simulation is at the first snapshot that contains haloes. As there are no progenitors available at this point, no trees can be made, so the code directly jumps to step 2:

- For every clump, identify up to $n_{mb}$ tracker particles with minimal energy across all processing units. If a clump consists of less than $n_{mb}$ particles, then take the maximally available number of particles.
- Write the tracker particles of all processing units into a single shared file. All processing units will read this file back in at the following snapshot. If past merged progenitors exist, also write these to (a different) shared file.
- Remove all clump finding data from memory and continue with the simulation.

At the next snapshot, the merger tree code will start once haloes have been identified. This time, progenitors exist, so the code proceeds as follows:

- Every processing unit reads in the progenitor data from the shared file of the previous snapshot.
- Process the progenitor data:
  - Find which tracer particles are on each processing unit's domain by checking the particles' global ID. Each processing unit needs to know which tracer particles are currently on its domain.
  - Find and communicate globally which processing unit is the "owner" of which progenitor (and past merged progenitor): The owner of any progenitor is defined as the processing unit which has the most strongly bound particle of that progenitor within its domain. (Analogously as for the past merged progenitors, this particle is referred to as the "galaxy particle" of this progenitor.)
  - Each processing unit henceforth only keeps track of the tracer particles that are on its domain. The rest are removed from memory.
- Find links between progenitors and descendants: Essentially find "which tracer particle ended up where":
  - After halo finding the halo to which any particle belongs is known.
  - After reading in progenitor data the progenitor halo to which any tracer particle belonged is known.
  - Each processing unit loops through all its local tracer particles. Using these two informations (in which halo the particle was and in which halo the particle is now) for every tracer particle, all descendant candidates for all progenitors are found and stored in a sparse matrix, where the rows correspond to progenitors and the columns are the descendants. The exact number of particle matches between a progenitor-descendant candidate pair is kept. Example: let $n_{mb} = 200$. For progenitor with ID 1, a possible result would be to find 50 particles in descendant with ID 2, 120

particles in descendant with ID 7, 10 particles in descendant 3 and 20 particles that aren't in a halo at the current snapshot.
  - The owner of progenitors gather and sum up all the matches found this way for that progenitor and then scatter them back to any processing unit that has at least one particle of that progenitor on their domain. (These are exactly the processing units that sent data to the owner of the progenitor in the first place.)
  - After communications are done, create the transverse sparse matrix, where the rows are descendants and the columns are progenitors. These matrices will be used to loop through progenitor or descendant candidates.
- Make trees:
  - Obtain an initial guess for the main progenitors of every descendant and for the main descendant of every progenitor by finding the candidate that maximises the merit function (14).
  - Loop to establish matches:
    A main progenitor-descendant pair is established when the main progenitor of a descendant is the main descendant of said progenitor, or in pseudocode:

    ```
    match = (main_prog(idesc)==iprog) && (main_desc(iprog) == idesc)
    ```

    While there are still descendants without a match and still progenitor candidates left for these descendants:
    * Loop through all descendant candidates of progenitors without a match, unless you find a match.
    * For all descendants without a match: Switch to the next best progenitor candidate as current best guess.

    The loop ends either when all descendants have a match, or if descendants run out of candidates.
    If a progenitor hasn't found a match, assume it merged into its best descendant candidate.
  - Add merged progenitors to the list of past merged progenitors.
  - If there are descendants that still have no main progenitor: Try finding a progenitor from an older, non-consecutive snapshot. Past merged progenitors are tracked by one particle, their "galaxy particle". All particles of the descendant under investigation are checked for being a galaxy particle of a past merged progenitor. The most strongly bound galaxy particle will be considered the main progenitor of the descendant under consideration. If a match is found, the past merged progenitor is removed from the list of past merged progenitors.
  - Descendants that still haven't found a progenitor at this point are deemed to be newly formed.
- The results are written to file, and the code goes on to the previously described step 2.

# Appendix B Testing Parameters of the Merger Tree Algorithm: Results Distinguishing Between Haloes and Subhaloes

In this section, the results for the evaluations of the merger tree algorithm parameters, as described in section 3.2 are given, including the distinction between subhaloes and haloes.

Displacement statistics (eq. 24) are shown in figure 17, logarithmic mass growths (eq. 26) and mass growth fluctuations (eq. 27) are shown in figure 18 for the `saddle/no saddle` comparison and in figure 19 for varying number of tracer particles $n_{mb}$.

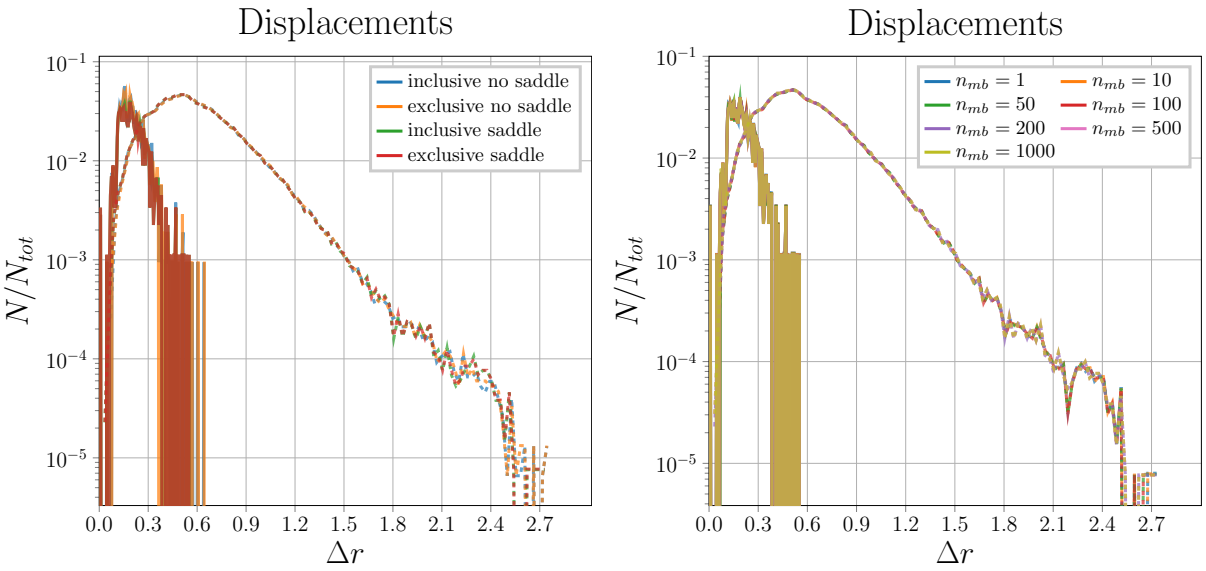


**Figure 17:** Distribution of the displacements for progenitor - descendant pairs with masses above $5 \cdot 10^{11} M_{\odot}$ throughout the entire simulation for the halo catalogue influencing parameters: whether subhalo particles are included (`inclusive`) or excluded (`exclusive`) in the clump mass of satellite haloes, and whether to consider particles which might wander off into another clump as bound (`no saddle`) or not (`saddle`) are shown in the left plot. The results for varying numbers of clump tracer particles, $n_{mb}$, are shown in the right plot. Solid lines are for haloes, dashed lines are for subhaloes. The distributions are computed as a histogram which is normalised by the total number of events found.

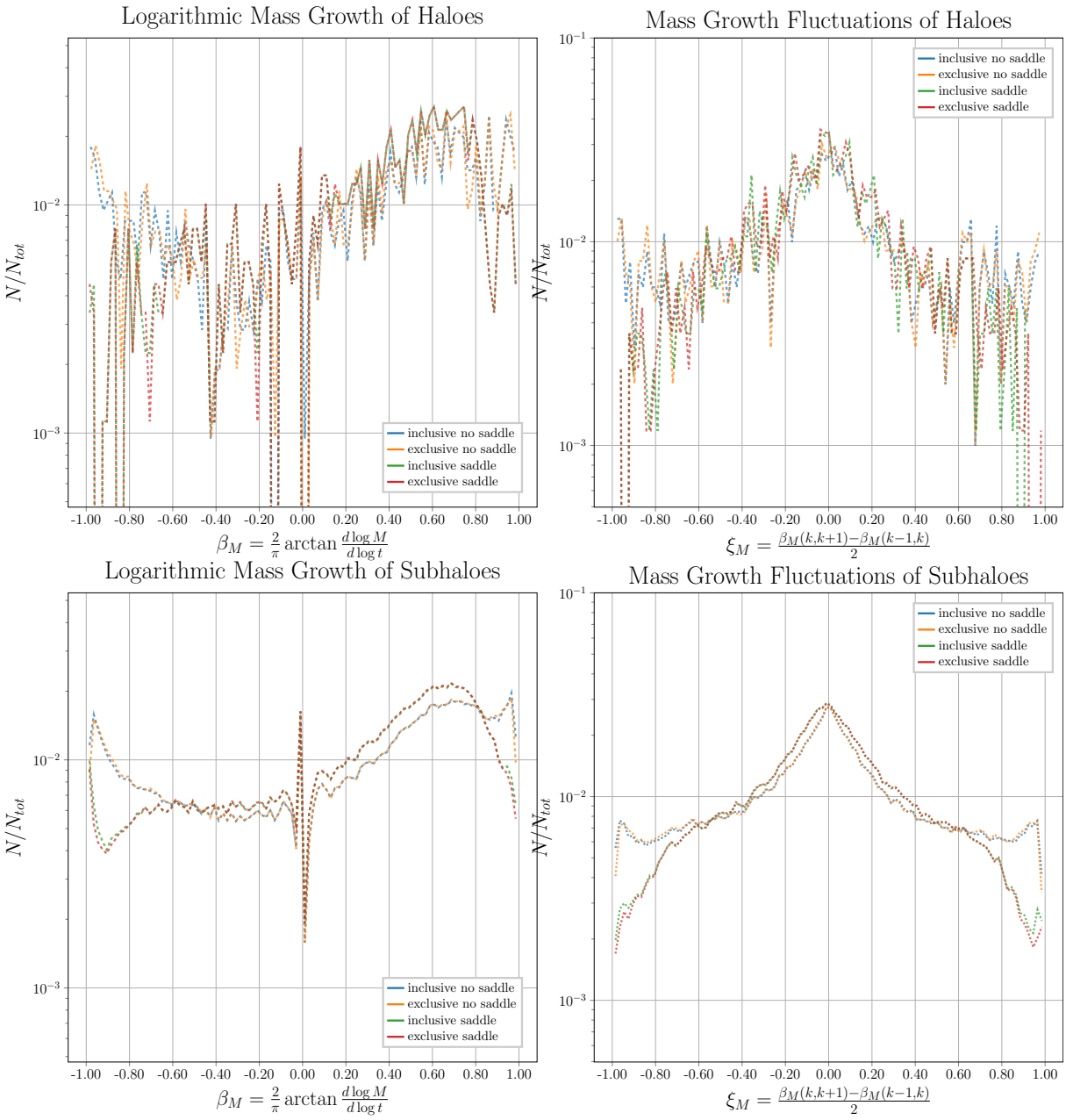


**Figure 18:** Logarithmic mass growth and mass growth fluctuation distributions for progenitor - descendant pairs or three consecutive nodes in a branch, respectively, with masses above $5 \cdot 10^{11} M_{\odot}$ throughout the entire simulation for the halo catalogue influencing parameters: whether subhalo particles are included (`inclusive`) or excluded (`exclusive`) in the clump mass of satellite haloes, and whether to consider particles which might wander off into another clump as bound (`no saddle`) or not (`saddle`). The distribution is computed as a histogram which is normalised by the total number of events found. The upper row contains the results for haloes, the lower for subhaloes.

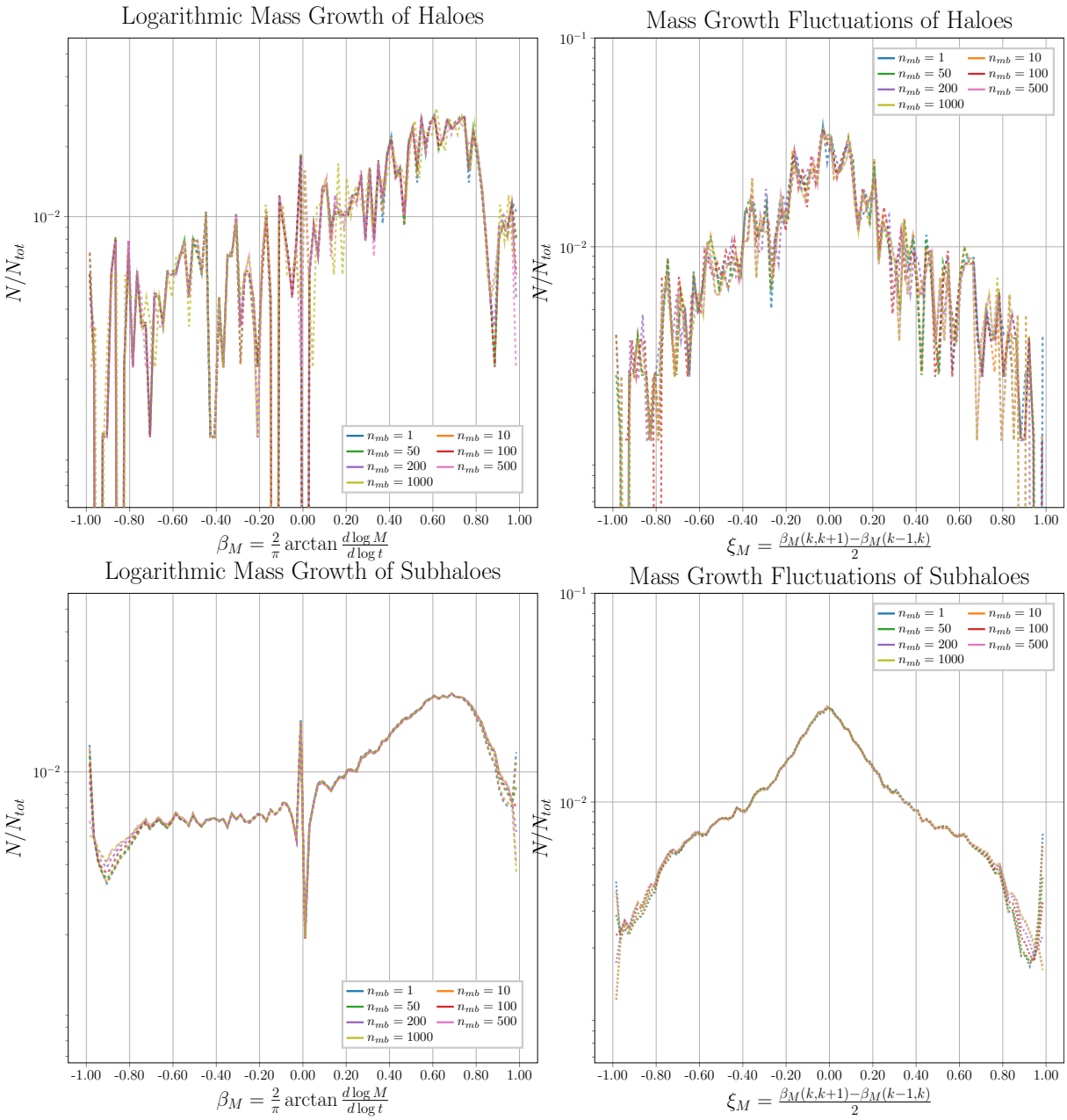


**Figure 19:** Logarithmic mass growth and mass growth fluctuation distributions for progenitor - descendant pairs or three consecutive nodes in a branch, respectively, with masses above $5 \cdot 10^{11} M_{\odot}$ throughout the entire simulation for varying numbers of clump tracer particles $n_{mb}$. The upper row contains the results for haloes, the lower for subhaloes.

# Appendix C Stellar Mass Functions at Mean Redshift vs Average Over Redshift Interval

Figure 20 shows the obtained stellar mass functions $\Phi(M_*)$ over redshift intervals of the `G69` simulation for both the snapshot with redshift closest to the central redshift of the interval used for observational data and the average SMF computed over all snapshots with redshift within the redshift interval. The differences are negligible.

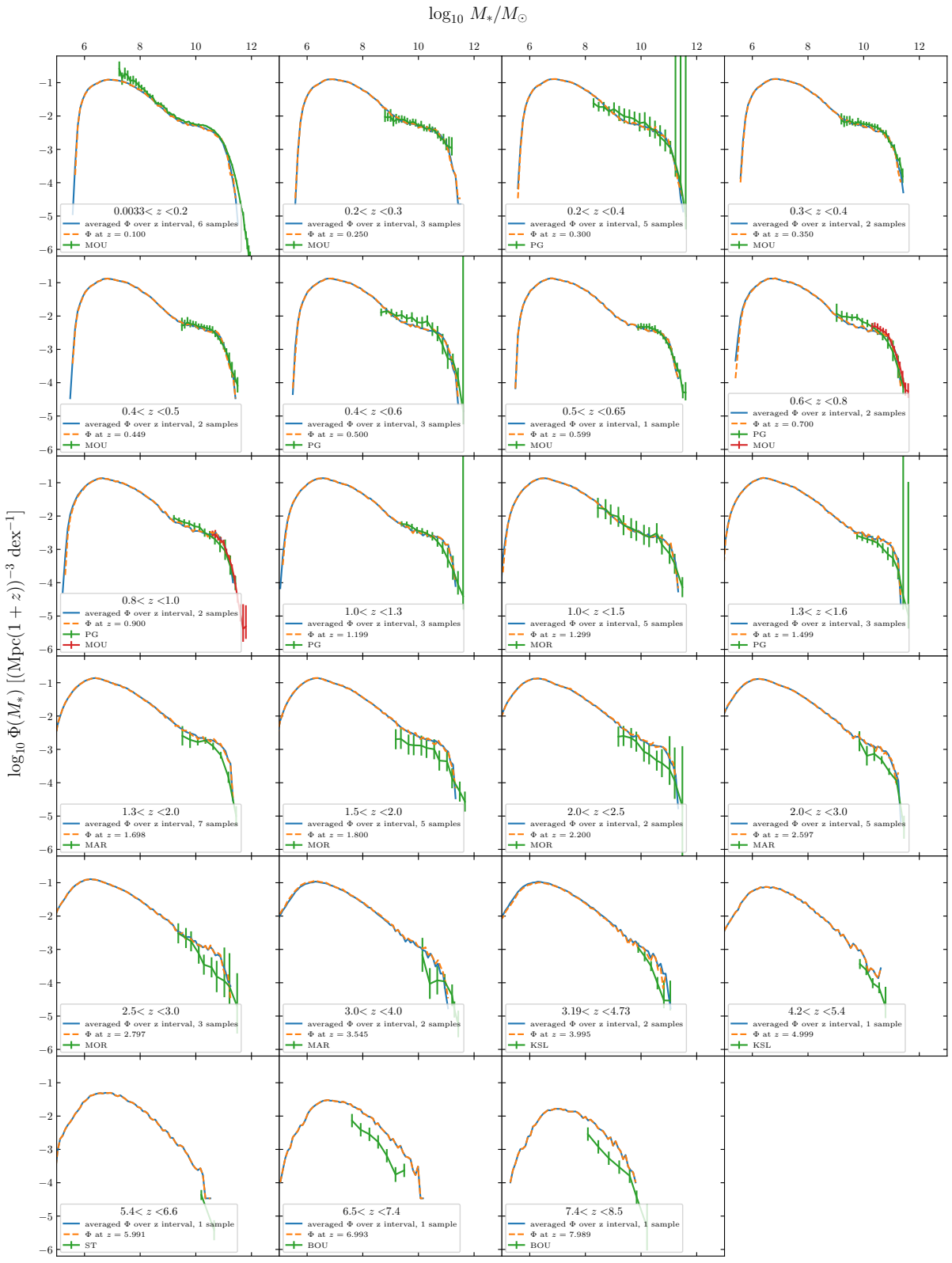


**Figure 20:** Obtained stellar mass functions $\Phi(M_*)$ of central galaxies for the G69 simulation datasets described in 4.2 both averaged over the redshift interval of the observational data and from a single snapshot closest to the centre of the interval compared to observed stellar mass functions. The abbreviations used for observational data are listed in table 4.