Università degli Studi di Padova

Facoltà di Scienze MM.FF.NN.

Dipartimento di Astronomia

Halo-matter cross-correlation in cosmological simulations

Relatore: Prof. Giuseppe Tormen
Correlatore: Prof. Ravi Sheth

Laureando: Brunetto Marco Ziosi

Anno Accademico 2010-2011

"Try again. Fail again. Fail better."
(Samuel Beckett)

"Stay hungry, stay foolish."
(Steve Jobs)

Contents

Contents **1**

Introduction **3**

1 Structure Formation **5**
1.1 Cosmological Principle 5
1.2 Friedmann's universes 7
1.3 Content of the universe 9
1.4 Structure formation 10

2 The Two Point Correlation Function **15**
2.1 Correlation estimators 16
2.2 Cross-correlation implementation 19
2.3 Cross-correlation and power spectrum 20

3 Halo model **23**
3.1 Halo structures and galaxies 24
3.2 Non-linear power spectrum 25
3.3 Cross spectra 27

4 Simulations **31**
4.1 The GIF and GIF2 Simulations 33
4.2 The Millennium Simulation 35
4.3 The Millennium II Simulation 37
4.4 Gadget and the database 38

5 Trees **41**

5.1 Definition of KD-Tree 43
5.2 The search algorithm 44
5.3 KD-Tdree optimization 47

6 Developing a Python correlation code 51
6.1 Python 52
6.2 Hardware 54
6.3 Developing the code 56

7 Results 65
7.1 GIF galaxy-galaxy auto-correlation 65
7.2 GIF2 matter-matter auto-correlation 66
7.3 GIF2 halo-matter cross-correlation 68
7.4 GIF2 halo-matter CC in mass bin 70
7.5 GIF2 subhalo-matter cross-correlation 73
7.6 Bias 76

8 Conclusions 81
8.1 Further works 82

A Better bugs 83
A.1 array2int 83
A.2 `.sort` instead of `.sort()` 84
A.3 Random population 84

Bibliography 85

INTRODUCTION

Gravitational lensing is a quite new and very powerful way of investigation of the large scale distribution of matter in the universe. Distant galaxies sometimes appear distorted, in a correlated way, because the foreground mass change the trajectory of the light reaching us. These distortions can be useful to map the mass distribution on large scales (Hayashi and White 2008). One of the methods to make use of this is to compute the cross-correlation between the cosmological structures and sub-structures with themselves or with the mass, and compare the resulting structure knowledge with the lensing observations.

The goal of this work is to compute the correlation function between the dark matter (DM) sub-haloes and the DM mass particles in the coordinate space. This is useful because it permits to check the theoretical prevision (Giocoli et al. 2010) of these quantities. Moreover the cross-correlation permits to compare different models of aggregation for the matter and gives a statistical tool to characterize a distribution of matter or the structures formed.

In this work the cross-correlation will be calculated on the result of some simulation: the GIF and GIF2, the Millennium Simulation, the Millimillennium Simulation and Millennium II Simulation. There are two ways to calculate the cross-correlation, one in the Fourier space, the other in the configuration space. In the Fourier space one has to create a grid over the particles distribution and then calculate the power-spectrum (cross-spectrum), from which he can obtain the CC. This method is subject to *shot-noise*. In case of a *sparse* distribution of matter in a big space (box) the shot-noise could be big enough to make the results useless. This happens especially in the case that we want to investigate small scales: we would need a fine grid, which dimension make its representation in memory difficult, almost empty because of the small number of particles to check. Moreover to obtain the correlation from the spectrum it is necessary to

deconvolve and anti-transform it and this operation lead to a loss of information, especially on the small scales. The second way, that we will follow is to calculate the correlation in the configuration space counting the pairs at each distance.

Some considerations and tests lead us to choose Python as programming language and a binary tree as data structure. Some optimization was done over the original library, in particular for what regards the inclusion or exclusion of nodes with some characteristics, based on the cache statistic and in case of auto-correlation. In addition the slow distance calculation was substituted with a faster function in Fortran imported as a shared library with *f2py*.

Some tests was made both for the speed of the code and for the correctness of the results and some tuning was done to understand the best parameters (leafsize, strategy, ...) to be used.

This dissertation is structured as follow: Chapter 1 introduce the standard model of cosmological structure formation. In Chapter 2 we will treat the two point correlation function, why it is important, how to calculate it and its connection with the power spectrum. The following chapters are dedicated to the presentation of the *halo model* (Chapter 3) and to the cosmological simulations (Chapter 4). In chapter 5 we will introduce and explain what *kd*-tree are and how they were developed. Chapter 6 treats the development of the main code used in this work. Chapter 7 is dedicated the results of our work. In Chapter 8 we presents some conclusions and propose further works.

CHAPTER 1

THE STANDARD MODEL OF STRUCTURE FORMATION

A little introduction on why the universe is as we see it.

1.1 Cosmological Principle

As of today, *Standard Cosmology* is based on the so called *Cosmological Principle*. This principle was introduced in the 1920's by Einstein, while he was developing his General Theory of Relativity, and before any astronomical observation could either confirm or refute it. The Cosmological Principle states that, on scales large enough, the universe is spatially homogeneous and isotropic, i.e. it has the maximum number of symmetries[1]. This assumption translates into powerful constraints on the viable cosmological models. In this way we can develop simplified models of the Universe starting from Einstein's equations, otherwise too difficult to be solved for an arbitrary matter distribution. The hypothesis of a homogeneous and isotropic Universe was confirmed by Penzias and Wilson in 1965 with the observation of the Cosmic Microwave Background (CMB), the

[1] In a more formal way, the Cosmological Principle states that the space-time has a maximally symmetric tridimensional sub-space.

remnants of the hot and dense past of the Universe. Its spatial features convinced the astronomers that the Cosmological Principle was valid.

To build a Universe model upon this idea and with tools offered by General Relativity we have now to consider the geometrical properties of the Universe as a fluid. The geometrical properties of a space are described by its metric, and the more general one for an homogeneous and isotropic space is the Robertson-Walker metric:

$$\mathrm{d}s^2 = (c\,\mathrm{d}t)^2 - a^2(t)\left[\frac{\mathrm{d}r^2}{1-kr^2} + r^2\left(\mathrm{d}\theta^2 + \sin^2\theta\,\mathrm{d}\varphi^2\right)\right] \tag{1.1}$$

where r is the comoving coordinate and $a(t)$ is the *scale factor*. Einstein's equations

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R = \frac{8\pi G}{c^4}T_{\mu\nu} \tag{1.2}$$

with the energy-momentum tensor

$$T_{\mu\nu} = \left(p + \rho c^2\right)U_\mu U_\nu - p g_{\mu\nu} \tag{1.3}$$

link the geometric properties of space-time with the energy momentum tensor that describes the content of the universe.

In a space described by the Robertson-Walker metric, with mass-energy density at rest given by ρc^2 and pressure p the equations (1.2) reduce to the Friedmann's equations:

$$\ddot{a} = -\frac{4\pi G}{3}\left(\rho + 3\frac{p}{c^2}\right) \tag{1.4}$$

$$\dot{a}^2 + kc^2 = \frac{8\pi G}{3}\rho a^2 \tag{1.5}$$

where $a = a(t)$ is the scale factor and t is the proper time. From Friedmann's equations we can extract the *spatial curvature*

$$k = \left(\frac{\dot{a}}{c}\right)^2 [\Omega(t) - 1] \tag{1.6}$$

with the *critical density* given by

$$\rho_c = \frac{3}{8\pi G}\left(\frac{\dot{a}}{a}\right)^2 \tag{1.7}$$

and *density parameter*

$$\Omega(t) = \frac{\rho}{\rho_c}. \tag{1.8}$$

From the same equations we can obtain the proper distance between two points from which we derive the *Hubble parameter*

$$H(t) = \frac{\dot{a}}{a} \tag{1.9}$$

and, expanding the scale factor near t_0, the *deceleration parameter*

$$q_0 = -\frac{\ddot{a}_0 a_0}{\dot{a}_0^2}. \tag{1.10}$$

Einstein soon understood that the equations were not consistent with his idea: a static universe can not be described only by an uniform distribution of matter unless it has negative pressure. The solution was to add to the equations a term called *cosmological constant* in a way that it did not change the covariant character of the equations. The consequences on the Friedmann equations were only the substitution of pressure and density with equivalent effective quantities:

$$\tilde{p} = p - \frac{\Lambda c^4}{8\pi G} \tag{1.11}$$

$$\tilde{\rho} = \rho + \frac{\Lambda c^2}{8\pi G}. \tag{1.12}$$

When the expansion of the universe became evident, this term was abandoned. However it has come back during the last 15 years, to describe the observed accelerated expansion of the Universe.

1.2 Friedmann's universes

Starting from the Friedmann equations and taking into account an adiabatic expansion given by

$$\mathrm{d}\left(\rho a^3\right) = -3\frac{p}{c^2}a^2\,\mathrm{d}a \tag{1.13}$$

we can build simple models for the observed universe. These are rough but accurate enough until the mean free path is much smaller than the physical considered scales. The equation of state can be written as

$$p = w\rho c^2 \tag{1.14}$$

where the parameter w is a constant with values $0 < w < 1$ (called Zed'dovich interval): $w = 0$ represents a fluid with zero pressure and approximates quite

well any non-relativistic fluid or gas because $m_p c^2 \gg k_B T$ so the pressure is relativistically non relevant. This kind of matter is conventionally called dust. More in detail, a non relativistic fluid has pressure $p = w(T)\rho c^2$ with $w \to 0$, a non-degenerate relativistic fluid in thermal equilibrium has equation of state given by

$$p = \frac{1}{3}\rho c^2 \tag{1.15}$$

For simplicity we will assume w constant with time.

Putting together the equation of state (1.14) and the equation describing the adiabatic expansion (1.13) we have

$$\rho a^{3(1+w)} = \rho_0 a_0^{3(1+w)} = const \tag{1.16}$$

from which we can obtain the relation between density and expansion for matter and radiation. The energy density for the radiation decreases faster (the exponent is 4 instead of 3) because one must consider the volume variation together with the drop of energy caused by redshift. A fluid characterized by $w = -1$ is equivalent to a Cosmological Constant, whose energy does not decrease with the expansion of the universe.

Solutions and Friedmann models

Depending on the value we choose for w and Ω we obtain different analytic solution for the Friedmann equations:

- $\Omega = 1$ gives spatially flat universes, in particular if $w = 0$ the solution is called Einstein-de Sitter universe and the expansion proceeds with constant deceleration, reaching infinity at zero speed.
- If $\Omega < 1$ we have the open universe for which the expansion proceeds decelerates less and less, and is asymptotically linear with time.
- In the case of $\Omega > 1$ the universe reaches a maximum size after which it collapses
- Model in which the matter (dust) dominates are characterized by an equation of state with $w = 0$ and are flat, open or closed depending on the value

of Ω: more, less or equal to 1. If $\Omega > 1$ the solution has the parametric form of a cycloid.

- Universes dominated by the radiation have $w = \frac{1}{3}$ and are flat, open or closed depending on the value of Ω.

1.3 Content of the universe

Now we consider the principal material components of the standard model for the cosmic structures formation as they are presented in Tormen 2011.

Cold Dark Matter (CDM)

Most of the matter in the universe is non-baryonic "cold dark matter", a collisionless kind of matter that are not yet clearly identificated. It is "dark" because it doesn't absorb, emit o scatter any type of radiation but we can observe it only through gravity. It is "cold" because we know that the particles that made it are not relativistic at any interesting epoch. It is non baryonic because it does not belong to the set of known particles that build up the ordinary matter but it is provided by supersimmetric theories.

There are at least two issues that clearly show the need of the CDM: the formation and the dynamic of galaxies. Without the DM both the cosmic structures as we can see them now and the observed velocity profiles of galaxies are not justifiable.

In the last years the most probable candidate seems to be the *neutralino*, a particle of mass about 100 GeV. In extreme dense conditions it could annihilate emitting gamma radiation.

The first cause of the introduction of the DM in the cosmological models was to reconcile the low value of $\Omega_b < 0.09$ with the dynamic one $\Omega_m \sim 0.3$ obtained from the mass estimation of spiral galaxies and galaxy clusters. The high rotational velocities of the spiral galaxies and orbital velocities of galaxies in the clusters show that the mass is much more than the contribution from the visible, baryonic, mass $\Omega \sim 0.3$. The solution was to introduce a non baryonic dark matter, gravitationally dominant.

Baryonic Matter

The baryonic matter, also called ordinary matter, are made by the well know particles such as protons, electrons, ... Its abundance agree with the nucleosynthesis provided by the Big Bang model, that is $\Omega_b = 0.044 \pm 0.003$. A little part of this constitutes the stars and galaxies we can see, the rest is in the form of diffuse interstellar, intergalactic and intracluster medium. If we consider an universe with different components the Ω in the previous equation became $\Omega_{tot} = \Omega_\Lambda + \Omega_{DM} + \Omega_b + \Omega_\gamma$ and whenever a component dominated the density that component determined also the expansion.

1.4 Structure formation

As we mentioned in a previous section there is no way to have the structure we can observe now in the universe using only the baryonic matter. If we define the density fluctuation as

$$\delta(\mathbf{x}, t) \equiv \frac{\rho(\mathbf{x}, t) - \rho_{bg}(t)}{\rho_{bg}(t)} \equiv \frac{\delta\rho(\mathbf{x}, t)}{\rho_{bg}(t)} - 1 \tag{1.17}$$

at recombination[2] $\delta(a = a_{rec}) \sim 10^{-5} \ll 1$, proportional to the temperature fluctuation.

In a universe composed only by baryonic matter ($\Omega = 1$) we would have $\delta(t) \propto a \propto t^{2/3}$, therefore

$$\delta a_0 = \delta(a_{rec}) \frac{a_0}{a_{rec}} \sim 10^3 \delta a_{rec} \sim 10^{-2} \tag{1.18}$$

contrary to what we observe. The existence of structures shows that fluctuations are definitely non-linear, that is $\delta(a_0) \gg 1$. Because of $\delta \propto a$, in order to have $\delta(a_0) \gg 1$ it is necessary that $\delta(a_{rec}) \geq 10^{-3}$ but $\frac{\delta T}{T}(a_{rec}) \sim 10^{-3}$ is not observed. What we need is a component whose perturbations can start to grow before recombination, in order to reach non-linearity by the present time.

[2]We call recombination the period in the history of the universe when, due to expansion, the temperature became low enough to let the electrons and the nuclei of the ionized matter recombine. We also say that from there the universe become transparent because the light can travel without being scattered by the free electrons.

Cosmic structures formation

In the previous section we have seen that the DM leads the formation of the cosmic structures. This happens because the growth of the first gravitational instabilities, due to the quantum fluctuations, permits the collapse of regions of DM and the formation of the first systems in gravitational equilibrium, the *DM haloes* in a hierarchic way: first the smaller haloes collapse, then the aggregation of those forms the bigger systems. This model is called *Hierarchical Clustering*.

The baryonic matter then follows the gravitational well of the DM: since the gas is collisional it warms up for the adiabatic compressions and its kinetic energy is converted into thermal energy by the shocks. The temperature increase until that of the virial equilibrium and then diminishes due to the radiative losses. The cooling let the gas condensate and begin to form the first molecular clouds and then stars. Those start evolve processing the elements and producing the heavier ones, possibly exploding as supernovae releasing in the interstellar medium the new elements. The energy provided by the explosion heats the gas giving rise to new stars that will eventually provide new materials. At the same time the stars are affected by the gravitational potential and start to cluster in galaxies.

Jeans instability

The gravitational instability can be better qualitatively understand with the Jeans criterion. We can think about a uniform distribution of fluid (in this case, dark matter) with little density fluctuations on all the scales. Let us now consider a region where the dark matter is overdense compared to the average density, and suppose that it is spherically symmetric, with radius R, mean density ρ, mass $M \propto \rho R^3$ and with typical particles speed v. Now, two processes compete in determining the evolution of the system: the gravity among the particles tends to collapse the region let it to became smaller and denser, the particles motion diffuse the particles decreasing the overdensity.

We can see this balance in terms of energies. We consider the kinetic and

potential energies:

$$E_k \sim \frac{1}{2}Mv^2 \tag{1.19}$$

$$E_p \sim -\frac{GM^2}{R} \sim -\frac{GM}{R}\rho R^3 \sim -GM\rho R^2. \tag{1.20}$$

If the system has zero total energy, i.e. the two energies equals, we obtain

$$\frac{v^2}{2} = G\rho R^2 \tag{1.21}$$

$$R = R_J = v\sqrt{\frac{1}{2G\rho}} \tag{1.22}$$

where R_j is the *Jeans Radius*. For $R > R_j$ the gravity overcome the kinetic energy and the perturbation collapse, for $R < R_j$ the kinetic energy diffuse the particles and the perturbation disappears.

We can obtain the same results considering the force or the characteristic times.

Linear and non-linear evolution

Let us now consider those perturbations with $R > R_J$. Until $\delta \ll 1$ the growth of these perturbations is linear, so it is possible to construct a set of equations to follow the evolution analytically. A simpler, qualitative, analysis can be done by solving the Friedmann's equations on scales $R \gg R_J$. The perturbation is now a sphere of radius R and density $\rho_1 > \rho_c$ inside a universe with $\Omega \equiv 1$, that is $\rho_{bg} = \rho_c$. The sphere is equivalent to a universe with $\Omega > 1$ surrounded by a universe with $\Omega = 1$ so we can consider the Friedmann's equation

$$H^2(t) = \frac{\dot{a}^2(t)}{a^2(t)} = \frac{8\pi G}{3}\rho_{bg}(t) - \frac{kc^2}{a^2(t)} \tag{1.23}$$

in the two cases, obtaining

$$H_{bg}^2 = \frac{8\pi G}{3}\rho_{bg} \tag{1.24}$$

$$H_1^2 = \frac{8\pi G}{3}\rho_1 - \frac{c^2}{a^2}. \tag{1.25}$$

We can examine the perturbation when the two Hubble constants have the same values without loss of information, obtaining

$$\delta t \propto \frac{1}{a^2(t)\,\rho_{bg}(t)} \ll 1. \tag{1.26}$$

Because in an Einstein-de Sitter's universe with equation of state $p = w\rho c^2$ we have $\rho(t) \propto a^{-3(1+w)}$ we obtain $\delta(t) \propto a^{1+3w}$. This implies that when $a < a_{eq}$ the radiation dominates, so $\delta(t) \propto t$ and when $a > a_{eq}$ DM dominates and $\delta(t) = t^{2/3}$.

A more rigorous treatment (e.g. Cooray and Sheth 2002) combines the continuity equation

$$\frac{\partial\delta}{\partial t} + \frac{1}{a}\nabla\cdot(1+\delta)\,\mathbf{u} = 0 \tag{1.27}$$

and the Euler equation

$$\frac{\partial\delta}{\partial t} + H\mathbf{u} + \frac{1}{a}\left[(\mathbf{u}\cdot\nabla)\,\mathbf{u} + \nabla\phi\right] = 0 \tag{1.28}$$

where the potential fluctuations due to density perturbations are given by the the Poisson equations

$$\nabla^2\phi = 4\pi G\bar{\rho}a^2\delta \tag{1.29}$$

and peculiar velocity

$$\mathbf{u} = \mathbf{v} - H\mathbf{x}. \tag{1.30}$$

In the linear regime $\delta \ll 1$ the equations can be combined in a second-order differential equation

$$\frac{\partial^2\delta}{\partial t^2} + 2H\frac{\partial\delta}{\partial t} - 4\pi G\bar{\rho}\delta = 0 \tag{1.31}$$

with two independent solutions, corresponding to the grow and decay modes respect to the time. The growing solution has the form

$$\delta(k, r) = G(t)\,\delta(k, 0) \tag{1.32}$$

where

$$G(r) \propto \frac{H(r)}{H_0}\int_{z(r)}^{\infty} \mathrm{d}z'\,(1+z')\left[\frac{H_0}{H(z')}\right]^3 \tag{1.33}$$

$$\sim \frac{5}{2}\frac{\Omega_m(z)/(1+z)}{\Omega_m(z)^{4/7} - \Omega_\Lambda(z) + (1-\Omega_m(z)/2)\,(1+\Omega_\Lambda(z)/70)}. \tag{1.34}$$

For $\Omega_m \to 1$ we have $G \propto a = (1+z)^{-1}$.

When the amplitude δ of the fluctuation reaches $\delta = 1$ the non-linear effects become important and can not be neglected. Here the region corresponding to the perturbation separates from the expansion of the universe and begins to collapse to a virialized halo and we can not still describe the fluctuation as

$$\delta(\mathbf{x}, t) = \delta_{(x,t_0)} \frac{D(t)}{D(t_0)} \tag{1.35}$$

where $D(t)$ is the linear growth factor.

At the beginning the two main approaches were the *Violent Relaxation* and the *Secondary Infall*. The first predict a mixing occurring in the phase space with a chaotic result respect to the initial conditions while the second a gentle shell collapse around a single perturbation. Both of them can be applied only to a single structure and do not permit to find, for example, a mass function. Press and Schechter provided a method to maintain part of the linear theory leading to a simpler treatment. Their technique was then corrected and reformulated in the *excursion sets* formalism (Bond et al. 1991). In the linear regime the collapse occur when $\delta \sim 1$ but the true value is $\delta_{NL,true}$. Nevertheless existence of a one-to-one relation between the linear and the non linear trends permits to have a linear value for $\delta_c \sim 1.686$ that correspond to the non-linear value at the collapse. This means that when the non- linear halo collapse the linear theory would present $\delta_{lin} = \delta_c$. The next step is to change the temporal dependence from the fluctuations to the critical value of δ.

CHAPTER 2

THE TWO POINT CORRELATION FUNCTION

Here we present the two point correlation function, the motivation on why to use it and the best way to estimate it.

The modern technologies permits huge observations (big surveys, deep observations, ...) as well as numerical simulations greater than ever before and so we have now a lot of data with the positions of the point representing DM particles, galaxies or halo centers or whatever is needed. What we need is a way to characterize this distribution in order to know the interesting properties of it and to compare this distribution with another, or with observative data or with analytic models. If we are interested in the clustering properties of our data, one of the most useful and most widely used tools is the *spatial correlation function*, in this case, the *two-point correlation function - TPCF* $\xi(r)$. In short, it measures the probability of finding a pair of objects (mass points, halos, sub-halos, galaxies) separated by a certain distance r. In a discrete case, here we have a discrete distribution of points, we have on average n points per volume unit. Following Peebles 1980 the probability to find one point in the infinitesimal volume $\mathrm{d}V$ is

$$\mathrm{d}P = n\,\mathrm{d}V. \tag{2.1}$$

If dV doubles, dP doubles too.

The mean number of points in a finite volume V is

$$N = nV. \tag{2.2}$$

The two point correlation function of this distribution is defined by the joint probability of find two point with separation r:

$$\mathrm{d}^2P = n^2\,\mathrm{d}V_1\,\mathrm{d}V_2\left[1 + \xi(r)\right]. \tag{2.3}$$

Because we are dealing with an homogeneous and isotropic universe, ξ depends only upon r. The probability is proportional to $\mathrm{d}V_1$ and $\mathrm{d}V_2$ because if one of these doubles, it doubles also the probability. In case of Poissonian process on a uniform random distribution the two probabilities are independent, so

$$\mathrm{d}^2P = n^2\,\mathrm{d}V_1\,\mathrm{d}V_2 \tag{2.4}$$

that is $\xi \equiv 0$. Then we can see that $\xi(r)$ represents the excess (or defect) of clustering of a distribution compared to a uniform Poissonian one. If $\xi(r) > 0$ we have a positive correlation, in case of $-1 < \xi(r) < 0$ we have anti-correlation. In any case it must be $\xi(r) > -1$.

2.1 Correlation estimators

While the role of the TPCF is central, estimators for extracting it from a set of spatial points are confusingly abundant in literature. The reason is partly due to the lack of a clear criterion to distinguish between the estimators. As described in Szapudi and Szalay 1998 the simple estimator

$$\xi(r) = \frac{DD}{RR} - 1 \tag{2.5}$$

was widely used, where DD denotes the number of pairs at a given range of separations and RR the number of random pairs, with the same separation, generated over a similar area as the data. In Landy and Szalay 1993 a new estimator is proposed:

$$\xi_{LS}(r) = \frac{DD - 2DR + RR}{RR} \tag{2.6}$$

In this case, in the limit of weak clustering, the variance is proportional to $1/n^2$, i.e. Poissonian in the pair counts whereas for all the other estimators the leading-order term is $1/n$ with n the number of points in the survey.

A lot of other estimators are also available, and one could have smaller variance under certain circumstances, but it could have a bigger bias. Therefore, *before* doing any numerical experiment it is a good thing to have an idea of which estimator better suits our needs. The cumulative probability distribution of the measured value lying within a certain tolerance of the *true* value is going beyond the concept of bias or variance, and it even takes into account any non-Gaussian behavior of the statistics. This is the mathematical formulation of the simple idea that an estimator that is more likely to give values closer to the truth is better. Kerscher, Szapudi, and Szalay 2000 collects the different forms of estimators and perform numerical experiments in several sub-samples of a large simulation, determining the cumulative probability of measuring values close to the true one and thereby ranking the different estimators.

The indicators were extracted from over 500 sub-samples of the Virgo Hubble volume simulation cluster catalog. The *real* correlation function was determined from the full survey in a 3000 Mpc/h periodic cube. The estimators were ranked by the cumulative probability of returning a value within a certain tolerance of the real correlation function. This criterion takes into account bias and variance and it is independent of the possibly non-Gaussian nature of the error statistics. As a result, for astrophysical application, a clear recommendation has emerged: the Landy&Szalay estimator is preferred in comparison with the other indicators examined, with performance almost indistinguishable from the Hamilton estimator.

Following Szapudi and Szalay 1998 they define the pair counts with a function Φ:

$$P_{DR}(r) = \sum_{x \text{ in } D} \sum_{y \text{ in } R} \Phi_r(x, y) \,. \tag{2.7}$$

The summation runs over coordinates of points in data set D and in the set R of randomly distributed points. They also define the TPCF as $\Phi_r(x, y) = [r \leq d(x, y) \leq r + \Delta]$ where $d(x, y)$ is the separation between two points and the condition in brackets equals 1 when the condition holds and 0 otherwise. P_{DD} and P_{RR} are defined analogously, with x and y taken entirely from the data and

random samples, under the restriction that $x \neq y$. They also introduce the normalized counts $DD(r) = P_{DD}(r) / [N(N-1)]$, $DR(r) = P_{DR}(r) / (NN_R)$ and $RR(r) = P_{RR}(r) / [N_R[N_R - 1]]$ with N and N_R being the total number of data and random points in the survey volume.

The estimators considered are:

$$\hat{\xi}_n = \frac{DD}{RR} - 1 \tag{2.8}$$

$$\hat{\xi}_{DP} = \frac{DD}{DR} - 1 \tag{2.9}$$

$$\hat{\xi}_{Hew} = \frac{DD - DR}{RR} \tag{2.10}$$

$$\hat{\xi}_{Ham} = \frac{DD\ RR}{DR^2} - 1 \tag{2.11}$$

$$\hat{\xi}_{LS} = \frac{DD - 2DR + RR}{rr} \tag{2.12}$$

Respectively the natural estimator and the estimators by David&Peebles, Hewitt, Hamilton and
Landy&Szalay (Landy and Szalay 1993).

To compare the estimators for typical cosmological simulations they used the cluster catalogue generated from the ΛCDM Hubble volume simulation (Colberg et al. 1998). In order to investigate the effects of shape, clustering and amount of random data used always one parameter at a time was varied. The result are that on small scales all the estimators are comparable, but on large scales the LS estimator and the Hamilton estimator significantly outperform the rest, showing the smallest deviations for a given cumulative probability. While the two estimators yield almost identical results for an infinite number of random points, the Hamilton estimator is considerably more sensitive to the number of random points employed than the LS version is. From a practical point of view the LS estimator is thus preferable.

So, the LS estimator emerges as the one they clearly recommend. These considerations apply to volume-limited samples. When the correlation function is estimated directly from a flux-limited sample the Hamilton estimator has the advantage of being independent of the normalization of the selection function. Szapudi and Szalay 1998 demonstrate that the LS estimator has superior shot noise behaviour compared with the existing alternatives. Labatie et al. 2010 show that the LS estimator is nearly of minimal variance for a random distribu-

tion (i.e. Poisson with no correlation) and that this estimator has second order variance decay in $1/n^2$ with n being the number of data points. They also compared the estimators regarding their sensibility to the fluctuation in the number of galaxies n (i.e. the uncertainty in the mean density): Peebles-Hauser and David-Peebles depend on the first order on that fluctuation whereas Hamilton and LS have a second order dependence. As a consequence the variances of the first two estimators have only a first order decay in the volume size whereas the two latter estimators have a second order decay. They confirmed, with the simulations that the Hamilton and the LS have much smaller variances. Thus we chose to use the LS estimator to compute the TPCF in this work.

2.2 Cross-correlation implementation

There are essentially two ways to compute the TPCF: the first is in the Fourier space, the second in the coordinate space. The first is achieved by gridding the space and calculating the power spectrum. Its Fourier transform is the TPCF. This way if affected by the shot noise and to minimized it the grid has to be as fine as possible. To investigate the clustering on small scales in a big simulation box the grid would be very big but also almost empty except in some, very dense, areas. This is computationally inefficient and very difficult to handle. This is why we choose to do this in the coordinate space.

In the coordinate space the TPCF is calculate through an estimator counting the number of pairs at each distance bin. The trivial operation is simple but computationally too expensive, so it is necessary to find an appropriate algorithm to do this in a reasonable way.

In this work we choose to use Python and in particular the modules of the *Scipy* project as it will be explained after. Scipy provides a lot of optimized scientific Python modules, mostly written in C. Among them there is the `KDTree` module which provide a base for the algorithm used in this work. The original modules, thanks to its "open-sourceness", was modified to fit better the need of this work, in particular implementing some of the suggestion from the dedicate articles. We added to the original model the possibility to prune the nodes tat have been yet checked (for example, if we are doing self-correlation, the pair 1-3 is equal to 3-1) or the equal nodes. This was done assigning a tag to each

node and touching only the node couples with `node1.tag>node2.tag`. In case of self.correlation if the nodes are the same the result is halved. We also tested the possibility to change strategy for the counting; tests were made to decide among six different strategies. We profiled the *kd*-tree algorithm and the entire code and decide to change the way the distance was calculated, from Python to Fortran, including the module with `f2py`. Other tests were made to find the best `leafsize`, i.e. the best number of objects per leafnode, the best way to sort the distance result when opening two leaves, if the traverse was sufficiently efficient and if there were other functions to optimize.

Correlation error

The following expression

$$\sigma_\xi(r) = (1 + \xi(r))/\sqrt{DD\,(r)} \tag{2.13}$$

is the Poisson expression for the error. If we consider $DD(r)$ which is the number of data pairs on scale r , for Poisson statistics its uncertainty is $\sqrt{DD}$. The same would be true for RR, the random pairs. So we can write

$$\frac{DD \pm \sqrt{DD}}{RR \pm \sqrt{RR}} = \frac{DD}{RR}\frac{\left(1 \pm \sqrt{DD}\right)}{\left(1 \pm / \sqrt{RR}\right)} \tag{2.14}$$

and if you used many more randoms than data, then $1/\sqrt{RR} \ll 1/\sqrt{DD}$ so this becomes $(DD/RR)(1 \pm 1/\sqrt{DD})$. This means that the error on $1 + \xi = DD/RR$ is

$$\frac{DD}{RR}\frac{1}{\sqrt{DD}} = \frac{1+\xi}{\sqrt{DD}} \tag{2.15}$$

but it might be more intuitively written as $\sqrt{DD}/RR$. Now, the error on ξ is the same as that on $1 + \xi$ (there is no error on 1) so the error on ξ is $\frac{1+\xi}{\sqrt{DD}}$.

2.3 The cross-correlation, power spectrum and cross-power spectrum

The density fluctuations field can be also studied decomposing it in Fourier series and considering the relative importance of each scale. This information is

provided by the spectrum. At the beginning one can consider a simple functional form for the spectrum, for example a power law $P(k) \propto k^n$. These are called also *scale-free* spectra, because their logarithmic slope is the same at every scale, so there is not a any relevant physical scale. n is called *spectral index*. The scale-free spectra are used to describe the primordial spectrum or a local final (evoluted) spectrum, processed by the micro-physics. Also the spectra coming from the inflation are scale-free.

So, let us write the density fluctuation field $\delta(\mathbf{x})$ as superposition of plane waves using the Fourier transform

$$\delta(\mathbf{x}) = \frac{1}{(2\pi)^3} \int \hat{\delta}(\mathbf{k}) \exp(i\mathbf{k} \cdot \mathbf{x}) \, \mathrm{d}^3 k \tag{2.16}$$

where

$$\hat{\delta}(\mathbf{k}) = \int \delta(\mathbf{x}) \exp(-i\mathbf{k} \cdot \mathbf{x}) \, \mathrm{d}^3 x. \tag{2.17}$$

Now, if we write the correlation function as

$$\xi(r) \equiv \langle \delta(\mathbf{x}) \, \delta(\mathbf{x} + \mathbf{r}) \rangle \tag{2.18}$$

and consider the following definition for the power spectrum,

$$\langle \delta(\mathbf{k}) \, \delta(\mathbf{k}') \rangle \equiv (2\pi)^3 P(k) \, \delta_D^{(3)}(\mathbf{k} + \mathbf{k}') \tag{2.19}$$

(with $\delta_D^{(3)}$ the tridimensional Dirac delta function) we obtain

$$\xi(\mathbf{r}) = \frac{1}{(2\pi)^3} \int \mathrm{d}^3 k \int \mathrm{d}^3 k' P(k) \, \delta_D^{(3)}(\mathbf{k} + \mathbf{k}') \exp[i\mathbf{x} \cdot (\mathbf{k} + \mathbf{k}')] \exp(i\mathbf{k} \cdot \mathbf{x}) \tag{2.20}$$

$$= \frac{1}{(2\pi)^3} \int P(k) \exp(i\mathbf{k} \cdot \mathbf{r}) \, \mathrm{d}^3 k \tag{2.21}$$

so the power spectrum is the Fourier transform of the correlation function. This is known as the *Wiener-Khintchine* theorem.

If we follow the same formalism and we substitute $\delta(\mathbf{x})$ in (2.18) with the halo and matter density perturbation fields respectively we obtain the halo-matter cross correlation

$$\xi_{hm}(r) = \langle \delta_h(\mathbf{x}) \, \delta_m(\mathbf{x} + \mathbf{r}) \rangle \tag{2.22}$$

and the Wiener-Khintchine theorem still holds, linking the cross-power spectrum and the cross correlation. This let us understand how that, in theory, it is possible to obtain the same things both working in the Fourier space or in the configurations space, but in practice change from one to the other reduce the resolution because of the convolution operations. Depending on what one needs (the power spectrum or the correlation) and what one already has it can be better one method respect to the other.

CHAPTER 3

HALO MODEL

Here we learn about the winning method in the last years to describe the clustering of the dark matter structures and of its extension by Giocoli et al. 2010.

As presented in Cooray and Sheth 2002, the Halo Model (HM) is one of the possible approach the description of the non-linear gravitational clustering of the DM, characterized by having all the mass associated with virialized dark matter haloes. The HM has its origins in a paper by Neyman, Scott *et al.*, interested in describing the spatial distribution of the galaxies. They thought that it would be useful to think of the galaxies distribution as being made up of distinct clusters with different sizes. Galaxies are discrete objects so the initial work described the statistical properties of a distribution of discrete points. This description require to know some parameters: the distributions of cluster sizes, the distribution of points around the cluster center and the description of the clustering of the clusters but none of this elements were known at that time.

Now we know that much of the mass of the Universe is made of DM, initially rather smoothly distributed and that galaxies are biased tracers of the DM distribution.

The initial fluctuation field was very close to a Gaussian random field and

there were developed a perturbation theories from linear regime to higher order to describe the gravitational clustering from those initial fluctuation. These describe the evolution and mildly non-linear clustering of the DM but they fail when the clustering became highly non-linear (typically on scales smaller than a few Mpc). In perturbation theory we also don't have a rigorous framework for describing the differences between the clustering of the galaxies and that of the DM.

Numerical simulations of the large scale structure clustering were developed to study the non-linear evolution of the the DM distribution. These show that initially smooth DM distribution evolves into sheets, filaments and knots. The denser knots are called DM haloes. Simulations also show that the halo abundance, spatial distribution and internal density profiles are closely related to the properties of the initial fluctuation field. If we consider these haloes as the Neyman&Scott's clusters, the formalism would provide a way to describe the spatial statistics of the dark matter density field from the linear to the non linear regimes.

3.1 Halo structures and galaxies

As stated in Giocoli et al. 2010, for example, to know the non-linear dark-matter power spectrum ($P_{DM;NL}(k)$) is important for understanding the large scale structure of the Universe. Giocoli *et al.* analytically model the dark-matter power spectrum ($P_{DM}(k)$) and the cross-power spectrum (CPS) of the DM with the DM haloes. This is an extension to the halo model formalism and includes realistic substructure population within individual DM haloes and the scatter of the concentration parameter at fixed halo mass. This extension increases the predicted power on the small scales and it is crucial for proper modeling the cosmological weak-lensing signal due to low-mass haloes. The halo model approach can be improved in accuracy increasing the number of ingredients calibrated from the n-body simulations and from large galaxy redshift surveys which made possible accurate studies of the large-scale cosmic structures.

Galaxies are believed to form and reside in DM haloes that extend much beyond their observable radii. According to the standard scenario of structure formation galaxies with dissimilar features reside in different DM haloes and have experienced different formation histories. Some of them are located at

halo centers while others orbit around it constituting the satellite population. The clustering strength of a given galaxy population is related to that of the DM haloes which host the galaxies and it is possible to provide an accurate description of the clustering in the small-scale non-linear regime even if one has no knowledge of how the haloes themselves are clustered. In this scenario DM haloes form by gravitational instability from DM density fluctuation and subsequent merge to form increasingly large haloes. Then gas follows the DM density perturbation and once it reaches sufficiently high densities dissipative process, shocks and cooling allows stars to form.

Although the main lines of this scenario are widely accepted, the details are still poorly understood. Almost all analytic work based on the halo model approach assumes that haloes are spherically symmetric and that the matter density distribution around the center is smooth. Numerical simulations show that haloes are neither spherical nor smooth and about the 10% of the mass in cluster-sized haloes is associated with sub-clamps. An accurate model of the substructures is a necessary first step to modeling the small-scale weak gravitational convergence and shear signal. At fixed redshift the bias (the ratio between the PS of the DM haloes and of the DM) is an increasing function of the halo mass: more massive haloes (forming later and having a lower concentration) are more biased compared to less massive haloes (forming earlier and with a higher concentration).

Increasing the mass resolution in numerical simulations, recent studies have found that cores of accreted satellites haloes survive in host haloes as orbiting substructures. More massive haloes contains more sub-haloes than less massive haloes: the number of substructures per host-haloes mass is universal and this was confirmed also by other analysis. More massive haloes assemble their mass quite recently, thus the time spent by their substructures under the influence of the gravitational field of the host is shorter than in less massive hosts. This also holds for haloes of similar mass but different formation times.

3.2 Non-linear power spectrum

In this article they show how the $P_{DM;NL}(k)$ can be decomposed when a realistic model for the substructure mass function in host haloes and two models for their spatial distribution are taken into account.

In the halo model approach the two-point DM correlation function is

$$\xi(\mathbf{x}-\mathbf{x}') = \xi_{1H}(\mathbf{x}-\mathbf{x}') - \xi_{2H}(\mathbf{x}-\mathbf{x}') \tag{3.1}$$

where the first (or Poisson term) describes the contribution to the matter density from individual haloes (and thus we will call it 1H, that is one-halo term, hereafter), while the second term (2H) describes the contribution from halo correlations and $\mathbf{x}$ is the comoving coordinate. Both of them require knowledge of the halo mass function and their DM density profile but the second term needs also the two-point correlation function of haloes of different mass. This is dominated by the two halo term and has to follow the linear correlation function, so it is expressed conveniently by $\xi_{hh}(r|M_1M_2) \sim b(M_1)\, b(M_2)\, \xi_{lin}(r)$ where ξ_{lin} is the dark matter linear auto-correlation.

Considering the contribution from substructures the 1H term can be further decomposed into four contributions from the mutual correlations between smooth and clump components[1]

1. Smooth-smooth correlation
2. Smooth-clump correlation
3. TPCF between different clumps in the same halo
4. TPCF between pairs in the same clump

The sub-halo concentration depends on its mass and also on its radial distance from the host halo center. The mass-concentration relation adopted is consistent with the one assumed for isolated host haloes, such that the sub-halo mass-concentration relation, for distance larger then the virial radius of the host follows the halo mass-concentration relation.

Likewise the large-scale (2H) term can be further decomposed according to the different correlations between smooth and clump components in three contributions:

1. Smooth-smooth component correlation on large scales

[1]Here the smooth component is the matter bound to the halo but not in substructures, the clump component are the particles in substructures.

2. Smooh-clump correlation

3. Clump correlation

Both in the 1H term and in the 2H term the dominant contribution is due to the pairs in which both members are in the smooth component. Smooth-clump and clump-clump pairs never contribute more than 10% of the signal and they became even less dominant at smaller scales (larger k).

The contribution from pairs within the same sub-clump increases with k reaching values of order 50% at $k \sim 10^4$.

3.3 Cross spectra and cross-correlations

Let us consider now the cross correlation (CC) between haloes and mass as well as between substructures and mass. On the small scales we consider the two contribution to 1H term from the center of the halo, assuming that the matter is divided into smooth and clumpy components. The halo-smooth self-correlation and the halo-clump self-correlation can be write in the Fourier space as

$$P^X_{1H,\,sHs}(k,\,z) = \int \frac{M}{\bar{\rho}} n(M,\,z) \int \frac{M_{sm}}{M} u\left(k|c\left(M_{sm}\right)\right) p\left(c|M\right) \mathrm{d}c\,\mathrm{d}M \tag{3.2}$$

$$P^X_{1H,\,Hc}(k,\,z) = \int \frac{M}{\bar{\rho}} n(M,\,z) \int U_s(k,\,c)\, \mathcal{I}_{c(M)}(k,\,z)\, p\left(c|M\right) \mathrm{d}c\,\mathrm{d}M \tag{3.3}$$

where $u\left(k|c\left(M_{sm}\right)\right)$ and $U_s(k,\,c)$ are the Fourier transforms of the density profiles of the smooth component and of the clump distribution respectively. M_{sm} is the smooth mass of the host halo.

Regarding the contribution to large scales (2H term) the idea is the same: we imagine sitting at the center of the host halo and cross-correlate with the smooth and the clumpy components of a distant halo. In this case we also have to take the halo bias relative to the DM distribution into account, as well as the $P_{DM;NL}(k)$.

The smooth and the clump cross-correlation PS are:

$$P^X_{2H,\,Hs}(k,\,z) = \frac{P_{lin}(k,\,z)}{\bar{n}_h} \int n(M,\,z)\, b(M,\,z)\, \mathrm{d}\mathcal{MS}^I(k,\,z) \tag{3.4}$$

$$P^X_{2H,\,Hc}(k,\,z) = \frac{P_{lin}(k,\,z)}{\bar{n}_h} \int n(M,\,z)\, b(M,\,z)\, \mathrm{d}\mathcal{MC}^I(k,\,,z)\,. \tag{3.5}$$

Now we have to place ourselves at the center of a clump and determine the cross-correlation on small scales.

The 1H term will be the sum of three components:

1. the self correlation (SC) with the substructure mass
2. the CC with the mass in the other substructures contained in the same host halo
3. the CC with the smooth component

We obtain

$$P^X_{1H,\,S\,s}(k,\,z) = \frac{1}{\bar{n}_s}\int \frac{M}{\bar{\rho}} n(M,\,z) \int \frac{M_{sm}}{M} u\left(k|c\left(M_{sm}\right)\right) U_s\left(k|c\left(M\right)\right) N_s\left(c\left(M\right),\,z\right) p\left(c|M\right) \mathrm{d}c\,\mathrm{d}M \tag{3.6}$$

$$P^X_{1H,\,S\,c}(k,\,z) = \frac{1}{\bar{n}_s}\int \frac{M}{\bar{\rho}} n(M,\,z)\, U_s^2\left(k|c\left(M\right)\right) \mathcal{I}_{c(M)}(k,\,z)\, N_s\left(c\left(M\right),\,z\right) p\left(c|M\right) \mathrm{d}c\,\mathrm{d}M \tag{3.7}$$

$$P^X_{1H,\,sS\,c}(k,\,z) = \int \frac{M}{\bar{\rho}} n(M,\,z) \int \mathcal{I}_{c(M)}(k,\,z)\, p\left(c|M\right) \mathrm{d}c\,\mathrm{d}M. \tag{3.8}$$

For the 2H term we have two equations analogous to those of the halo-matter cross-correlation and we need to integrate over the substructure mass function:

$$P^X_{2H,\,S\,s}(k,\,z) = \frac{P_{lim}(k,\,z)}{\bar{n}_s}\int n(M,\,z)\, b(M,\,z) \int N_s\left(c\left(M\right),\,z\right) U_s\left(k|c\left(Mz\right)\right) p\left(c|M\right) udc\,\mathrm{d}M\mathcal{S}^I(k,\,z \tag{3.9}$$

$$P^X_{2H,\,S\,c}(k,\,z) = \frac{P_{lin}(k,\,z)}{\bar{n}_s}\int n(M,\,z)\, b(M,\,z) \int N_s\left(c\left(M\right),\,z\right) U_s\left(k|c\left(M\right)\right) p\left(c|M\right) \mathrm{d}c\,\mathrm{d}M\mathcal{C}^I(k,\,z)\,. \tag{3.10}$$

The cross-correlation between substructures and mass is sensitive to the choice of the radial sub-clump distribution. In the model by Hayashi&White the 1H term is due to the dark matter density profile of the host halo, while the two halo term by the product between bias factor and the mass autocorrelation function predicted by the linear theory.

As presented in Section 2.3 the we can also obtain the same information in another way. Let us consider the configurations point of view (Hayashi and

White 2008). The cross-correlation between halo centers and mass ξ_{hm} is the spherically averaged halo density profile averaged over all the haloes in the dataset. Its shape is composed by two parts, the one-halo and two-halo terms presented in the previous sections. They are dominated by the particles within the same halo or in different haloes respectively.

It can be also easily shown that ξ_{hm} on large scales follows closely the mass auto-correlation function with a mass-dependent offset in amplitude, the *halo bias factor* $b(M)$.

So we can write:

$$\xi_{hm}(r; M) = \begin{cases} \xi_{1h}(r) & \text{if } \xi_{1h}(r) \gg \xi_{2h}(r) \\ \xi_{2h}(r) & \text{if } \xi_{1h}(r) \ll \xi_{2h}(r) \end{cases} \tag{3.11}$$

with

$$\xi_{1h}(r) = \frac{\rho_{halo}(r; M) - \bar{\rho}_m}{\bar{\rho}_m} \tag{3.12}$$

$$\xi_{2h}(r) = b(M)\, \xi_{lin}(r)\,. \tag{3.13}$$

The one-halo term has been studied extensively and it is well fitted by the NFW profile or some modified forms of it.

As presented before, regarding the substructures-matter cross correlations, the contributions are:

- for the 1H term the correlation with the substructure mass, the mass in other substructures of the same halo and the correlation with the smooth component
- for the 2H term we have only two contributions, the one from the correlation with the smooth component and the one from the correlation with the clump one.

In this work we do not distinguish the contributions from the smooth and the clump component, they appear as a single contribution.

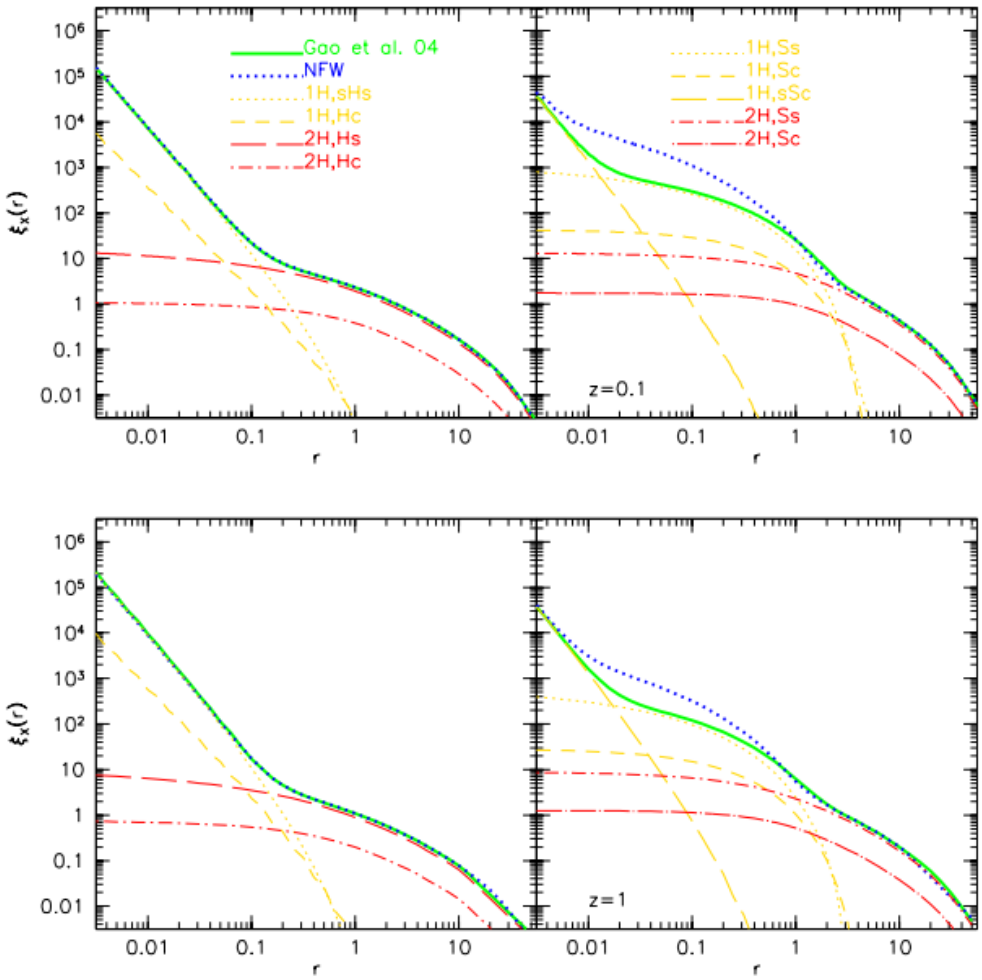

Figure 3.1: Halo and subhalo-mass cross-correlation at redshift $z = 0.1$ (top) and $z = 1$ (bottom) from Giocoli et al. 2010. In each panel can be seen the contribution of haloes and substructures and mass, respectively.

CHAPTER 4

SIMULATIONS

In this chapter we will take confidence with the concept of simulation and we would have a look to one of the most important simulation of the astrophysics done till now.

A good introduction on what is a simulation and its implication on the modern scientist trade is provided by Karniadakis and Kirby 2003. In particular the focus on how the new technologies changed the scientist workflow and skills. Indeed, although until not so many years ago the work of the scientists was connected with the observations in a laboratory (or at the telescope) and/or with paper and pencil. Some rudimentary machines were available to help the counts sometimes. In few years those machines became powerful enough not only to became a fundamental part of the analysis but also to permit the scientist to recreate a numerical model of the system under study and let it evolve into the computer.

The modern scientist then often spend more and more time in front of a laptop, a workstation, or a parallel supercomputer and less and less time in the physical laboratory or in the workshop. Sometimes the old approach of "cut-and-try" has been replaced by "simulate-and-analyse in several disciplines, from the astrophysics, to particle physics, to biology. As a side effect, the modern

scientist must be able to use the new tools, so he have to join together the knowledge in his field of research with the computer programming.

In the classical scientific approach, the physical system is first simplified and set in a form that suggests what type of phenomena and processes may be important, and correspondingly what experiments are to be conducted. In the absence of any known-type governing equations, dimensional inter-dependence between physical parameters can guide laboratory experiments in identifying key parametric studies. The database produced in the laboratory is then used to construct a simplified "engineering" model which after field-test validation will be used in other research, product development, design, and possibly lead to new technological applications. This approach has been used almost invariably in every scientific discipline, i.e., engineering, physics, chemistry, biology, etc.

The simulation approach follows a parallel path but with some significant differences. First, the phase of the physical model analysis is more elaborate: the physical system is cast in a form governed by a set of partial differential equations, which represent continuum approximations to microscopic models. Such approximations are not possible for all systems, and sometimes the microscopic model should be used directly. Second, the laboratory experiment is replaced by simulation, i.e., a numerical experiment based on a discrete model. Such a model may represent a discrete approximation of the continuum partial differential equations, or it may simply represent a statistical representation of the microscopic model. Finite difference approximations on a grid are examples of the first case, and Monte Carlo methods are examples of the second case. In either case, these algorithms have to be converted to software using an appropriate computer language, debugged, and run on a workstation or a parallel supercomputer. The output is usually a large number of files of a few Megabytes to hundreds of Gigabytes, being especially large for simulations of time-dependent phenomena. To be useful, this numerical database needs to be put into graphical form using various visualization tools. Visualization can be especially useful during simulations where interactivity is required as the grid may be changing or the number of molecules may be increasing.

The question is if this is a new science, and how one could formally obtain such skills. Moreover, does this constitute fundamental new knowledge or is it a "mechanical procedure" an ordinary skill that a chemist, a biologist or an en-

gineer will acquire easily as part of "training on the job" without specific formal education. It seems that the time has arrived where we need to reconsider boundaries between disciplines and reformulate the education of the future simulation scientist, an inter-disciplinary scientist.

Let us re-examine some of the requirements following the various steps in the simulation approach. The first task is to select the right representation of the physical system by making consistent assumptions in order to derive the governing equations and the associated boundary conditions. The conservation laws should be satisfied; the entropy condition should not be violated; the uncertainty principle should be honored. The second task is to develop the right algorithmic procedure to discretize the continuum model or represent the dynamics of the atomistic model. The choices are many, but which algorithm is the most accurate one, or the simplest one, or the most efficient one? These algorithms do not belong to a discipline! The third task is to compute efficiently in the ever-changing world of supercomputing. The fourth task is to assess the accuracy of the results in cases where no direct confirmation from physical experiments is possible such as in nanotechnology or in biosystems or in astrophysics. Reliability of the predicted numerical answer is an important issue in the simulation approach as some of the answers may lead to new physics or false physics contained in the discrete model or induced by the algorithm but not derived from the physical problem. Finally, visualizing the simulated phenomenon, in most cases in three-dimensional space and in time, by employing proper computer graphics (a separate specialty on its own) completes the full simulation cycle. The rest of the steps followed are similar to the classical scientific approach.

4.1 The GIF and GIF2 Simulations

The GIF simulation (Diaferio, White, and Kauffmann 1999) is a set of N-body simulations developed to follow the formation and evolution of galaxies and their clustering properties. They use dissipationless simulations to track the formation and merging of dark matter haloes as a function of redshift and some prescriptions taken from semi-analytic models of galaxy formation for for gas cooling, star formation, supernova feedback and the merging of galaxies within the haloes. One of the motivation of the simulation was to test the theories to

estimate cosmological parameters, such as the density Ω,or the cosmological constant Λ. The simulations were run as a part of the GIF project and the code used is called Hydra. A set of four simulations with $N = 256^3$ and with different cosmological parameters was run but in this work we were interested in the one with ΛCDM cosmology and the following parameters:

- $\Omega_m(z = 0) = 0.3$
- $\Omega_\Lambda(z = 0) = 0.7$
- $H_0 = 70$ km/s/Mpc
- $\sigma_8(z = 0) = 0.9$
- gravitational softening = 30 kpc/h cubic spline
- boxsize = 141.3 Mpc/h on a side
- $N_{part} = 256^3 = 16777216$
- particle mass = $1.4x10^{10} M_\odot/h$

The Gif2 simulation (Gao, White, and Jenkins 2004) was developed to investigate the subhalo populations of dark matter haloes in the concordance ΛCDM cosmology. The simulation was performed with Hydra in the first part and finished using GADGET to save computational time.

The simulation parameters are

- $\Omega_m(z = 0) = 0.3$
- $\Omega_\Lambda(z = 0) = 0.7$
- $H_0 = 70$ km/s/Mpc
- $\sigma_8(z = 0) = 0.9$
- gravitational softening = 6.6 kpc/h cubic spline
- boxsize = 110 Mpc/h on a side
- $N_{part} = 400^3 = 64000000$
- particle mass = $1.73x10^9 M_\odot/h$

4.2 The Millennium Simulation

The Millennium Simulation (MS) is one of the bigger and most important simulation ever made in astrophysics. Its main purpose is to study the formation of the structures now visible in the Universe. Its based on the ΛCDM model. This model, together with the theory of cosmic inflation, makes a clear prediction for the initial conditions for the structure formation and predicts that structures grow hierarchically through gravitational instability. Testing this model requires that the precise measurements delivered by galaxy surveys can be compared to robust and equally precise theoretical calculations. The MS, presented in Springel et al. 2005, is a simulation of the growth of the DM structure using 2160^3 particles, following them from redshift $z = 127$ to the present in a cube-shaped region 2230 billions light-years on a side. In post-processing they also follow the formation and evolution of the galaxies and quasars.

The principal goals of all of these surveys are to shed light on how galaxies form, to test the current model for the growth of cosmic structure and search for signatures that may clarify the nature of DM and dark energy.

Two problems have so far precluded such predictions:

- accurate estimates of clustering require simulations of extreme dynamic range, encompassing volumes large enough to contain representative population of rare objects (such as rich galaxy clusters or quasars), yet resolving the formation of individual low-luminosity galaxies

- critical aspects of galaxy-formation physics are uncertain and beyond the reach of direct simulation: for example the structure of interstellar medium and its consequences for star formation and for the generation of galactic winds, the ejection and mixing of heavy elements, and AGN (active galactic nuclei) feeding and feedback effects that must be treated by phenomenological models whose form and parameter are adjusted by trial and error as part of the overall data-modelling process

While the initial, linear growth of density perturbation can be calculated analytically, the collapse of fluctuations and the subsequent hierarchical build-up of structure is a highly non-linear process that is easy accessible only through direct

numerical simulation. The dominant mass component, the CDM, is assumed to be made of elementary particles that currently interact only gravitationally, so the collisionless DM fluid can be represented by a set of discrete point particles. This representation as a N-body system is a coarse approximation whose fidelity improves as the number of particles of the simulation. This high-resolution simulation is called the "Millennium Simulation" because of its size, it follows $\sim 10^{10}$ particles from $z = 127$ to $z = 0$ in a cubic region of 500 Mpc/h and it offers a substantially improved spatial and time resolution within a large cosmological volume.

The mass distribution in a ΛCDM universe has a complex topology, often described as a "cosmic web". On larger scales, there is a little discernible structure and the distribution appears homogeneous and isotropic. Zooming in, the final image reveals several hundred DM substructures, resolved as independent, gravitationally bound objects orbiting within the cluster halo. These substructures are the remnants of DM haloes that fell into the cluster at earlier times. At present, there are about 18 million haloes above a detection threshold of 20 particles; 49.6% of all particles are included in these haloes. These statistics provide the most precise determination to date of the mass function of CDM haloes. In the range that is well sampled in the MS ($z \leq 12$, $M \geq 1.7 \times 10^{10} M_{\odot}/h$) the results are remarkably well described by an analytic formula derived from fits to previous simulations.

On large scales and at early times, the mode amplitudes of the dark matter power spectrum grow linearly, roughly in proportion to the cosmological expansion factor. Non-linear evolution accelerates the growth on small scales when the dimensionless power $\Delta^2(k) = k^3 P(k) / \left(2\pi^2\right)$ approaches unity (the power spectrum $P(k)$ measures the variance of the density fluctuations on the scale of wave-number k). This regime can only be studied accurately using numerical simulations. In the Millennium Simulation they are able to determine the non-linear power spectrum over a larger range of scales.

N-body simulations of CDM universes are now of such size and quality that realistic modelling of galaxy formation in volumes matched to modern surveys has become possible.

The Millennium Simulation was carries out with a customized version of the GADGET-2 code, using the *TreePM* method for evaluating gravitational forces.

This is a combination of a hierarchical multipole expansion, or *tree* algorithm, and a classical Fourier-transform particle-mesh method. The calculation was performed on 512 processors of an IBM p690 parallel computer at the Computing Centre of the Max-Planck Society in Garching, Germany, It used almost all of the 1 terabyte of physically distributed memory available. It required about 350000 processor hours of CPU time, or 28 days of wall-clock time. The mean sustained floating-point performance (as measured by hardware counters) was about 0.2 teraflops, so the total number of floating-points operations carried out was of the order of 5×10^{17}.

The cosmological parameters of their ΛCDM simulation are $\Omega_m = \Omega_{DM} + \Omega_b = 0.25$, $\Omega_b = 0.045$, $h = 0.73$, $\Omega_\Lambda = 0.75$, $n = 1$ and $\sigma_8 = 0.9$. Ω_m denotes the total matter density in units of the critical density for closure $\rho_{crit} = 3H_0^2/(0\pi G)$. Similarly Ω_b and Ω_Λ denote the densities of baryons and dark energy at the present day. The simulation volume is a periodic box of size 500 Mpc/h and individual particles have mass of $8.6 \times 10^8 h^{-1}\, M_\odot$. This volume is large enough to include interesting rare objects but still small enough that the haloes of all luminous galaxies brighter than $0.1L$ (where L is the characteristic luminosity of galaxies are resolved with at least 100 particles).

A smaller test version of the Millennium Run, the Milli-Millennium Simulation, was also performed. This simulation used the same cosmology and resolution as the Millennium Simulation but in a 62.5 Mpc/h box with 19,683,000 particles. We used the data from this simulations to check the code in the early stages.

4.3 The Millennium II Simulation

The Millennium-II Simulation, MSII, (Boylan-Kolchin et al. 2009) has the same parameters of the MS but it was carried out in a periodic cube of 100 Mpc/h so it has 5 times better spatial resolution and 125 better mass resolution. Comparing the results of the two simulations demonstrates the very good convergence in DM statistics such the halo mass function, the sub-haloes abundance distribution and so on. Putting together this two simulations one can obtain precise results on these statistics over an unprecedented range of scales. Thus we are investigating the properties of the structures and substructures at small scales we decided to

use the MSII because of its better spatial resolution on those scales.

Opposite to the simulations trend after the MS (bigger volume at fixed particles number - it reduces the computational effort) the MSII focused on smaller volumes. At fixed particles number this led to a higher mass resolution but it requires much more computation effort. The higher mass resolution therefore is of great interest in galaxy formation, with relevant mass scales smaller than the scales of large clustering. The evolution of low-mass galaxies is important because they prepare the initial condition for the more massive system formed later.

The MSII high resolution is useful also for the study of the DM substructures in DM haloes.

In particular this simulation follows 2160^3 particles in a box of side length 100 Mpc/h and was run with GADGET-3 code.

4.4 Gadget and the database

Gadget (Springel 2005), the code used for these simulations, is based on a combination of a tree algorithm, also called hierarchical multipole expansion, and a classical Fourier transform particle-mesh method. The code decompose the simulation volume into compact domains, each served by one processor, using a space filling Peano-Hilbert curve divided into segments. This choice also minimize the communication between the different processors during the short-range forces computation. During the execution of the code some calculation are performed to characterize the simulation output, such as the dark matter auto-correlation. Moreover, in each output of the simulations, haloes are identified using a friend-of-friend groupfinder and each FOF halo is decomposed into locally over-dense and self-bound substructures (subhaloes) using the SUBFIND algorithm. In addition they use semi-analytic methods to simulate the evolution of galaxies population taking care of the gas cooling, the star formation and so on (Hayashi and White 2008)

The results of the two Millennium simulations are stored in a SQL database (Lemson 2006) accessible from Internet. The access to the complete particles positions database is limited because of its dimension. The database is organized in tables and the contents are indexed in more than one way to permit a faster

access to the data. Some view, functions and stored procedures are yet available to the users, for example the possibility to search for particles in spherical ball around a position.

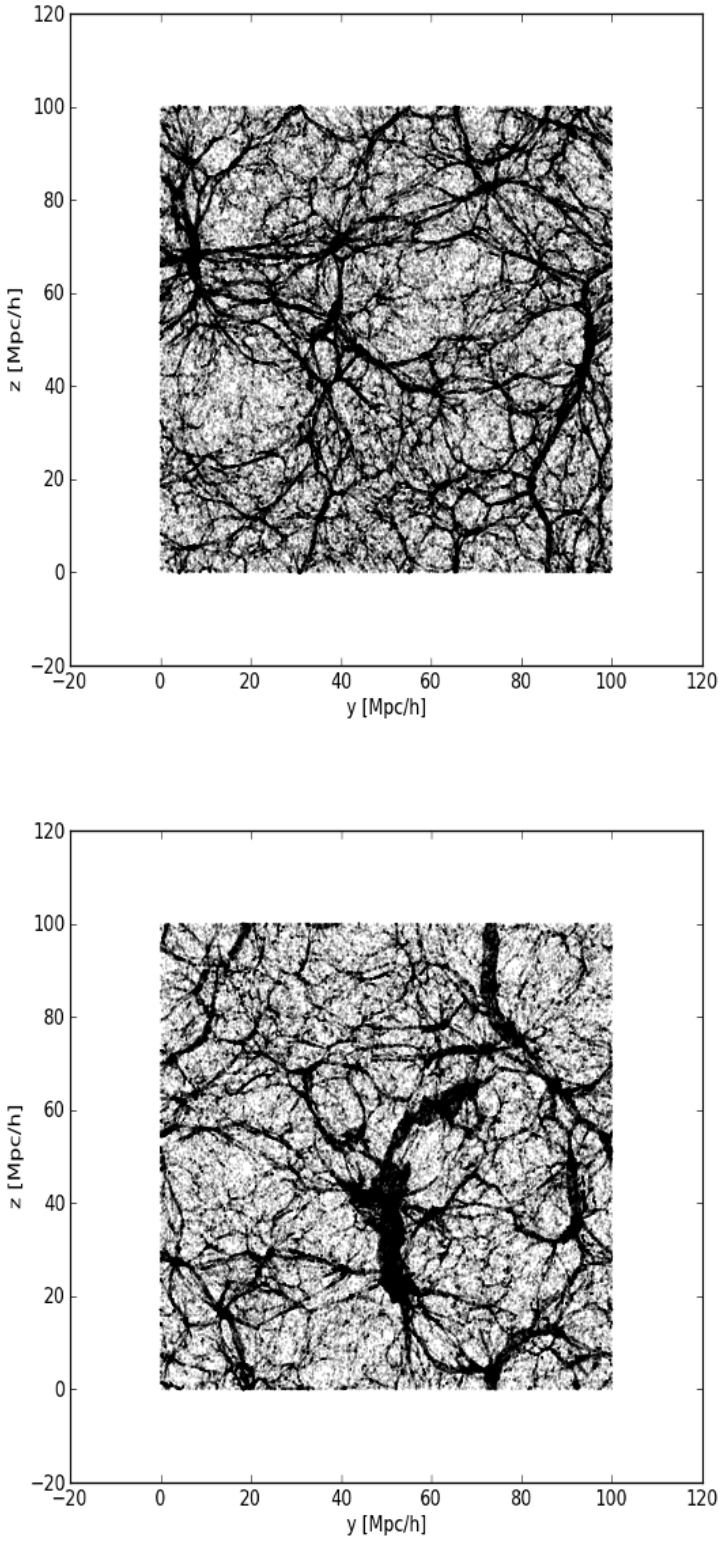

Figure 4.1: Two slice of the Millennium II simulations. It can be seen the presence of a big cluster in the second images that slowed down the counting operations.

CHAPTER 5

TREES

In this chapter we will understand what a kd-tree is, why it was created and the basics of the kd-tree optimization.

Pairwise distance computations are of fundamental importance in many fields, not only in astronomy and cosmology. They are often used in machine learning, graphics computation and so on. A particular class of this type of problems is called *all-query*, *all-point-pairs* or *N-body*-like problems (Ram et al. 2009, Gray and Moore 2001). In particular, as we stated above, the two-point correlation function can be thought of roughly as a measure of the clumpiness of a set of points and it is easily defined as the number of pairs of points in a dataset which lie within a given radius r of each other.In this type of problems we have the interaction between a reference of two sets of points of size $O(N)$. The direct solution requires a total running time of $O\left(N^2\right)$. This type of dependence is often too high to permit the computation. The general approach usually followed to solve this problem is to reduce the number of distance computation. One of the most famous problem that need this approach is the *nearest neighbours* problem (NNP). Indeed one of the first appearance of *kd*-tree was in 1977 by Freidman, Bentley, and Finkel 1977 to solve the "best matches", or nearest neighbours problem. The problem is to find, among big datasets, the m closest matches or

nearest neighbours to a given query record according to some dissimilarity or distance measure. Formally, given a dataset of N data-points (each described by K keys) and a dissimilarity measure or distance D we have to find the m matches closest to a query point (non necessary in the dataset). The most famous problem of this kind is the so called "post-office problem"[1]. The solution proposed by Freidman, Bentley, and Finkel 1977 require a computation proportional to $kN \log N$. The expected number of points examined in each search is independent of the dataset size and the expected computation to perform each search is proportional to $\log N$.

Structures use for associative searching

The first, rough, technique they present for solving the NNP is the *cell* methods. The k-dimensional key space is divided into small identically sized cells. A spiral search of the cells from any query record will find the best matches of that record. This is extremely costly in terms of space and time, especially when the dimensionality of the space is large.

The key point, in every strategy, is to minimized the number of record examinated. In Freidman, Bentley, and Finkel 1977 the authors quickly presents some of the strategies, including clustering techniques (using the triangle inequality), the formation of a projection of the records onto one or more keys keeping a linear list of these keys (this does not require to satisfy the triangle inequality). The expected computation required for the last strategy is proportional to $km^{1/k}N^{1-1/k}$. Other ways use binary keys with the Hamming distance[2] or the Voronoi diagram[3] in the special case of only two keys and Euclidean space. The last case can search for best matches in worst case of $O\left[(logN)^2\right]$ after a dataset

[1]The post-office problem is a problem presented by Donald Knuth in The Art of Computer Programming in 1973. The problem is about assigning to a residence the nearest post office among all.

[2]In information theory, the Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. Put another way, it measures the minimum number of substitutions required to change one string into the other, or the number of errors that transformed one string into the other. For example the Hamming distance between "toned" and "roses" is 3.

[3]In mathematics, a Voronoi diagram is a special kind of decomposition of a metric space determined by distances to a specified discrete set of objects in the space, e.g., by a discrete set of points.

organization that requires a storage proportional to N and a time proportional to $O(N \log N)$.

The last strategy is proposed by Finkel and Bentley (Finkel and Bentley 1974) and involve a tree structure, called *quad-tree* for the storage of the keys. It is a generalization of the binary tree for storing data on single keys. Bentley develops a different generalization of the same one-dimensional structure, called *kd*-tree. This paper introduces an optimized *kd*-tree for the problem of finding best matches. This data structure is very effective in partitioning[4] the points in the dataset so that the average number of examinations involved in searching the set is quite small. This method can be applied with a wide variety of dissimilarity measures and does not require that they obey the triangle inequality. The storage required for data organization is proportional to N while the computational is proportional to $kN \log N$. For large datasets the expected number of examinations required for the search is shown to be independent of the dataset size, N. The time spent in descending the tree during the search is proportional to $\log N$ so that the expected time required to search for best matches with this method is proportional to $\log N$.

5.1 Definition of KD-Tree

A *kd*-tree is a generalization of the simple binary tree used for sorting and searching, in particular the *kd*-tree is a binary tree in which each node represents a subset of the dataset. The root of the tree represents the entire dataset. Each non terminal node has two *sons* or *successor nodes*. These successor nodes represents the two subsets defined by the partitioning. The terminal nodes, also called *leaves*, represent mutually exclusive small subsets of the data records, which collectively form a partition of the data space. The leaves are originally called by Freidman buckets.In the case of one-dimensional searching a point is represented by a single key and a partition is defined by some value of that key. All points in a subset with key values less than or equal to the partition value belong to the left son while those with a larger value belong to the right son. The original *kd*-tree proposed by Bentley chooses the discriminator for each node on the

[4]We can consider a partition the way we divide the space of the data.

basis of its level in the tree cycling through the keys in order. The paper (Freidman, Bentley, and Finkel 1977) deals with choosing both the discriminator and the partition value for each subset, as well as the leaves size, to minimize the expected cost for searching for the nearest neighbours.

5.2 The search algorithm

The KD-Tree data structure provides an efficient mechanism for examining only those points closest to the query point, greatly reducing the computation required to find the best matches. The search algorithm is in general a recursive procedure. The first invocation pass the root-node of the tree and the query point to the recursive function. The domain of the node, that is, the geometric boundaries that delimit the subset represented by the node are available in some way or they can be calculated. The domain of the root node is defined to be plus and minus infinity on all the keys in the early implementation but now is usually the minimum "box" containing the data. The geometric boundaries at each level are determined by the partitions defined at the nodes above it in the tree. At each node the partition divides the current sub-file and set the limits for the discriminator key in the new sub-files. These limits define a volume in the multidimensional *key*-space containing the sub-file. This volume is smaller for nodes deeper in the tree. If the node under investigation is terminal then all the points in the leaf are examined. If the node under investigation is not terminal the recursive procedure is called for the nodes representing the subset.

The optimized KD-Tree

The goal of the optimization is to minimize the expected numbers of records examined with the search algorithm. The parameters to be adjusted are the discriminating key number, the partition value at each non-terminal node (innernode) and the dimension of the leaves.

The solution to the optimization will in general depend upon the distribution of query points in the data key space. In practice it depends also on the hardware to be used, i.e. the memory and the CPUs available, and on the speed of the various component. Usually one has no knowledge of this in advance. Thus

they propose a procedure that is independent of the distribution of queries and only uses information contained in the dataset. Such a procedure will be seen to be good for all possible query distribution but will not be optimal for any particular one.

A second restriction is that the solution values for discriminating key number and partition value at any particular node depended only on the sub-file represented by that node. This restriction is necessary so that we can define the *kd*-tree recursively avoiding a general binary tree optimization that will be require a non-polynomial time complexity. Under these two restriction they provide a prescription for choosing the discriminating key and partition value at each innernode. Because the information provided to a binary choice is maximal when the two alternatives were equally likely each data point should have had equal probability of being on either side of the partition. This criterion dictates that we locate the partition at the *median* of the marginal distribution of the key values, irrespective of which key is chosen for the discriminator. The search algorithm can exclude some subsets if the distance to the partition is greater than the radius of the query ball. The probability of the partition intersecting the ball is least (average over all possible query location) for that key which exhibited the greatest spread or range in values before the partitioning. The prescription for optimizing the *kd*-tree, then, is to choose at every innernode the key with the largest spread in values as the discriminator and to choose the median of the discriminator key values as partition. In practice Arya et al. 1998 state that can be a good idea chose split the partition key in two halves.

Analysis of performance

The storage required for data organization is proportional to the data size N. The discriminating key number and partition value must be stored for each non-terminal node. The number of innernodes is $\lceil N/b \rceil - 1$ where b is the number of points in each leaf. The computation required to build the *kd*-tree is easily derived. At each level of the tree the entire set of key values must be scanned. Thus requires a computation proportional to kN. The depth of the tree is $\log N$ so the total computation to build the tree is proportional to $kN \log N$. The expected time performance of the search is not so easily derived but with some geometric

arguments it is possible to reach some results. First, to minimize the number of bucket examined every bucket should contain only one record. Then, it is possible to demonstrate that choosing the median for the partitioned ensures that each leaf will contain very nearly b points, where b is the maximum leaf size. More, choosing the key with the largest spread in values at each node ensures that the geometric shape of these leaf will be reasonably compact. In fact the expected shape of these buckets is hypercubical with edge length equal to the kth root of the volume of the space occupied by the leaf. The edges are parallel to the coordinate axes. The effect of the optimized *kd*-tree partitioning, then, is to divide the coordinate space into approximately hypercubical subregions, each containing very nearly the same number of points.Another important result is that the expected number of points examined is *independent* of the file size N but it can depend on the distribution of the data points. The goal is hence to minimized the leaves overlapped for a small area by the query-ball. This is because we would analyze a big leaf only to find few interesting data point in the query-ball. Then partitioning should be as fine as possible. Ultimately it is possible to demonstrate that the expected search time for the m best matches is proportional to $\log N$.

Implementation

Till here the discussion is focused on the number of records examined as the sole criterion for performance evaluation of the algorithm. This has the advantage that evaluation is independent of the details of implementation and the computer upon which the algorithm is executed. Although the computational requirements of the algorithm are strongly related to the number of records examined, there are other considerations including the consideration required to build the *kd*-tree and the overhead computation required to search the tree. In the overhead required to search the tree it must be considered the calculation required to find the nodes intersected by the query-ball. This calculation must be performed at each innernode visited in the search. It is then more computationally efficient to have larger leaf sizes even thought this increase the number of records examined. The final result is a balance between all of this consideration, trying to find the minimum of the combination of all the contributes.

Implementation on secondary storage

Efficient operation on the *kd*-tree algorithm does not require that all of the leaf reside in fast memory. During the preprocessing these data can be arranged on an external storage device so that records in the same leaf are stored together. Leaves close together in the tree can be stored similarly. Since the search algorithm examines a small number of leaves on the average, there will be few access to the external storage for each query. For extremely large files it is not even necessary that the entire *kd*-tree reside in fast memory. Only the top levels of the tree need to be in fast memory, the lower levels can be stored on an external device under an arrangement that keeps innernodes close to their sons.

5.3 KD-Tdree optimization

Gray and Moore 2001 presents a suite of other geometric considerations which are applicable in principle to any "N-body" computation. Those algorithms exhibit favorable asymptotic scaling and are empirically several orders of magnitude faster than the naive computation even for small datasets.First the authors stress the importance of the *mrkd*-tree, that is a conventional *kd*-tree decorated at each node with extra statistics about the node's data, such as their count, centroid and covariance. The older techniques can be summarized as follows.

Quadratic algorithm. The most naive approach is to simply compare each point to each other one, incrementing a count if the distance between them is less than r. This has $O\left(N^2\right)$ cost, unacceptably high for problems of practical interest.

Binning and gridding algorithms. It is like the "cell" method presented early. The idea of binning is simply to divide the data space into a fine grid defining a set of bins, perform the quadratic algorithm on the bins as if they were individual data, then multiply by the bin sizes as appropriate to get an estimate of the total count. The idea of gridding is to divide the data space into a coarse grid, perform the quadratic algorithm within each bin and sum the result over all bins to get an estimate of the total count. These

are both of course very approximate methods yielding large errors. They are not usable when r is small or r is large, respectively.

Range-searching with a *kd*-tree. The idea is that we will make each data point in turn a query point and then execute a range search of the *kd*-tree to find all other points within distance r of the query. A search is a depth-first traversal of the *kd*-tree, always checking the minimum possible distance d_{min} between the query and the hyper-rectangle surrounding the current node. If $d_{min} > r$ there is no point in visiting the node's children and the computation is saved. This is called *exclusion-based pruning*. The range searching avoids computing most of the distances between pairs of points further than r apart, which is considerable saving if r is small.

Better geometric approaches

After this brief review of the past approaches we can consider some of the advices presented in Gray and Moore 2001. The use of this techniques in our code permitted to touch from 0.1% to 0.01% of the nodes, saving time considerably.

Single-tree search (Range-Counting Algorithm)

A straightforward extension can exploit the fact that unlike the conventional use of range searching these statistics frequently do not need to retrieve all the points in the radius but merely to count them. The *mkd*-tree has, in each node, the count of the number of data it contains, the simplest kind of cached sufficient statistic. At a given node, if the distance between the query and the farthest point of the bounding box of the data in the box is smaller than the radius r, clearly every point in the node is within the range of the query. We then can simply add the node's stored count to the total count. This is called *subsumption*.

Dual-tree algorithm

The idea is to consider the query points in chunks as well as defines by the nodes in a *kd*-tree. Dual-tree search can be thought of as a simultaneous traversal of two trees, instead of iterating over the query points in an outer loop and only exploiting single tree search in the inner loop. Dual tree search is based on

the node-node comparisons while single-tree search was based on the point-node comparisons. Importantly, both kind of pruning can now be applied to many query points at once, instead of each nearby query point rediscovering the same prune during the single-tree search. The intuition behind dual-tree search advantage can be seen by considering two cases. First, if r is so large that all pairs of points are counted then single tree search will perform $O(N)$ operations where each query point immediately prunes at the root, while dual-tree search will perform $O(1)$ operations. Second, if r is so small that no pairs of points are counted, single-tree search will run to one leaf for each query, meaning total work $O(N \log N)$, whereas dual-tree search will visit each leaf one, meaning $O(N)$ work. Note, however, that in the middle case of a medium-size r, duel-tree is theoretically only a constant factor superior to a single-tree. (Also in Ram et al. 2009).

Non-redundant dual-tree search

So far, we have discussed two operations which cut short the need to traverse the tree further, exclusion and subsumption. Another form of pruning is to eliminate node-node comparisons which have been performed already in the reverse order. This is the case in which the two datasets to be compared are the same, so that the pair $A - B$ coincide with $B - A$. This improvement can be done simply by (virtually) ranking the data-points according to their positions in a depth-first traversal of the tree, then recording for each node the minimum and the maximum ranks of the points in it owns and pruning whenever *query-node*'s maximum rank is less than *data-node*'s minimum rank. This kind of pruning is not practical for single-tree search. In our opinion this can be done in practice in a better way. In this work we add some kind of rank to the nodes during the tree build.

Multiple radii simultaneously

Most often in practice the two point is computed for many successive radii so that a curve can be plotted indicating the clumpiness on different scales. Though the method presented so far is fast, it may have to be run once for each of, for example, 1000 radii. It is possible to perform a single a single, faster computa-

tion for all radii simultaneously by taking advantage of the nesting structure of the ordered radii, with an algorithm which recursively narrows the radii which still need to be considered based on the current closest and farthest distances between the nodes.

Previous methods were based on linked lists due to memory lack because the tree structure need a lot of memory respect to the linked lists. In this way one has to divide the space into cells and touch the cells to know where to calculate the distances.

CHAPTER 6

DEVELOPING A PYTHON CORRELATION CODE

The goal of this work, as we previously stated is to obtain the cross-correlation between the halo centers and the DM, on scales from 10 kpc/h up to 5 Mpc/h. To achieve this target we had to build some tools not available, for example an easy way to count the couples as fast as possible and some functions to read the initial data and to store the data read in a comfortable format.

The first thing to do was to decide which language to use. The candidates were

- well-known Fortran, compiled, probably the most tested and the fastest language for scientific computations
- C/C++, compiled, maybe the most used language in the world, fast, low level, and with a not-so-comfortable syntax
- Python, interpreted, a modern, flexible and widely used language by Guido Van Rossum

We have decided to use Python.

6.1 Python

The first objections one can do is that I needed a fast way to do computation, and Python is not famous for its speed. It is interpreted, high level and most of the existing code in astrophysics is Fortran or, at least, C. A `for` loop in Python takes order of magnitude more time respect to C or Fortran. Despite of all of these considerations and other, more or less technical, the use of Python is nevertheless growing fast in the scientific community. Why?

Flexibility

Python is a general purpose, object and aspect oriented language, so it permits a lot of different styles of programming and to implement own code following many different design patterns. It doesn't have the strict and innatural syntax of C/C++ nor the obsolete style of Fortran. In Python everything is an object, with its methods and attributes. In Python its easy to cover the entire workflow, from the data analysis to the plot of the results with only one language.Because of its modularity it permits an easy integration, improvement and reuse of the code.

Fast developing

It is modern and flexible syntax and design permits a fast develop and debug because one can test in the interpreter the pieces of code. The libraries are self-documenting and the *introspection* permits to easy understand what is happening or how a tool works. Introspection means that you can explore an object interactively, it is possible to ask the properties, the values or any important information about it.In Python no `segmentation fault` is possible: every error (in Python you call them *exception*) is managed automatically or following the user instruction.

Libraries

Python itself is usually not fast enough for a massive computational application, both because it is interpreted and because of its design. For some things (list comprehension, ...) is quite fast, usually enough for the average user needed. It is really not fast enough for the purpose of this work. Because it is a general

purpose language it is obviously not specialized in treat mathematical objects like array, matrices and so on. To fix this "lack" it was implemented a set of libraries specialized in mathematical stuffs.`Numpy` and its extension `Scipy` permits to glue together the ease and the power of Python with a speed that in some cases can equal o beat Fortran and C. `Numpy&Co` are build on a C/Cython/Fortran core but following the "Python model".

Portability

Python code runs on a virtual machine, quite like Java, so that it can run on every architecture for which a Python interpreter and the libraries are available. On the machines without the necessary environment we had to manually compile and install the missing libraries.

Parallel

The first version used `mpi4py` to parallelize the recursive traverse function using "masks" to select which of the processes to activate. In practice with every call of the function it was passed a list of processes ranks (the "mask"). At every recursive call the list is distributed and every piece of it goes with a function call. When the number of ranks in the list are less than the number of call the list is regenerated. This solution was abandoned because it was very complicated and difficult to handle. More, it imply a lot of message passing so it was no convenient.

Therefore the code was developed as a serial code and optimized without parallel capabilities but with the embarrassing parallel goal in mind. The code is designed like a patchwork of function, i.e. the "main" call serially call the functions one after another, and the "main" is a function itself, capable of receive arguments like a function but also from the command line. Hence it is possible to call the code from a script many times on different dataset but also to use the main as a function in a "Parallel MapReduce" contest or to parallelize the independent calls inside it, without the need of a massive message passing.

The final implementation of the code is used from a parallel starter that create n processes with the `multiprocessing` library. Each process run a loop that take the number of the data package to analyze from a shared queue and start the

code on it. When one process finished the process start another process, until the queue is empty.

MPI

MPI is an API specification developed to permits the communication between processes running in parallel with non-shared memory. There are various implementations, both open source and proprietary. The MPI specifications permits the exchange of data one-to-one, one-to-many, one-to-all (broadcasting), all-to-one (reduce). There are possible both blocking or non-blocking calls (a blocking call let the process waiting for the communication to finish) and it is possible to synchronize the different processes, that is every processes are stopped until every process reaches the synchronization point. MPI is not the only possibility to develop a parallel code. Another possibility is, for example, OpenMP. It is an API that supports the shared memory programming, for example on multi-core processors.

Cloud computing

One of the possibilities to use the code developed for this work is in a cloud architecture. The term cloud computing refers to the technology that permits to use computing resources that are distributed and usually virtualized. Examples are the Amazon EC2 service, or the Google Apps. In particular, a possibility was to use the Picloud service, a Python cloud computing service based on Amazon EC2. It permits to transfer the compute work on the Amazon EC2 cloud through the use of some libraries provided.

6.2 Hardware

The code developed was run on different machines with different architecture. In this case the use of python permitted to leave the code unchanged because it runs on every architecture on which the python interpreter and the libraries are available. The disadvantages are the need to recompile the Fortran module that compute the distances and the installation of the interpreter and of the libraries.

For machines with Intel processors the EPD distribution permits the immediate installation of all the needed.

Now we present a brief view of the hardware used.

Leonardo

Leonardo is the last of the machines of Theoretical Cosmology group of Padua. It was the storage machine and it is about 8 years old, so it is not so fast. It is a Intel Xeon quad-core at 2.40 GHz with 2 GB of RAM. The OS is an Ubuntu Server 10.04.

Monster

Monster is the Department of Astronomy cluster, constituted of

- 12 HP BladeServer BL640c with two Intel Xeon e5430 quad-core @2.66 GHz, 32 GB of RAM and 146 GB of hard disk storage
- 1 HP BladeServer BL680c with two Intel Xeon e7330 eight-core @2.40 GHz, 64 GB of RAM and 146 GB of hard disk storage
- a HP DL180 management and storage node with one Intel Xeon e5430 quad-core @2.66 GHz, 16 GB of RAM and about 1 TB of hard disks storage
- about 4 TB of hard disks storage for the data

The OS is Rocks, a CentOS cluster distribution, and the jobs are managed with the resource and queue manager TORQUE (Terascale Open Source and QUeue manager) and with the job scheduler MAUI.It was not used because some problems with the recompilation of the Fortran module.

Pico

Pico is the new Theoretical Cosmology group computing server. It has two Intel Xeon L5640 @2.27 GHz six-core with HyperThreading (i.e. we have 24 virtual CPUs), 32 GB of RAM and 1 TB of hard disk storage. The OS is an Ubuntu server 10.04.

SP7

It is a IBM Power7 cluster hosted at DEI (Department of Information Engineering of Padua), with 192 Power7 CPUs @ 3.1 GHz. It has 600 GB of RAM splitted between the two virtual machines available. The OS are AIX6.1 and Suse Linux Enterprise (SLES). Here we had some problems installing the needed libraries, luckily resolved by the system administrator.

SP6

Is is the Power6 system accessible at Cineca. Unfortunately it was not possible run the code on this because of the difficulties to recompile some libraries on the AIX OS.

Linus

A personal laptop, a EEEPC 1000HE, AtomN280 with 1 GB of RAM. The OS is Ubuntu 10.10.

Uno

A personal laptop, a Dell Latitude E6410, Intel Core i7M 640 @ 2.80GHz dual-core with HyperThreading and 8 GB of RAM. The OS is Ubuntu 10.10.

6.3 Developing the code

The original code used as base for this work is the *kd*-tree code available in the "spatial" set of functions provided by Scipy, whose source can be retrieved at `https://github.com/scipy/scipy/blob/master/scipy/spatial/kdtree.py` .

Some of the techniques described in Chapter 5 was yet implemented in the original code. It was able to handle multiple radii simultaneously (Section 5.3) and to save computation excluding both the nodes which distance is too small or to big that none of the radii under investigation is to be taken into consideration (Section 5.3). If some radii entirely include a node, the code could directly add the total number of couples instead of opening it (Section 5.3).The main

additions are the capability to save computation pruning redundant node check in case of auto-correlation (Section 5.3) using tags for each node and checking for those before the recursive call, the minimum radius for correlation, the collection of statistics during the computation and some tests on what is the better strategy to open the leaves, removing the square roots and to convert to Fortran the distances calculations.To better understand this it is useful to know how the *kd*-tree code works.

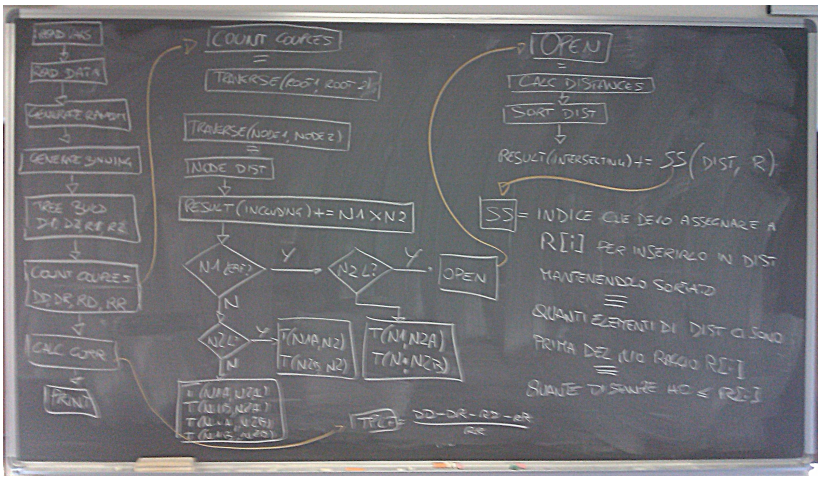

Figure 6.1: Scheme of the serial part of a previous version of the code on a blackboard.

Building the tree

In the first step a *kd*-tree is built from a $n \times 3$ array containing the positions of the points. The tree is a python object containing some attributes and methods, and two sub-objects: the inner nodes and the leaves. The building routine at each step checks the number of datapoints, and if they are more than the leafsize, it checks for the dimension (x, y or z) with the maximum spatial range and set the split point in the middle of that range. This is not optimal, because splitting at the media lead to a better balancing of the tree, but this choice give a faster code with a quite well balanced tree. The dataset is now divided respect to the split point, an inner node is created and the building routine is recursive called on the two subsets. When the number of points is less than the leafsize a leaf is created.

Every node contain some information about the number of points contained, the tag of the node, its memory size and in case of an innernode, the pointers to the sub-nodes.

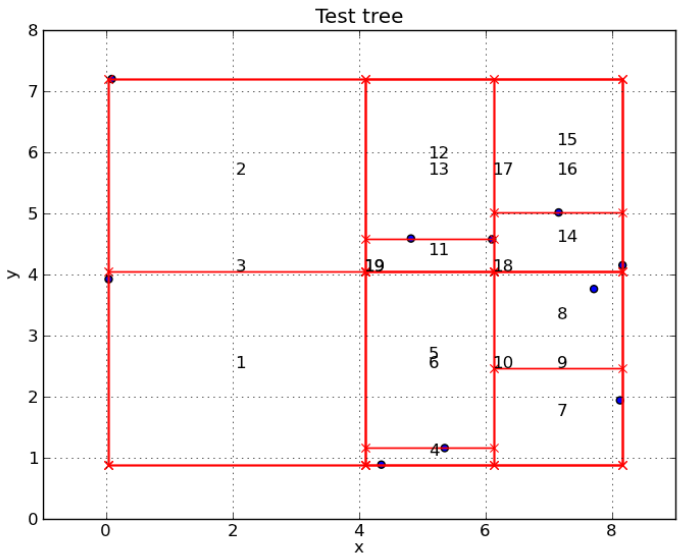

Figure 6.2: A 2D example with few particles of how the tree is built. The red square are the nodes boundaries, both for leafnode and innernode, and in the center there is the tag that uniquely identify the node in the tree. The nodes are built to optimize the distribution of the particles, for example by splitting in two halves the dimension with bigger spatial range.

The counting method

The counting routine is a method of the tree object and takes another tree as argument. Starting from the biggest nodes that are the entire trees, at every step it takes as arguments two nodes, one for each tree, and retrieve the maximum and minimum distance not between the two nodes (i.e. the maximum and minimum distance between the boundaries of the nodes). These two quantities are then compared with the array containing the radii for which to compute the correlation.

Three cases are possible:

- some radii are greater than the maximum distance between the nodes: for these radii the total number of possible couples (including all the sub-trees inside the nodes) are added to the corresponding cells in the array containing the counts as function of the radius and then these radii are dropped
- some radii are shorter than the minimum distance: this radii are dropped
- some radii intersect the nodes: for these radii the nodes are opened

Now we have some radii for which the two nodes must be opened. If the two nodes are inner nodes, each node is divided and the counting routine is recursively called on any combination of the sub-nodes. If instead the two nodes are leaves, the code compute all the possible distances between the nodes (taking care of saving computation in the self-correlation case). Then the distances are sorted and using `numpy.searchsorted()` the code calculates for each radius how many couples are separated by a distance shorter than that radius and add them to the result array.

Counting strategies

In order to achieve better performance we try different approaches to open the leaves. I particular, the differences between the strategies tested were if to maintain or not the square roots, if to use a linear or logarithmic scale for the radii and if to use the sort or another method for counting the distances. The winning strategy is to drop the square roots, to use a linear scale (but, the radii are logarithmically equally spaced) and to maintain the distance counting using `numpy.sort()` and `numpy.searchsorted`.

Searchsort and alternate distance counting

The original version of the code calculates the distances between all the possible couples between the two nodes and sort the resulting array. Then, the `numpy.searchsorted` function identifies the index where to insert each radius to maintain the array ordered, that is the number of distances shorter than the radius. The other way consists in to divide (using an integer division) the distances

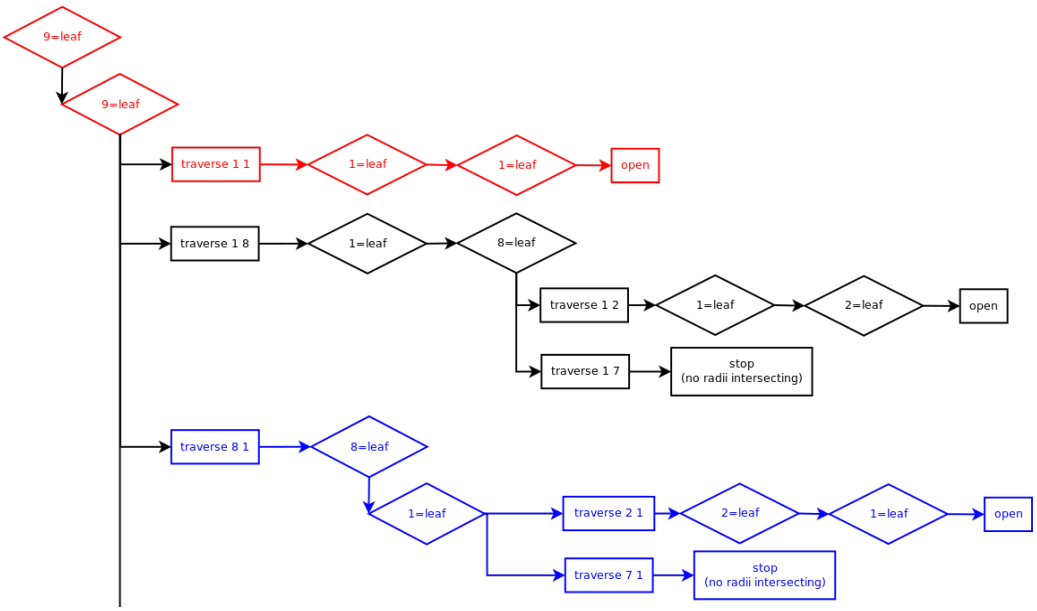

Figure 6.3: A simplified example of how an autocorrelation traverse works. This is the beginning of a traverse. In this case the two trees are traversed up-down, starting from the root node. The rhombs represents an `if` statement, for example when the code check if a node is a leaf. The square are the action to be taken, e.g. start a traverse or stop. The red branches are the non redundant actions, the black and the blue are the redundant actions that the code recognize and perform only once.

array by the bin length (the bins must be all equals): the result represents how many times the distance is greater than the bin spacing, thus it is the index of the bin to which we have to add 1.

Distances

The two most time consuming part of the code are traversing the tree and open the leaves. In both cases is necessary to calculate distances, and in fact timing the code shows that most of the time is spent in the routine. After some tests the solution was to create an ad-hoc Fortran library containing few lines that

compute the distances in a brute force way. This was included as a module using `f2py`. The Fortran solution was thousands times faster than the Python one.

Goodies

Some other things was added to the code. They were not essential to the computation but their presence eased our lives. First we add a lot of statistic during the execution, such as where the execution is at any time or the characteristic of the data: number of particles, tree dimensions, ... In particular to compute the dimension in memory of the tree is not trivial. In Python every "variable" is a link between the name and object in memory and an array is a link to the array object, that contain the references to every object in the cells. In this view, the dimension of a an array is only the size in memory of the "array object" but it does not include the size of elements in the array. To obtain the "real" size of the array it is necessary to recursive scan the parts of the array object and to multiply the dimension of a cell by the number of cells. To obtain the size of a tree is something similar, we had to scan the entire structure of the tree and consider the sub-objects and the elements contained. The code was also fixed to permits the persistence of the trees because the modules that permits to save the Python objects can not handle nested object, such as the nodes of the tree. This was done updating the module dictionary that contains the list of the object and the namespaces references. Some checkpoints were added to the code to avoid the analysis run on corrupted data.

Another useful feature was to embed all the logging needs in a logging structure provided by the `logging` module. In this way we had a powerful logging engine that permits a coherent handle of all the messages, together with some useful information about the the generation of the log message, e.g. where and in which module the log message was generated, at what time, and so on.

Sometimes we take advantage of the `tryexcept` Python feature to permits a clean exit from the code execution in case of problems, handling warning messages and saving the data before the stop. This, together with the Python error handling, lead to an easy debug and to the total absence of nasty errors like "segmentation fault" typical of the C/C++/Fortran programming.

Leafsize optimization

To optimize the code we simulate a complete traverse of the trees built with different leafsizes recording the time needed to a full traverse without opening any leaves and multiply the number of virtually opened leaves by the time needed to open leaves of different sizes. The result are shown in figure.

Timing the code results also in a rough model for the time dependence. We find these approximate dependence:

$$t_{trav} \propto N_{leaf}^{-1.66} \tag{6.1}$$

$$t_{dist} \propto N_{leaf}^{2} \tag{6.2}$$

$$t_{trav} \propto N_{tot}^{1.77} \tag{6.3}$$

$$t_{dist} \propto N_{leaf}^{2} \tag{6.4}$$

$$trav_{ratio} = n_{trav} \sim 0.1\% - 0.01\% \tag{6.5}$$

where t_{trav} is the time needed for traverse the entire tree, t_{dist} the time to calculate all the distances between two leaves and the other quantities the leafsize and the total number of points.
The result is:

$$t_{tot} = n_{trav}(n_0, l_0)\left\{t_{0,\,trav}(n_0,\, l_0)\left[\left(\frac{N_{tot}}{n_0}\right)^{1.77}\left(\frac{N_{leaf}}{l_0}\right)^{-1.66}\right] + t_{0,\,dist}\left[\frac{N_{tot}}{n_0}\left(\frac{N_{leaf}}{l_0}\right)^{2}\right]\right\} \tag{6.6}$$

It is not precise, also because the time depend strongly on the partial distribution of the point.

Periodic boundary condition

There are some considerations to be done handling the border of the simulation box. Out of the border, clearly, the counts drops. Doing the simulations usually the so called “periodic boundary conditions” are used, that is, one side of the box is linked to the opposite side. Analyzing the data one can use the same technique, or, in a simpler way, approximate assigning a lower weight to the counts near the border. In this work we prefer assuming that the division by the random counts (also affected by the same problem) suppress the problem.

Self-correlation

As presented before, the tree built with the modified code add to the nodes some properties and one of these is a unique *tag*, that is a number that identify the node. In case of auto-correlation there are two things to consider: the comparison between different nodes two times and the comparison between the same node in the two trees.

In the first case, choosing to consider only the nodes where `node1.tag < node2.tag` solve the problem. In the second case the solution is simple again. For the radii that entirely include the nodes it is sufficient to add `node1.children*(node2.children-1)/2` to the radii counts, for the opened leaves it is enough to half the result.

Data formats and persistence

The original data were in unformatted Fortran binary format, so it was necessary to try to understand how they were written using the original Fortran I/O routines or some informations provided by the author of the simulations. To manage these data was create some Python functions. The original data was also sorted, splitted and indexed to improve the usability and the performance of the code (memory vs CPU usage and so on).After some tests about the different possibilities (pickle, zoobd, neodb, ascii, python binary files, ...) we chose the HDF5 format to convert the original data and to store possible outputs. We also create a routine capable of creating the tree using directly the file hierarchy, but for the moment it was too slow to be used. One possible solution is to work on the HDF5 library (PyTables) cache. For now we chose to not save the trees because the time needed to build them is quite low.

Libraries

All the code, except for the small Fortran routine to compute the distances, is written in Python using `numpy` for the numerical part and `PyTables` for the I/O. For simplicity and optimization we adopted the Enthought Python Distribution (EPD), which provide all what we need yet optimized with libraries such as atlas, mkl and so on. The EPD packages are compiled for Intel's hardware and

for Linux, MacOSX and Windows, so it was necessary to compile the needed packages to run on a Linux Power7 cluster. Untill now we didn't find any way to run on AIX.

CHAPTER 7

RESULTS

7.1 GIF galaxy-galaxy auto-correlation

The first test we made to check the code was the computation of the galaxy-galaxy correlation on the GIF data. The data was downloaded from the GIF site `http://www.mpa-garching.mpg.de/Virgo/data_download.html` and the original `ASCII` data were converted to the `HDF5` format to be compatible with the code and the other data. The distance range we sample was between 10 kpc/h and 10 Mpc/h. The lower limit is because of the softening length, the upper limit was enough to sample the 2H term. It took about 16 seconds to compute the auto-correlation between 15445 galaxy centers on Uno. Because of the geometry and clustering properties of the dataset we chose to use the same number of random as the data particles. There was no need to sample very small scales. This particular data was preferred because of the small computational effort needed and the big literature comparison available. A χ^2 test on the result shows that our result is compatible with the literature. We can compare these results with the results obtained by Diaferio et al. presented in figure 7.2. We can see that the results of our code are in good agreement with the Diaferio, White, and Kauffmann 1999 results. We would expect a slope of about -1.8 and we find -1.7 with an unweighted χ^2 fit.

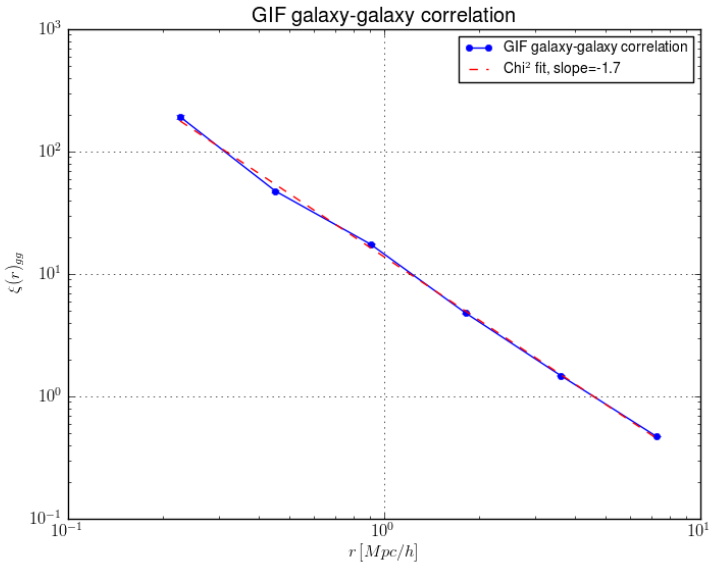

Figure 7.1: GIF galaxy-galaxy correlation.

7.2 GIF2 matter-matter auto-correlation

Because of the success of the first test we decided to test the code on the matter-matter two point correlation function. In this case we chose the GIF2 simulation data to have better mass and force resolution. The GIF2 particles data were already available on our servers in Fortran binary files. We convert the original data in a set of 22 `HDF5` files containing the particles positions sorted on the x coordinate. Every file contains 5 Mpc/h in x and about 2.5×10^6 particles. The computation was performed in parallel using 22 CPUs on Pico and correlating each file with itself and the two following to reach the 10 Mpc/h of distance for the correlation. The minimum distance for the correlation was fixed at 10 kpc/h because of the softening distance of 6.6 kpc/h. The computation needed 63 processes and a total CPU time of 88 days, with a minimum of about 10 hours and a maximum of 120 hours for a single process. In figure 7.3 is shown the results in a logarithmic scale (the last two distance bin are not visible because they have negative values.). In this case we compare our results with the results obtained by Hayashi and White in Hayashi and White 2008 in figure 7.4 on the

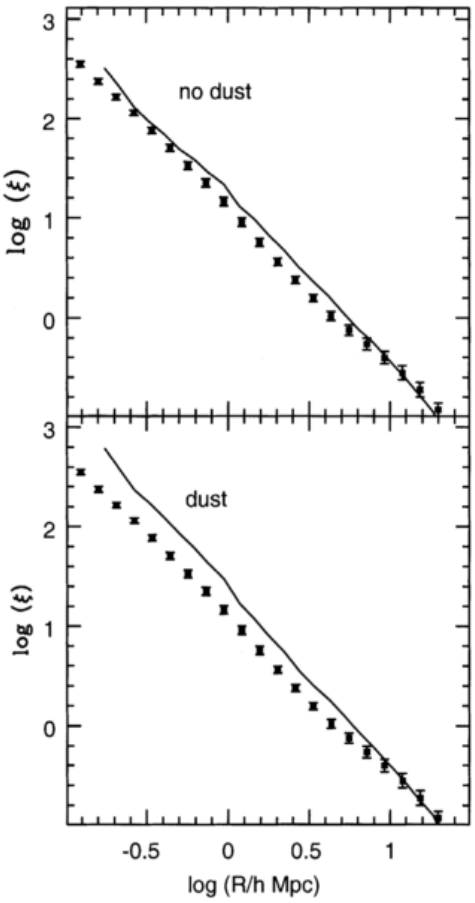

Figure 7.2: Two point correlation function for galaxies by Diaferio, White, and Kauffmann 1999.

Millennium data. The match is good but not perfect due to the different data and parameter and resolution of the simulations.

The dark matter auto-correlation in figure 7.3 was the first test performed to check the code. The result for the GIF2 data was compared with the results by Hayashi and White 2008 on the Millennium 7.4. The differences on small scales depend on the different smoothing length.

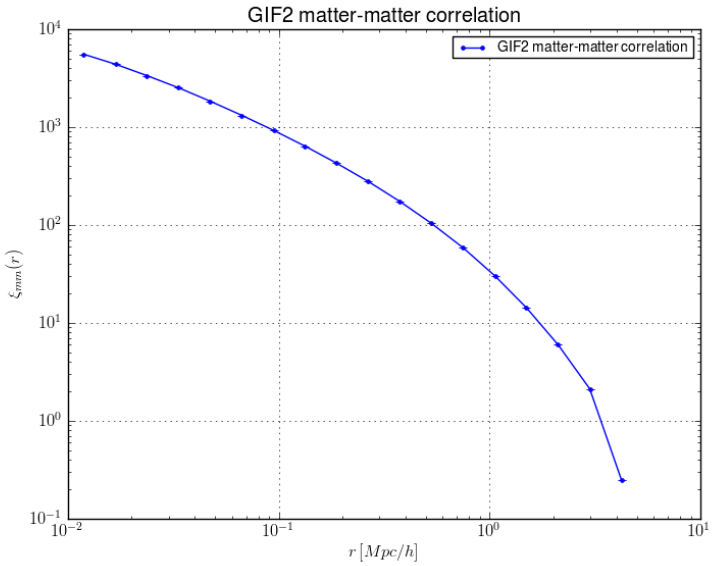

Figure 7.3: GIF2 matter-matter correlation.

7.3 GIF2 halo-matter cross-correlation

The next test in preparation to the MII analysis was to check the routine on something similar to what we would do on the MII so we computed the halo-matter cross-correlation on the GIF2 data.The halo-matter correlation, as presented in Chapter 2, has a shape dominated by a NFW-like halo profile on small scales, averaged over all the haloes in the simulations, and by the linear bias with the DM auto-correlation on the bigger scales. The transition between the two regimes happens around the virial radius R_{vir}, that can be considered the radius of the halo. Because of the GIF2 softening the correlation was performed from 10 kpc/h to 10 Mpc/h to be able to check the big scale shape. On the smaller scale, from 10 kpc/h to 316.2 kpc/h, the random data was ten times more than the data to sample the halo-matter clustering properly.

The particles data was the same of the previous test, the halo centers was extract from the original files of the simulation available on our servers, sorted in x, indexed and converted in the `HDF5` format. In this way only the amount of data needed by the particles slice was load in memory improving the performance.

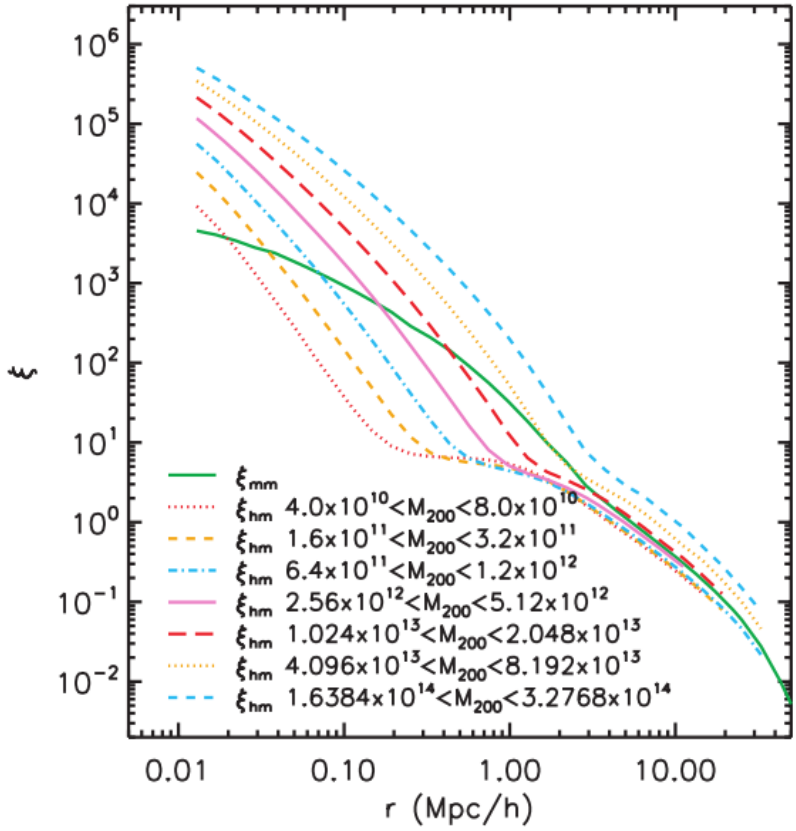

Figure 7.4: Millennium matter auto-correlation and halo-matter cross-correlation from Hayashi and White 2008.

The computation for the small scales was performed on 20 CPUs on Pico and required about 26 hours and it was splitted on 1091 processes with a minimum of 48 seconds and a maximum of 234 seconds. For the larger scales the random was as many as the data. This takes 46 hours with a minimum of 56 seconds and a maximum of 466 seconds. In this case the cross-correlation profile is obtained by virtually sitting on each halo center and counting how many matter particles there are at each radius.

The results, in figure 7.5, can be compared with the results by Hayashi and White 2008 in figure 7.4 and with the theoretical predictions by Giocoli et al. 2010 in figure 7.6.

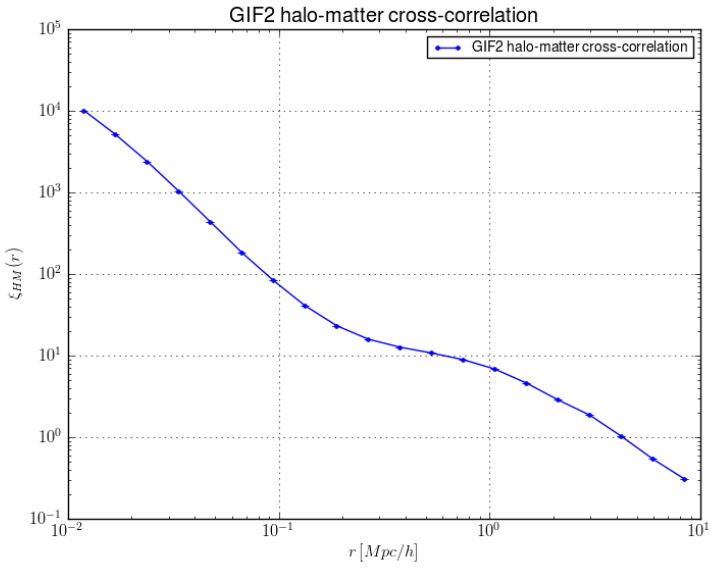

Figure 7.5: GIF2 halo-matter correlation.

7.4 GIF2 halo-matter cross-correlation in mass bins

Once checked that the code was working well calculating the halo-matter cross-correlation giving a result in good agreement with the results in literature and with the theoretical previsions the next step is to investigate the contribution of the signal from haloes of different masses. To do this we first extracted the haloes belonging to the mass bins from the dataset. The bins were centered on $\frac{M^*}{64}$, $\frac{M^*}{16}$, $\frac{M^*}{4}$, 1, $4M^*$, $16M^*$ where M^* is the mass where $M^* = 8.9 \times 10^{12} M_\odot$ and correspond to 5171 simulation particles. The bins were from $\frac{1}{\sqrt{2}}$ to $\sqrt{2}$ respect to the center. In table 7.1 it is possible to have an idea of the selection in mass.

In this case we performed the analysis with 1099 processes for each mass bin, 6548 in total, for a total time of 30 hours, a minimum of 0.5 seconds and a maximum of 104 seconds. What we expected was to see the virial radius (that is where the transition between the 1H and the 2H terms happens) to move at bigger scales increasing the mass considered. This can be seen in figure 7.7 and

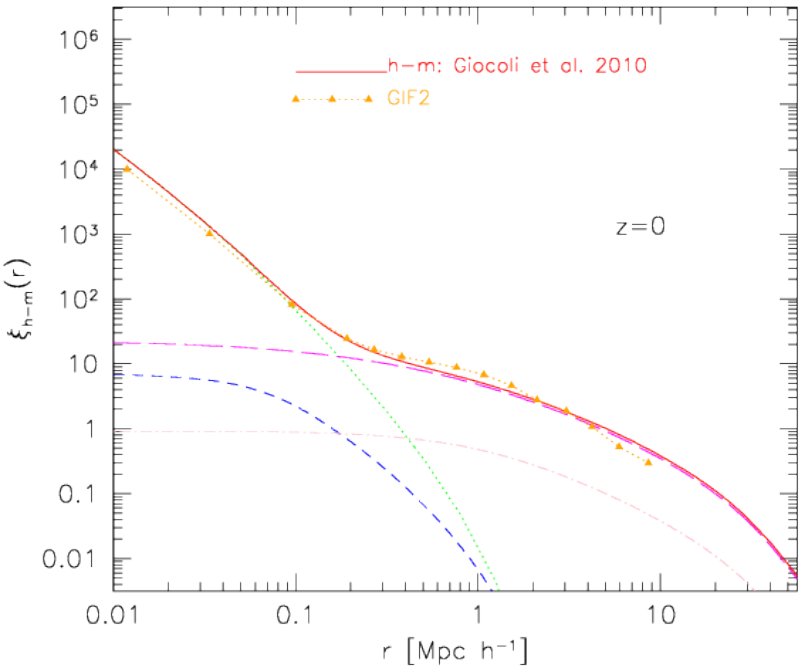

Figure 7.6: Giocoli et al. 2010 halo-matter correlation predictions. In this figure are clearly visible the contributions by the 1H and 2H terms and the sum of these two. The green line is the contribution of the correlation between the halo center with the smooth component of the halo, that is the particles belonging to the halo but not to the substructures, the blue is the correlation between the halo center and the clump component, that is the particles in substructures. The 2H term is the composition of the halo center with the smooth and clump components of other haloes. In addiction the profile calculated by our code is superposed to be easily compared.

the results can be compared with those by Hayashi and White 2008 obtained from the Millennium simulation. Another interesting analysis would be to fit each obtained profile with the NFW profile, as in Hayashi and White 2008 but due to the low resolution of the simulation and of the poor sampling of the data pairs with random pairs this test was not so significant. For example the two profiles corresponding to the higher masses do not cover the smaller scales. It

Bin center M^*	**bin center** $(M_\odot)$	**min** $(M_\odot)$	
1/64	1.4×10^{11}	1.0×10^{11}	2.0×10^{11}
1/16	5.6×10^{11}	3.0×10^{11}	7.9×10^{11}
1/4	2.2×10^{12}	1.6×10^{12}	3.1×10^{12}
1	8.9×10^{12}	6.3×10^{12}	12.6×10^{12}
4	3.6×10^{13}	2.5×10^{13}	5.1×10^{13}
16	1.6×10^{14}	1.1×10^{14}	2.3×10^{14}

Table 7.1: Mass bins for the halo-matter cross-correlation in units of M^* and $M_\odot$.

can be also noticed that lower masses haloes have steeper slope on small scale than the more massive haloes. This is because on average they are more centrally concentrated.

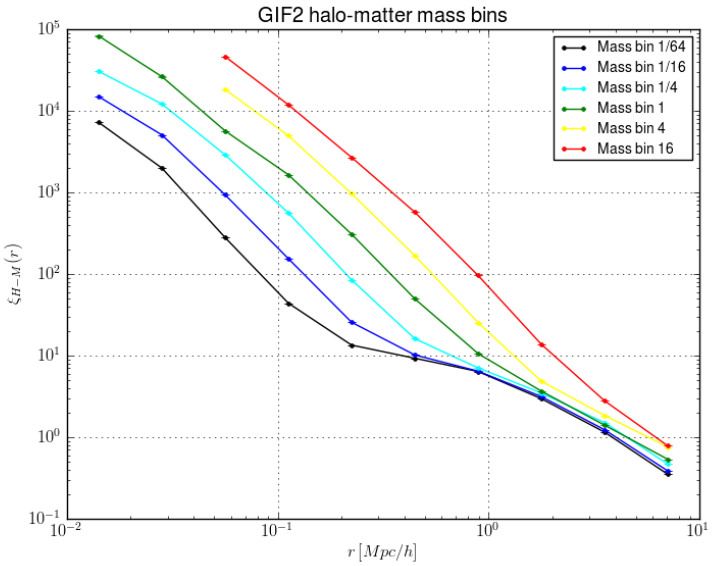

Figure 7.7: GIF2 halo-matter in mass bin correlation. in this figure can be seen how the transition between the 1H and 2H terms move to bigger scale increasing the mass of the haloes.

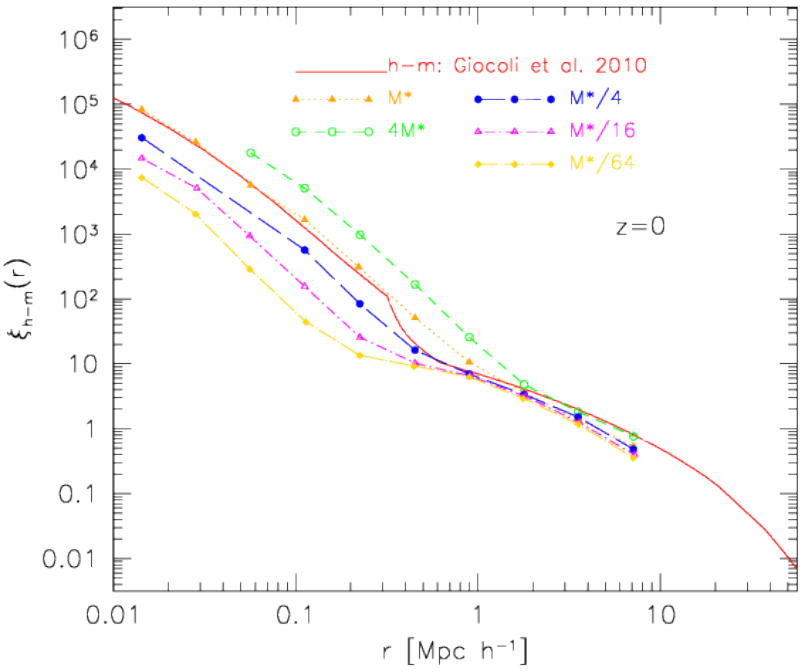

Figure 7.8: In figure you can see the GIF2 halo-matter cross-correlation per different mass bin computed by our code and the theoretical prediction by Giocoli *et al.* for a mass bin centered on M^*. The drop of the theoretical prediction at 2×10^{-1} is due to the mass cutoff in the Fourier space. A part from this the profiles are in good agreement.

7.5 GIF2 subhalo-matter cross-correlation

As in the halo-matter cross correlation the subhalo-matter cross-correlation is the composition of more than one contribution. In this case, as presented in 3.1, the 1H term is the sum of the correlation with the smooth and clump component of the same sub-halo and with the matter in other substructures. The 2H term contain the correlation with the smooth component and clump component of other haloes. To select the sub-haloes we extracted the haloes with more than 200 particles, than for each halo we retrieve the list of the contained sub-haloes.

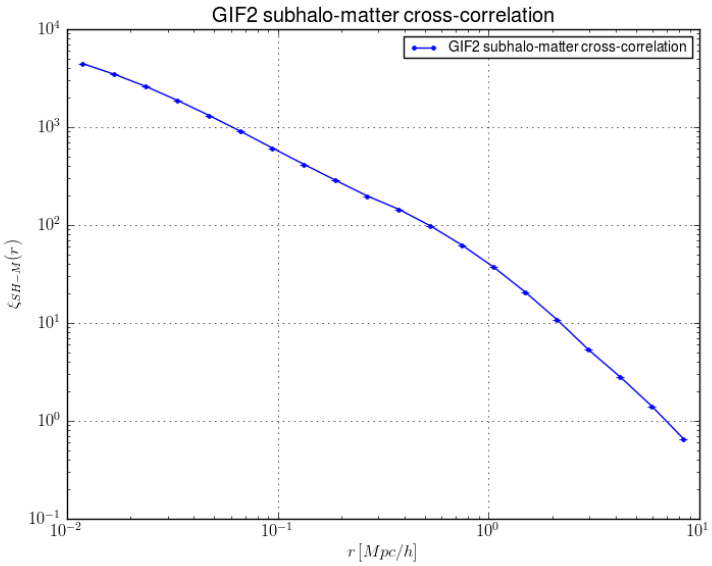

Figure 7.9: GIF2 subhalo-matter correlation.

For each sub-halo the coordinates were converted from those of the host halo center to the global ones. Some check were done to avoid problems related to the presence of void files.

It took a total CPU time of 35 hours and the computation was performed splitting it in 2184 different processes, with a minumim and maximum time needed of 23 and 272 seconds.

Subhalo-matter cross-correlation in host halo mass bins

In the same way we would computed the cross-correlation for the haloes belonging to different mass bins we were interested in investigate the signal due to the substructures hosted in haloes of different masses. On the base of the host haloes mass selection we extracted the corresponding substructures and computed the cross-correlation with all the particles.

As it was done for the halo-matter correlation we splitted the computation in many different processes, 1100, feeding many CPUs at the same time. The total CPU time is 43 hours and the minimum and the maximum are 5 and 91 seconds respectively.

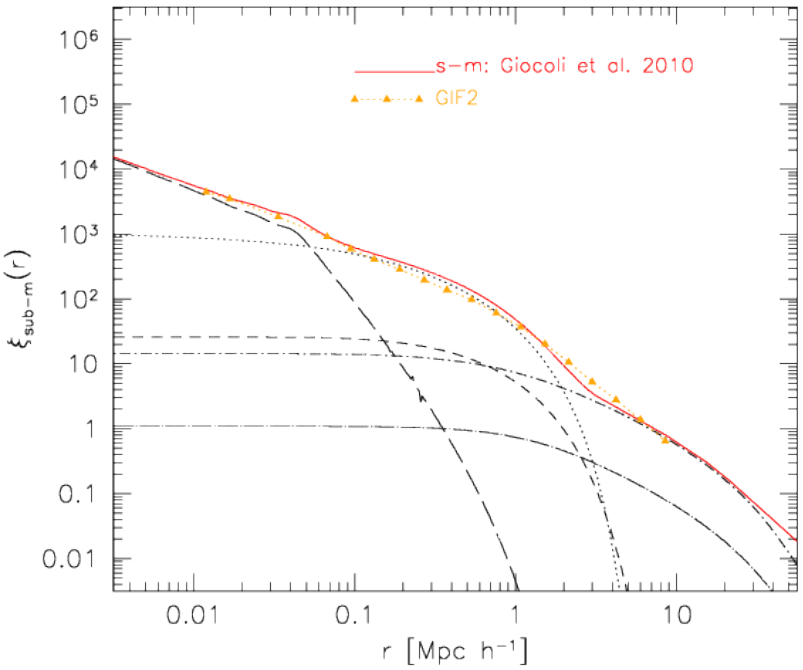

Figure 7.10: Subhalo-matter cross-correlation. The figure compares our results from the GIF2 analysis with the theoretical predictions from the model of Giocoli et al. 2010. It can be seen the good agreement between the two profiles. For the theoretical predictions are also shown the different contributions: the subhalo correlation with the subhalo matter, the smooth matter in the host halo, the matter in other subhaloes in the same host halo, the smooth and clump components of other haloes.

The result now depends on new quantities: at the smallest scales the density profile of the substructures itself; at intermediate scales the distributions of smooth matter around a sub-halo; at the largest scales the biased distribution of sub-haloes compared to matter. What we obtained shows some interesting features but it is limited by the low resolution at small scales but it is difficult to interpret. At small scales the sub-haloes hosted in smaller haloes present a steeper profile, probably due to their dynamical history inside the halo. At intermediate scales we can see that sub-haloes hosted in larger haloes live in denser

regions, because they are located, on average, at smaller fraction of the host halo virial radius. At the largest scales we observe the subhalos-matter biased distribution.

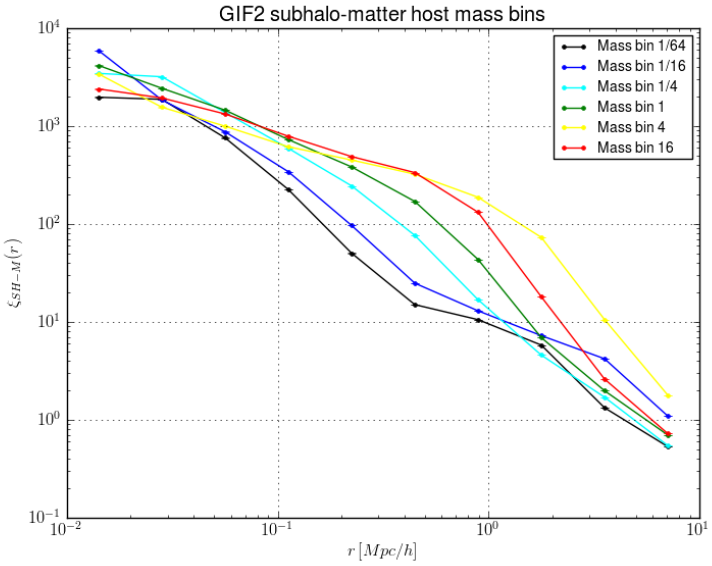

Figure 7.11: GIF2 subhalo-matter in mass bin correlation.

7.6 Bias

We can consider the *bias* as an indicator of how two different distribution we are considering differs one from the other. It also tells us how one distribution can well represents the other. We have, in the previous chapter, shortly mention that the large scale halo-matter correlation follows, biased, the matter-matter correlation profile. If we divide the halo-matter profile by the matter-matter one on large scale we have a an idea of how well the halo-matter profile represents the matter-matter one. Our results were compared with those in Sheth and Tormen 1999 and there is no a perfect agreement as you can see in table 7.2. Because our profiles are in good agreement with those in literature we think that a possible explanation can be the scale we sampled, 3.5 Mpc/h because of the data available. It is possible that the same operation on bigger scales would yield a better

result. We checked our profiles for each mass bin we analyze in the previous sections. Another possibility is to divide the entire profiles as presented in Mo and White 1996. In this way we obtain a more general idea of the bias between the two distribution, that is a complete view of how the halo-matter distribution differ on all scales to the matter-matter one. The results can be seen in figure 7.12 and can be compared with the results from Mo and White 1996 presented in figure 7.13. Our results are quite different respect to those by Mo *et al.* because their simulations did not sample the small scales. We have much more resolution on these scales so our bias profile present is modified by the presence of the halo profile. Moreover, they investigate scales larger than the ours, so in our profiles we do not reach the scale where the shape is flat but only where the profile grows.

M∗	**M$_\odot$**	**data b (M)**	**lit. b (M)**
1/64	1.4×10^{11}	1.25	0.32
1/16	5.6×10^{11}	1.34	0.4
1/4	2.2×10^{12}	1.64	0.6
1	8.9×10^{12}	1.53	1
4	3.6×10^{13}	2.01	1.1
16	1.6×10^{14}	3.04	–

Table 7.2: Comparisons between the bias measured from the results at r = 3.5 Mpc/h and the bias from Sheth and Tormen 1999. The values are not in agreement and this can be due to because our profile does not represent the bias on such scale but it is necessary to sample bigger scales.

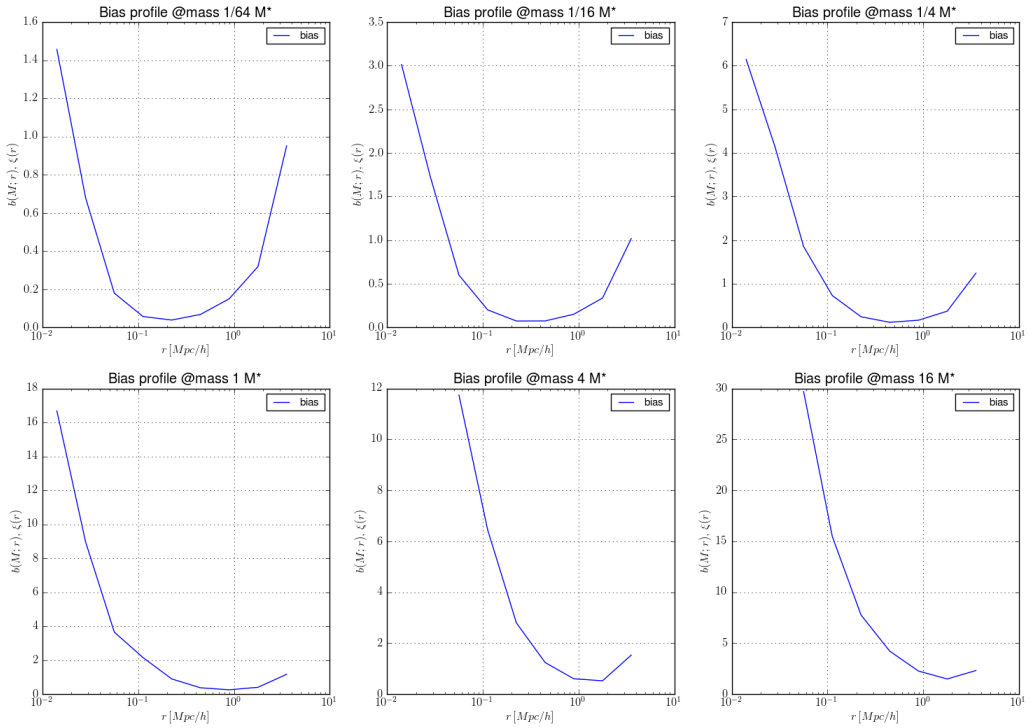

Figure 7.12: Gif2: bias profiles obtained dividing the halo-matter cross-correlation in mass bins by the matter-matter auto-correlation as in Mo and White 1996.

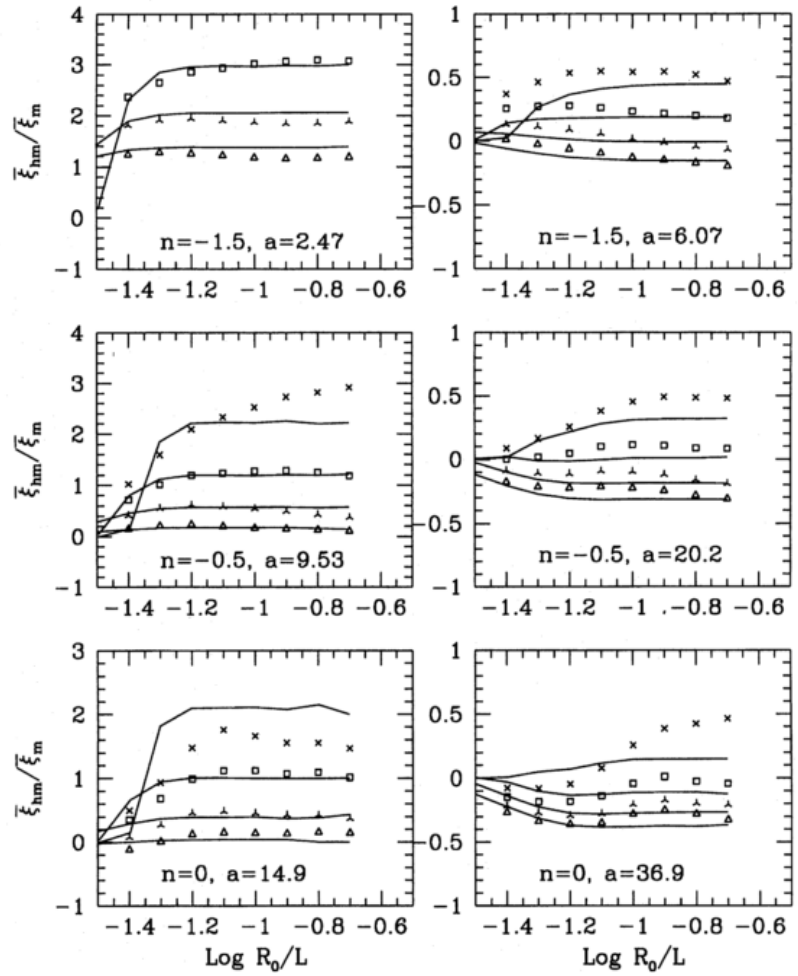

Figure 7.13: The halo-matter correlation function profile divided by the matter-matter one as function of the normalized radius, from Mo and White 1996.

CHAPTER 8

CONCLUSIONS

The goal of this work was to investigate the cross-correlation between sub-haloes and dark matter using the Millennium II simulation data. This goal was not achieved because of some technical problems. First we had some problems to obtain the data: they arrived on an hard disk from MPA in Garching but due to a failure in the first hard disk we had to wait for a second shipping. Moreover the Millennium II data was in a "private" Fortran binary format and we can only access the particles positions from the hard disk and the FOF centers from the online database, but we were unable to reconstruct the information about the sub-haloes centers. We also had some problems with our servers and to gain access to some CPU time on other machines. Some of the machines furthermore had some compatibility problems with the code so we had to spend a lot of time on this.

Despite this problems we successfully tested the code on the GIF and GIF2 code and obtained the results presented in the previous chapters. Some test slices of the Millennium II are now under analysis and we will start with the subhalo-matter cross-correlation as soon as possible.

8.1 Further works

The code is now fully working and its results tested. There are however some possible improvements. First we have to do is to clean the code and to let it be more general. The second step is about optimization. Now the code has good times but it is possible that with some changes the times will be smaller. Some possible improvements are to rewrite some parts in Chyton or C/C++/Fortran, but we thinks that this way is not so promising. In the future we will try to convert some part of the calculation to be run on GPUs and may be some further modifications to the tree code such as more statistics on the nodes and less on the fly work will give us better execution times.

APPENDIX A

BETTER BUGS

A.1 array2int

With the introduction of the Fortran version of the distance calculation the output of the distance between two nodes became a one-element array (instead of the float obtained from the previous, Python, version). Thinking about the previous conversion from the configuration file parsing I converted the output of the Fortran module to `int` instead of converting it to `float`. Converting the data coordinates in kpc/h give the correct results.The result was that for a defined `leafsize`, depending on the total number of particles, the differential data counts dropped at little radii.What actually happened was that the distances between nodes were truncated to `int` and so they were underestimated. In this way little radii that should be excluded or at least should open leaves included all the leaves, leading to higher cumulative counts. The subtraction of the pair of adjacent counts eventually gave lower differential counts. Random counts, data in kiloparsecs or bigger leaf give a smaller effect that was invisible, the same happened for bigger radii, with higher counts.

A.2 .sort instead of .sort()

Two trivial parenthesis forgotten in calling the .sort() method of the array containing the distances return the memory address of the method instead of sorting the array, according to the Python syntax. In this way no error was raised but the counting for opened leaves was wrong.

A.3 Random population

In case of self-correlation we thought that use two set of random with only one set of data will improve the error by overestimate the random. Bad idea!! This choice let the code counts $\frac{2n}{(n-1)} \sim 2$ times the random couples. To fix this one can simply use the same number of sets both for the data and for the random or divide the final counts. The first solution save a lot of computation.

BIBLIOGRAPHY

[1] Sunil Arya et al. “An optimal algorithm for approximate nearest neighbor searching fixed dimensions”. In: *Journal of the ACM* 45.6 (1998), pp. 891–923. ISSN: 00045411. DOI: 10.1145/293347.293348. URL: http://portal.acm.org/citation.cfm?doid=293347.293348.

[2] JR Bond et al. “Excursion set mass functions for hierarchical Gaussian fluctuations”. In: *The Astrophysical Journal* 379 (1991), pp. 440–460. URL: http://adsabs.harvard.edu/full/1991ApJ...379..440B.

[3] M. Boylan-Kolchin et al. “Resolving cosmic structure formation with the Millennium-II Simulation”. In: *Monthly Notices of the Royal Astronomical Society* 398.3 (2009), pp. 1150–1164. ISSN: 1365-2966. eprint: arXiv:0903.3041v2. URL: http://onlinelibrary.wiley.com/doi/10.1111/j.1365-2966.2009.15191.x/full.

[4] a Cooray and R Sheth. “Halo models of large scale structure”. In: *Physics Reports* 372.1 (Dec. 2002), pp. 1–129. ISSN: 03701573. DOI: 10.1016/S0370-1573(02)00276-4. URL: http://linkinghub.elsevier.com/retrieve/pii/S0370157302002764.

[5] Antonaldo Diaferio, Simon D M White, and Guinevere Kauffmann. “Clustering of galaxies in a hierarchical universe I. Methods and results at z 0 ”. In: *Monthly Notices of the Royal Astronomical Society* 206 (1999), pp. 188–206.

[6] R A Finkel and J L Bentley. “Quad trees a data structure for retrieval on composite keys”. In: *Acta Informatica* 4.1 (1974), pp. 1–9. ISSN: 00015903. DOI: 10.1007/BF00288933. URL: http://www.springerlink.com/index/10.1007/BF00288933.

[7] Jerome H. Freidman, Jon Louis Bentley, and Raphael Ari Finkel. "An Algorithm for Finding Best Matches in Logarithmic Expected Time". In: *ACM Transactions on Mathematical Software* 3.3 (Sept. 1977), pp. 209–226. ISSN: 00983500. DOI: 10.1145/355744.355745. URL: http://portal.acm.org/citation.cfm?doid=355744.355745.

[8] L Gao, SDM White, and A Jenkins. "The subhalo populations of ΛCDM dark haloes". In: *Monthly Notices of the* 000.February (2004). eprint: 0404589v3. URL: http://onlinelibrary.wiley.com/doi/10.1111/j.1365-2966.2004.08360.x/full.

[9] Carlo Giocoli et al. "Halo model description of the non-linear dark matter power spectrum at k 1 Mpcˆ-1". In: *Monthly Notices of the Royal Astronomical Society* 15.March (2010), pp. 1–15. URL: http://arxiv.org/abs/1003.4740.

[10] A.G. Gray and A.W. Moore. "N-Body'problems in statistical learning". In: *Advances in neural information processing systems* (2001), pp. 521–527. ISSN: 1049-5258. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.7138\&rep=rep1\&type=pdf.

[11] Eric Hayashi and Simon D. M. White. "Understanding the halo-mass and galaxy-mass cross-correlation functions". In: *Monthly Notices of the Royal Astronomical Society* 388.1 (July 2008), pp. 2–14. ISSN: 00358711. DOI: 10.1111/j.1365-2966.2008.13371.x. URL: http://blackwell-synergy.com/doi/abs/10.1111/j.1365-2966.2008.13371.x.

[12] G. Karniadakis and R.M. Kirby. *Parallel scientific computing in C++ and MPI*. Cambridge University Press Cambridge, UK, 2003. URL: http://embedded.ufcg.edu.br/~ivocalado/ebooks/mpi/CambridgeUniversityPress-ParallelScientificComputinginC++andMpi.pdf.

[13] M. Kerscher, I. Szapudi, and A.S. Szalay. "A comparison of estimators for the two-point correlation function". In: *The Astrophysical Journal Letters* 535.1994 (2000), p. L13. URL: http://iopscience.iop.org/1538-4357/535/1/L13.

[14] Antoine Labatie et al. “Uncertainty in 2-point correlation function estimators and BAO detection in SDSS DR7”. In: *Arxiv preprint arXiv:1009.1232* Umr 7158 (2010). eprint: arXiv:1009.1232v1. URL: http://arxiv.org/abs/1009.1232.

[15] S.D. Landy and A.S. Szalay. “Bias and variance of angular correlation functions”. In: *The Astrophysical Journal* 412 (1993), pp. 64–71. ISSN: 0004-637X. URL: http://adsabs.harvard.edu/full/1993ApJ...412...64L.

[16] Gerard Lemson. “and Galaxy Formation Histories from the Millennium Simulation: Public release of a VO-oriented and SQL-queryable database for studying the evolution of galaxies in”. In: *Arxiv preprint astro-ph/0608019* (2006), pp. 1–7. eprint: 0608019v2. URL: http://arxiv.org/abs/astro-ph/0608019.

[17] HJ Mo and SDM White. “An analytic model for the spatial clustering of dark matter haloes”. In: *Monthly Notices of the Royal Astronomical Society* 282 (1996), pp. 347–361. ISSN: 0035-8711. URL: http://adsabs.harvard.edu/full/1996MNRAS.282..347M.

[18] P. J. E. Peebles. *The large-scale structure of the universe*. Ed. by Peebles, P. J. E. 1980.

[19] P. Ram et al. “Linear time algorithms for pairwise statistical problems”. In: *Advances in Neural Information Processing Systems* 23 (2009), pp. 1–9. URL: http://www.cc.gatech.edu/~dongryel/NIPS2009_0436.pdf.

[20] Ravi K Sheth and Giuseppe Tormen. “Large-scale bias and the peak background split”. In: *Monthly Notices of the Royal Astronomical Society* 126 (1999).

[21] Volker Springel. “The cosmological simulation code gadget-2”. In: *Monthly Notices of the Royal Astronomical Society* 364.4 (Dec. 2005), pp. 1105–1134. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2005.09655.x. URL: http://www.blackwell-synergy.com/doi/abs/10.1111/j.1365-2966.2005.09655.x.

[22] Volker Springel et al. “Simulating the joint evolution of quasars, galaxies and their large-scale distribution”. In: *Nature* 435.June (Apr. 2005), p. 42. DOI: 10.1038/nature03597. eprint: 0504097. URL: http://arxiv.org/abs/astro-ph/0504097.

[23] I. Szapudi and A.S. Szalay. “A new class of estimators for the N-point correlations”. In: *The Astrophysical Journal Letters* 494 (1998), p. L41. URL: http://iopscience.iop.org/1538-4357/494/1/L41.

[24] Giuseppe Tormen. “Formazione delle Strutture Cosmiche”. 2011.