Computational Modeling of Large-Scale Structure With Abacus

Computational Modeling of Large-Scale Structure with Abacus

A dissertation presented

by

Lehman Haley Garrison IV

to

The Department of Astronomy

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Astronomy & Astrophysics

Harvard University

Cambridge, Massachusetts

May 2019

Dissertation Advisor: Professor Daniel J. Eisenstein Lehman Haley Garrison IV

Computational Modeling of Large-Scale Structure with Abacus

Abstract

Interpreting galaxy surveys in a cosmological context requires an accurate forward model of large-scale structure. N-body simulations are the state-of-the-art tool for this but are not without their challenges. For one, they are computationally expensive, demanding large allocations of supercomputer time; for another, they are only as accurate as their discrete particle representation of dark matter allows. In my dissertation, I address both of these challenges. Abacus is a code for cosmological N-body simulations based on an exact decomposition of the near-field and far-field force, making it exceptionally accurate and fast. Using one dual-GPU node, Abacus can solve a supercomputer-sized N-body problem many times faster than other codes while retaining orders-of-magnitude higher force accuracy. We present a full description of the Abacus code, including discussion of the combination of mathematical techniques, software implementation, and commodity computer hardware that makes this possible.

Abacus's accuracy has allowed us to identify and correct non-physical effects that arise from the discrete N-body representation of dark matter. Using modified initial conditions, we can suppress discreteness errors in the late-time matter power spectrum by an order of magnitude, including effects that bias the outcome of linear perturbation theory. We have released a suite of more than 150 across 40 cosmologies simulations called "Abacus Cosmos" based upon these results. Finally, we develop a technique for maximizing the reuse of N-body simulations by perturbatively changing the background

cosmology of a simulation output. By focusing on high accuracy for small changes in cosmology, the "warped" results are indistinguishable from a full N-body realization in key analysis metrics like the galaxy power spectrum. ABACUS will be transformative for executing simulations of the size and fidelity required for analysis of surveys such as DESI, *Euclid*, and *WFIRST*.

Contents

Acknowledgments

Many theses are the result of collaborative effort—this thesis more so than most. In particular, the Abacus code that I present here is in no way the result of my sole effort. I am indebted to the authors of the base implementation of Abacus, which existed before the start of this thesis: Marc Metchnik, Phil Pinto, Doug Ferrer, and Daniel Eisenstein. It has also been a pleasure to work with Nina Maksimova on Abacus in recent years. Doug and Nina also helped build many of the computers used in this work.

I am profoundly grateful to my advisor, Daniel Eisenstein, for his patience, kindness, and sense of humor. I feel lucky to have stumbled across an exciting project like Abacus, but even luckier to have found an advisor like Daniel. I didn't realize at the outset of grad school the extent to which students can come to (unconsciously) emulate their advisors, but realizing this now only makes me more appreciative for having worked with Daniel.

The CfA community has been a wonderful environment in which to do a PhD. The grad students, especially the lunch crew of P3, have made even the most ordinary of days a joy. My officemate Jenny and I probably delayed our respective graduations by at least a year with the amount of laughter that went on in our office.

I'm forever grateful to the D&D group: Ashley, Alex, Apollo, Phil, Ryan, Josh, Harshil, Tarraneh, and Larissa, and especially our DM Ben for bringing us all together. Ben worked every week to build an amazing story, only to have it regularly derailed by his players' short attention spans. Laughing with you all every week has been a highlight of my time in grad school.

I'm also grateful for late-night gaming sessions with Joseph, Jackson, Connie, and

Edward; Kaitlyn and Ryan for weekend getaways; and Rosaria and Joel for hosting me whenever I'm in your respective cities. I'm glad to have remained close with you all despite now living far from many of you.

Finally, I am deeply grateful to my family—my parents, for encouraging me to pursue my interest in science, and Haley and Skyler, who are not just my siblings but also my friends. I'm particularly grateful to my parents for giving me the means and encouragement to pursue my interest in computers from a young age; I never thought the skills I learned building computers in high school would be useful during a cosmology PhD! But above all I am deeply grateful to my parents for a lifetime of unconditional love and support, without which I would not be the person I am today.

For my parents: I love you to the stars and beyond.

Chapter 1

Introduction

1.1 The Concordance Model of Cosmology

In the last two decades, a "concordance cosmology" has emerged that describes a wealth of extra-galactic observations with remarkable success. From the hot afterglow of the Big Bang to the cosmic web of galaxies, the ΛCDM theory has successfully unified observations across cosmic time and space with only a handful of free parameters. Despite this success, the theory invokes energy in two components—dark energy and dark matter—whose fundamental natures are yet unknown. Indeed, 95% of the energy density of the universe is thought to be "dark"—that is, having no known electromagnetic interactions—with the remaining 5% in the form of ordinary "baryonic" matter (Planck Collaboration et al. 2018).

The phenomenology of dark energy and dark matter is reasonably well understood, however. Dark energy appears close to the "cosmological constant" of General Relativity

(GR, Einstein 1917): an extra source of cosmic acceleration arising from the vacuum energy of empty space. Dark matter appears to behave as a cold, collisionless particle species whose only interactions are gravitational. Neither of these explanations are on solid theoretical footing, however. Consider dark energy: its energy density calculated under the vacuum energy model of quantum field theory is too large by some 120 orders of magnitude—a discrepancy that has been called "probably the worst theoretical prediction in the history of physics" (Hobson et al. 2006). Many alternative models of dark energy have been proposed in light of this discrepancy, and not all of them predict cosmological-constant behavior. A common form is one in which the behavior of dark energy can change with time, such as "quintessence" (Caldwell et al. 1998; see Copeland et al. 2006 for a review). Constraints on quintessence-like models are tightening, with recent results constraining deviations from a cosmological constant to about 10% at 2-sigma in the effective equation-of-state parameter w (Planck Collaboration et al. 2018). A detection of a deviation of dark energy from a cosmological constant would be a tremendous stride towards understanding its physical origin; indeed, controlling theoretical systematics from galaxy surveys at the level necessary to make a robust claim of such a detection is a major thrust of this thesis. But even in the absence of a positive detection, tightened constraints on w will narrow the theoretical landscape and provide guidance for the next generation of dark energy models (Weinberg et al. 2013).

Dark matter behaves as a cold, collisionless particle species; much like ordinary matter, it clusters gravitationally, but unlike ordinary matter, it does not radiate and cool and thus cannot form the tightly bound clumps that we call "galaxies". Instead, dark matter forms quasi-spherical, dispersion-supported "halos" via gravitational collapse and virial relaxation. These structures are sites of galaxy formation, with smaller halos host-

ing individual galaxies and larger halos hosting galaxy clusters. The interplay between the formation and growth of dark matter halos and their resident galaxies is the domain of the "galaxy-halo connection" (e.g. Kereš et al. 2005; Conroy & Wechsler 2009; Reddick et al. 2013; see Wechsler & Tinker 2018 for a recent review). This gravitational link between luminous and non-luminous matter is a key opportunity: by studying the distribution of galaxies, we can learn about the distribution dark matter. The distribution of dark matter, in turn, is tightly linked to the cosmology; this is the domain of "large-scale structure", which we will discuss in Section 1.2.

The dark matter halo model has had great success in describing observations from galaxy surveys, and yet dark matter's fundamental nature remains a mystery. A preponderance of evidence suggests that it is non-baryonic and dynamic, in the sense that it can move and cluster as opposed to being a static modification of GR. For example, gravitational lensing cluster masses (Fischer & Tyson 1997; Bartelmann & Schneider 2001 for a review) and galaxy rotation curves (e.g. Rubin et al. 1980; Brand & Blitz 1993) require massive reservoirs of non-luminous matter; phenomena such as the CMB acoustic peaks (Smoot et al. 1992; Spergel et al. 2003; Planck Collaboration et al. 2014) and, famously, the Bullet Cluster (Markevitch et al. 2004) require that this matter be non-baryonic. Common classes of theoretical dark matter models include WIMPs (weakly interacting massive particles, e.g. Jungman et al. 1996), axions (e.g. Preskill et al. 1983), and primordial black holes (e.g. Hut & Rees 1992; Carr et al. 2016), yet detection of any of these classes remains elusive (and not for lack of trying). Observational tests for deviations of dark matter's behavior from that of a cold, collisionless species will continue to provide key constraints that theoretical dark matter models must satisfy.

Despite its success, ΛCDM is still a relatively young theory: it was only placed on

solid observational ground 20 years ago with the supernovae observations of Riess et al. and Perlmutter et al. in 1998 and 1999. The ensuing generation of ensuing CMB, BAO, weak lensing, and supernova observations (the "Stage II" dark energy experiments, in the taxonomy of the U.S. Department of Energy's Dark Energy Task Force, Albrecht et al. 2006) confirmed this paradigm. The "Stage III" experiments that followed pushed ΛCDM to its limits, most notably with the 4.4σ tension between the value of the Hubble constant H_0 inferred from type Ia supernovae in the local universe and that inferred from the CMB with *Planck* (Riess et al. 2019). Similarly, weak lensing experiments consistently find a lower amplitude of density fluctuations than CMB experiments, although the significance and origin of this effect is still debated (Leauthaud et al. 2017; Abbott et al. 2018a). We thus live in an exciting time for cosmology: ambitious "Stage IV" dark energy programs are being planned and constructed (DESI, DESI Collaboration et al. 2016; *WFIRST*, Spergel et al. 2015; *Euclid*, Laureijs et al. 2011; CMB-S4, Abazajian et al. 2016, among others) and with this deluge of high-fidelity data must come high-fidelity theoretical models. Generating such models for large-scale structure is the focus of this thesis.

1.2 Large-Scale Structure as a Probe of Cosmology

1.2.1 Overview

From the earliest days of extra-galactic studies, it has been known that galaxies are not distributed randomly on the sky (c.f. Shapley's 1933 studies with the Harvard College Observatory photographic plates; see Abell 1965; Bahcall 1977; Oort 1983; Geller & Huchra 1983 for more examples of early work). The defining feature is the clumpiness of the

distribution: many galaxies live in clusters and groups thought to form at the intersection of filaments (themselves the intersections of walls or "pancakes", Zel'dovich 1970). In the ΛCDM paradigm, this clustering arises from gravitational instability of tiny Gaussian density fluctuations imprinted in the matter distribution at the epoch of recombination, $z \approx 1100$, or about 380,000 years after the Big Bang (Planck Collaboration et al. 2018). These density fluctuations grow from an initial fractional amplitude of about 10^{-5} into the rich "cosmic web" of structures observed today—the so-called "large-scale structure" (LSS).

1.2.2 Growth of Structure and Redshift-Space Distortions

The growth of large-scale structure is a sensitive probe of the underlying cosmology. In General Relativity, the scale factor a of the Friedmann-Lemaître-Robertson-Walker metric describing the background expansion is linked to the amplitude of linear growth $D(a)$ of cold dark matter via the growth equation

$$\frac{d^2D}{dt^2} + 2H\frac{dD}{dt} = \frac{3\Omega_m H^2}{2} D. \tag{1.1}$$

The cosmology enters implicitly through the Hubble factor $H(a) = \dot{a}/a$ and explicitly through the matter density parameter Ω_m. For $\Omega_m = 1$, this has the familiar solution $D \propto a$, but in general this differential equation does not admit an analytic solution and its time evolution will have characteristic scales imprinted by the cosmology. Thus, measuring D as a function of redshift constrains cosmology.

Of course, one does not measure growth factors; in a traditional galaxy survey, one measures galaxy angular position and redshift. To infer the growth of structure from the

3D galaxy density field, one must thus infer a distance from the redshift[1]. The redshift arises from a number of sources but primarily (i) the cosmological expansion and (ii) the peculiar velocity of a galaxy relative to the Hubble flow. The latter is typically sourced by random motions of a galaxy within a halo ("Finger of God"; Jackson 1972) or coherent motions of matter draining out of voids and into over-dense regions ("Kaiser effect"; Kaiser 1987); these are collectively called "redshift-space distortions" (RSD).

While RSD is a contaminant in reconstructing the real-space density field, it also contains cosmological information in its own right. This is unsurprising: if different cosmologies predict different growth amplitudes, then so too must differ the velocities that delivered the matter there. In particular, Kaiser RSD imprints an anisotropy in the clustering of galaxies relative to the observer's line of sight that probes the parameter combination $f(z)\sigma_8(z)$, where $f(z) = d\ln D/d\ln a$ is the logarithmic growth rate and σ_8 sets the amplitude of linear power spectrum (Hamilton 1998; Percival & White 2009). RSD is particularly interesting because it directly links an observable (redshift-space clustering) with the background cosmology.

In practice, RSD is typically limited in its usefulness by systematic uncertainties in small-scale clustering. Many analyses are thus restricted to quasi-linear regimes; for example, the SDSS-III BOSS RSD analysis (Chuang et al. 2016) only considers scales larger than $40h^{-1}$ Mpc. Better modeling in the non-linear regime will be needed to take full advantage of the sub-percent constraints on $f(z)\sigma_8(z)$ that Stage-IV dark energy experiments will provide. In terms of dark energy science, RSD is expected to improve the

[1]Projected and angular correlation functions/power spectra are one way to avoid this requirement but are lossy compared to 3D measurements.

errors in the DETF FoM[2] by 10–15%(Weinberg et al. 2013), if theoretical modeling can be improved to sub-percent accuracy in the mildly non-linear regime. Precision numerical simulations will be an important part of this modeling which is one of the motivations for the work in this thesis.

RSD has another important use: constraining "modified gravity" theories. These theories appeal to a modification of GR to explain the apparent cosmic acceleration rather than an extra source of energy density with a negative pressure; thus, the growth of cosmic structure generically couples differently to the background expansion than it would in GR. RSD is therefore expected to yield a different signature than that of ΛCDM for the same expansion history (e.g. Wang 2008). This will be a key probe by which GR is tested in the dark energy Stage-IV era (Weinberg et al. 2013).

1.2.3 Weak Lensing

Just as galaxy redshifts are distorted by the motion of large-scale structure, so also are galaxy angular positions deflected by the mass. Matter along the line of sight to a galaxy gravitationally lenses the emitted light such that the observed position and shape is distorted. As with RSD, this may be viewed as a contaminant or an additional source of information: the correlation of coherent galaxy shears over large scales "weighs" the lensing structure along the line of sight (Gunn 1967; Kilbinger 2015 for a recent review). The signal is quite weak and requires an exquisite understanding of instrumental effects and intrinsic alignments (e.g Mandelbaum et al. 2005; Troxel & Ishak 2015) but is a power-

[2]Dark Energy Task Force Figure of Merit: the inverse of the area encompassed by the error ellipse in the w_0–w_a plane.

ful probe of cosmology; most recently, the Dark Energy Survey (a photometric imaging survey, Abbott et al. 2018b) used it to constrain $S_8 = \sigma_8(\Omega_m/0.3)^{0.5}$ to 3%. Furthermore, weak lensing is sensitive to the combined Newtonian gravitational potential and relativistic curvature of space; modified gravity models thus predict different behavior for weak lensing than growth-of-structure probes that respond only to the gravitational potential (Weinberg et al. 2013).

On the topic of weak lensing, it is worth mentioning its power as a tomographic probe of LSS. In tomography, one measures structural properties as a function of redshift, usually by dividing a nominal galaxy sample into multiple redshift bins. Measuring the redshift evolution of the bins probes something more akin to the actual growth factor (Eq. 1.1) than the velocity measurements of RSD or the angular diameter distance measurements of BAO. See especially Hu (1999) and Hildebrandt et al. (2017) for recent measurements; a similar method is possible with 21cm observations of neutral hydrogen at high redshift (Madau et al. 1997; Morales & Wyithe 2010 for a review).

1.2.4 Halo and Galaxy Bias

Galaxies are useful tracers of the total matter density, as on large scales baryonic matter follows the (gravitationally dominant) dark matter (e.g. Eisenstein & Hu 1998; Angulo et al. 2013 for a numerical study). However, galaxies are not *fair* tracers of the dark matter, in the sense that they preferentially inhabit dense regions and tend to avoid voids. On large scales, this can be described to an excellent approximation with linear bias: $\delta_g = b_{gm}\delta_m$, where δ_g and δ_m are the overdensity fields of the galaxies and dark matter, and b_{gm} is the (linear) galaxy-matter bias. This approximation is not empirical

but is actually an exact result of linear perturbation theory, making it a highly robust description of large-scale clustering (Desjacques et al. 2018).

As with redshift-space distortions and lensing, galaxy bias is both a complication and an opportunity. The astrophysical processes that govern galaxy formation and (small-scale) clustering are substantially more complicated than the collisionless dark-matter physics thought to govern large scales; thus, first-principles simulations of bias as a function of redshift and galaxy population are much more difficult than predictions for dark matter clustering. However, the bias boosts the signal-to-noise ratio in large-scale studies, which is very helpful for BAO analyses that need to identify a particular feature in the linear clustering regime. Indeed, only 47,000 galaxies were needed for the original 3.4σ BAO detection using massive, biased galaxies (Eisenstein et al. 2005). This signal boost can be seen clearly in the well-known expressions for the two-point correlators of biased tracers: $P_g(k) = b_{gm}^2 P_m(k)$ and $\xi_g(r) = b_{gm}^2 \xi_m(r)$ both receive an enhancement of b_{gm}^2.

Looking forward, this signal boost will be key opportunity in the campaign to detect the clustering of early galaxies in the redshift 8–10 universe. There, the extreme halo bias values of 8–30 may lend itself to clustering detections with merely hundreds of objects (Zhang et al. 2017).

The physical origin of bias may be considered through the formalism of the "peak-background split" (Sheth & Tormen 1999). Fluctuations in the matter density may be considered as the sum of a large-scale, smooth component and a small-scale, stochastic component. A halo is considered to form when the total matter density in a region exceeds a critical threshold. Halos will thus preferentially form in large patches where the smooth component has enhanced the local density "floor"; clustering is therefore enhanced in

high-density regions, but not in a directly proportional way; the critical density threshold probes the non-linear tail of the Gaussian density excursion distribution—whence bias.

While galaxy bias generally is greater than one, it is entirely possible for a given population to be less clustered than the dark matter. This would typically happen for a low-mass tracer: in dense regions, the hierarchical assembly of clusters through mergers will remove low-mass halos, carving out "no-go" regions for small halos, causing them to be less clustered than the dark matter (e.g. Mansfield & Kravtsov 2019).

1.2.5 Baryon Acoustic Oscillations

Overdensities in the primordial plasma of the pre-recombination universe launch baryonic sound waves known as the "baryon acoustic oscillations" (BAO, Eisenstein & Hu 1998). These waves, supported by photon pressure, propagate until $z_* \approx 1100$, when the pressure disappears due to the recombination of protons and electrons into neutral hydrogen. The distance these sound waves travelled imprints a characteristic scale in the matter density: the overdense regions that initially launched the waves will be surrounded by a ring of likewise overdense material at the acoustic scale. As structure formation proceeds through gravitational instability, galaxies will preferentially form at these sites of overdensity, and thus an excess of low-redshift galaxy pairs can be observed at a separation corresponding to the high-redshift acoustic scale. By using CMB observations to constrain the physical acoustic scale, the BAO can thus be used as a standard ruler to infer the angular diameter distance $D_A(z)$ to a low-redshift galaxy sample. In the redshift (line of sight) direction, BAO probes the Hubble parameter $H(z)$.

BAO is a remarkably robust technique. The physics that determine the acoustic scale

are well understood, depending only on the integral

$$r_s = \int_{z_*}^{\infty} \frac{c_s(z)}{H(z)} dz, \tag{1.2}$$

where $c_s(z)$ is the sound speed. The expansion history $H(z)$ and sound speed are readily determined by the ratios of matter, baryon, and radiation density; Planck has determined this scale to 0.2% (Planck Collaboration et al. 2018). Furthermore, the sound waves are relativistic, meaning they travel large comoving distances (~ 150 Mpc) where the physics of galaxy formation can cause no appreciable shift in the acoustic scale. Indeed, for any effect to shift (and not just broaden) the acoustic scale, pairs of galaxies separated by ~ 150 Mpc must "know" about each other. This is difficult even for non-linear gravitational evolution, since the cosmic density variation smoothed on those scales is only 1% (Weinberg et al. 2013). Perturbation theory predictions place the expected scale shift at around 0.25% at $z = 0$ (e.g. Padmanabhan & White 2009); N-body simulations find a similar effect (Seo et al. 2010).

The galaxy displacements that shift and broaden the acoustic peak are governed by the large-scale gravitational potential. If one could measure this potential, one could integrate the galaxies' trajectories backwards to restore the sharp, unbiased acoustic peak of the linear power spectrum. Indeed, with a galaxy survey one can do just that: since the galaxies give an excellent guess as to the matter distribution, the potential under which they move can be inferred. Even a one-step backwards integration using the Zel'dovich Approximation (Zel'dovich 1970) gives a large reduction on the BAO distance error bar, equivalent to increasing the survey volume by a factor of a few (Eisenstein et al. 2007; Padmanabhan et al. 2012). This is due to both the removal of any scale shift and the sharpening of the peak which enables better centroiding (Weinberg et al. 2013).

However, the late-time potential field is not the correct field to use for reconstruction; in the Lagrangian picture, one should use the initial potential. In other words, the initial density field recovered with reconstruction does not self-consistently reproduce the final, observed positions. But one does not observe a galaxy's Lagrangian displacement, only its Eulerian position. A whole class of techniques exist to deal with this complication and the effects of RSD, biased tracers, and sparse sampling in real and Fourier space, in two point and higher order statistics (e.g. Tassev & Zaldarriaga 2012; Burden et al. 2014; Schmittfull et al. 2015; Seo et al. 2016; Zhu et al. 2017; Schmittfull et al. 2017; Hada & Eisenstein 2018).

In addition to being astrophysically robust, BAO probes cosmology in a fundamentally different way than CMB, SNe Ia, RSD, or WL. BAO is based on simple physics (propagation of sound waves in a plasma, FLRW metric evolution, and gravitational instability) and provides an absolute distance scale to the high-z universe ($z \gtrsim 0.5$) not well-probed by supernovae. Having multiple avenues by which to probe cosmology (all with different systematic errors) is a key aspect of the modern dark energy campaign (Albrecht et al. 2006; Weinberg et al. 2013). In a qualitative sense, BAO is a tremendous validation for the concordance model of cosmology and the connection between the vastly different regimes of the early universe and the cosmic web, separated by billions of years, a factor of ~ 1000 in scale factor, and factors of billions in density.

1.3 N-body Simulations as a Modeling Tool for LSS

1.3.1 Overview

Large-scale structure has tremendous power for constraining cosmology; indeed, far more potential power than the CMB. This can be seen from a simple "mode counting" argument: due to its three-dimensional nature, the number of LSS modes grows as $k_{\rm max}^3$, while the number of CMB modes on the two-dimensional surface of last-scattering grows as $k_{\rm max}^2$. To the extent that modes are independent samples of the linear power spectrum, the error bar thus shrinks with $k_{\rm max}^{3/2}$ instead of with $k_{\rm max}$. The trade-off is that physics of the CMB is linear on all scales and thus its modeling can be nearly exact, while analytic modeling of LSS breaks down even on quasi-linear scales. Improving this with better standard perturbation theory (SPT) or effective field theory (EFT) is a highly active area of research, but the data far outpace the theory on this front. For example, the *Euclid* requirements for the simplest dark matter clustering statistic, the power spectrum, are 1% at $k = 10h^{-1}$ Mpc (Schneider et al. 2016a); SPT presently reaches this level to about $0.1h$ Mpc^{-1}; EFT roughly $0.5h$ Mpc^{-1} (e.g. Bernardeau et al. 2002; Foreman et al. 2016). And this is before considering the complications of biased tracers in redshift space!

Fortunately, the physics of the cold, collisionless material thought to dominate the ΛCDM matter budget is well understood, even if the emergent phenomena are not. Thus, it is possible to attempt to model the clustering with first-principles simulations. The primary technique in this vein is known as "N-body", since it traces the evolution of structure with a discrete sampling of the density field with N particles.

The task of N-body is to integrate the equations of motion of particles under mutual

self-gravity (e.g. Efstathiou et al. 1985):

$$\frac{d^2\mathbf{x}_i}{dt^2} + 2H\mathbf{v}_i = -\frac{1}{a^3}\sum_{j\neq i}^{N} Gm_j \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|^3}, \tag{1.3}$$

where $\mathbf{x}_i$ and $\mathbf{v}_i$ are the position and velocity of particle i in comoving coordinates (in which the Hubble expansion is divided out). This is a massively coupled ODE, and for simulations with billions or trillions of particles the challenge becomes clear: the right-hand side of Eq. 1.3 naively requires evaluating all $N(N-1)/2 \approx 10^{18-24}$ pairwise interactions! This is quite intractable. From a computational perspective, modern vectorized processors can execute $\mathcal{O}(10^{10})$ instructions per second, and a given computer node will have ~ 20 processors which is approximately the same as the number of floating-point operations (FLOPs) per interaction. Thus if we allow each simulation time step to take 1000 seconds, we require at least $\mathcal{O}(10^5)$ nodes—an inordinate number. Graphics processing units (GPUs) significantly reduce this hurdle, as they have approximately $100\times$ more cores than CPUs, but evaluating all pairwise interactions is still hugely inefficient (and ignores periodic boundary conditions; see below). Most pairs occur at large separations, but the exact details of distant mass distributions are relatively unimportant. Thus, tree, multipole, or mesh methods are commonly used to approximate the far-field interactions. We will discuss these in more detail below and then introduce our Abacus method in Chapter 2.

Regardless of the force evaluation method, we need an integration technique to evolve the particles forward in time. Most cosmological N-body codes use a variation of leapfrog with sub-cycling inheriting from Quinn et al. (1997). Leapfrog is ubiquitous because it is symplectic and accurate to second-order in the time step, even though it only requires one force evaluation (compare with Euler's method which also requires one force evalua-

tion but is only first-order accurate). Force evaluation is relatively expensive in N-body simulations, and most higher-order methods require storing intermediate data which is prohibitive with billions of particles. However, N-body simulations do probe a large range of densities and thus a large range of timescales, so most codes assign a time step or time-step level to individual particles and do many smaller steps for a given global step—this is the idea of sub-cycling. Effectively, the force is split into slowly varying and rapidly varying components, with a different time step assigned to each. The exact implementations vary, notably with some preserving the symmetry of pairwise interactions (and thus conserving momentum) and some not, but the potential performance gains are large.

The potential of N-body as a probe of gravitational dynamics has long been recognized (e.g. Aarseth 1963; Aarseth et al. 1979; Efstathiou et al. 1985; Hockney & Eastwood 1988; Bertschinger & Gelb 1991; Aarseth 2003). Indeed, the earliest such simulation is likely the analog experiment of Holmberg (1941) using light bulbs and photocells to measure gravitational interactions between galaxies—the $1/r^2$ gravitational force law being replaced with the $1/r^2$ luminous flux, and the galaxies being represented by 37 light bulbs each. In the 1980s, the famous simulations of the DEFW "Gang of Four" (Davis et al. 1985) provided the necessary theoretical framework in which to interpret the CfA Redshift Survey's observations of the cosmic web (Davis et al. 1982), thereby correctly inferring the cold nature of the dark matter (and earning a Gruber Prize[3] in the process).

Today's cosmological simulations are designed for rigorous validation of galaxy survey analysis and exploration of the growth of large-scale structure. This requires billions or trillions of particles; notable early simulations in this regime include Millennium (Springel

[3] `https://gruber.yale.edu/prize/2011-gruber-cosmology-prize`

et al. 2005) and Bolshoi (Klypin et al. 2011). Each of these has generated tremendous legacy value from data products such as halo catalogs and merger trees. The current and upcoming generation of simulations will feature 10^{11-13} particles (e.g. the 2 trillion particle simulation of Potter et al. 2017 completed in support of *Euclid*); the aim of this thesis is partly to make this goal computationally accessible, but more importantly to lay the theoretical and computational groundwork for exquisite understanding of N-body systematics and the way in which our simulations differ from the observed universe.

1.3.2 Initial Conditions

Initial conditions for N-body simulations can be computed from Lagrangian perturbation theory (LPT; Zel'dovich 1970). The task of LPT is to produce a set of particle displacements and velocities that self-consistently reproduce initial density power spectrum of the universe while obeying the dynamics. The density power spectrum is well-predicted by linear Boltzmann codes like CAMB (Lewis et al. 2000) or CLASS (Lesgourgues 2011). At early enough times when density fluctuations are significantly below unity, LPT is an excellent approximation. Thus, LPT can be used to "integrate" the dynamics from $z = \infty$ to $z_{\rm init}$, at which point the N-body simulation can take over. We review the basics of LPT in Appendix D.

Because the validity of LPT depends on the amplitude of density perturbations, the choice of initial redshift depends on the particle mass (with smaller mass requiring higher initial redshift for red-tilted power spectra). Typical initial redshifts are $z_{\rm init} \approx 200$ to 50, with 2nd-order LPT (2LPT) allowing a lower starting redshift than 1st-order LPT, usually called the "Zel'dovich Approximation" (ZA; Zel'dovich 1970; Crocce et al. 2006).

A lower starting redshift is also typically favorable for N-body codes that have difficulty evolving the delicate, nearly-force-free initial state of particles, as this usually requires precise cancellation of a near-field and far-field force. Furthermore, even an exact N-body solver yields incorrect growth rates compared to the Vlasov solution, so a later starting redshift is preferred to avoid build-up of this effect. This is the subject of Chapter 4.

The initial conditions are also simplified due to the assumption that dark matter is "cold"; that is, has non-relativistic velocity at the time of decoupling. Relativistic, fermionic dark matter such as neutrinos obeys a Fermi-Dirac momentum distribution, so each point in space must be sampled by many particles with different velocities. This leads to hugely increased shot noise, as the phase volume one needs to simulate is no longer a 3D sheet but instead a 6D volume. Recently, a large amount of effort has gone into suppressing this shot noise in N-body simulations with neutrinos (Emberson et al. 2017; Banerjee et al. 2018; Bird et al. 2018; Liu et al. 2018; Tram et al. 2019); fortunately (from a computational prospective), CDM is still a good match to observational data, and thus the velocity is single-valued initially at each point in space.

1.3.3 N-body Force Solvers

The biggest difference among N-body codes is in the force solvers—how they evaluate Newtonian gravity at each time step. Explicit evaluation of all pairwise forces is prohibitive, so several classes of techniques exist to reduce the computational complexity.

Particle Mesh & P^{3}M

Particle mesh (PM) methods (e.g. Hockney & Eastwood 1981; Centrella & Melott 1983; Klypin & Shandarin 1983) leverage the Fast Fourier Transform (FFT; Cooley & Tukey 1965) to solve Poisson's equation on a mesh. Particles do not interact directly with each other but instead move in the smoothed potential of all other particles. The particle mass is deposited on a lattice at each time step using an interpolation scheme such as nearest grid point (NGP) or triangle-shaped cloud (TSC), and the discrete density field is then Fourier transformed to solve for the potential. The gradient of the potential is then interpolated back to individual particle positions to give the acceleration.

This scheme is quite efficient, as the work is $\mathcal{O}(N)$ in the number of particles. Furthermore, the use of FFTs is amenable to periodic boundary conditions. However, the force resolution is quite poor near the mesh spacing, and the work scales as $\mathcal{O}(M^3 \log M)$ in the number of mesh cells per dimension. The memory requirements of a large mesh can be very restrictive, too, scaling as $\mathcal{O}(M^3)$. PM is not used today as a "full N-body" solver, although it has received some renewed interest as an approximate method comparable to 2LPT or COLA (Tassev et al. 2013; Feng et al. 2016).

To solve the issue of small-scale force resolution without a brute-force increase in the FFT mesh size, a class of hybrid "particle-particle particle-mesh" (P^{3}M) methods has evolved in which nearby particles interact via direct summation and distant particles interact via the smooth mesh potential (Eastwood & Hockney 1974; Eastwood et al. 1980; Efstathiou & Eastwood 1981; Efstathiou et al. 1985). This method can resolve close encounters while keeping the amount of $\mathcal{O}(N^2)$ work to manageable levels. The difficulty is that one cannot naively co-add the direct forces and mesh forces, since the

mesh forces include the contribution from all particles, including nearby ones. Thus, one would end up double-counting particles. Instead, the direct force law must be modified from $1/r^2$ to a "compensated" form—essentially $1/r^2 - F_{\text{mesh}}(r)$. The resulting expression is quite complex and prohibitively expensive to compute, so it is usually approximated with interpolation or spline. This is also typically done isotropically which introduces an additional error because the mesh force is anisotropic on small scales at the few to 10% level (Habib et al. 2016).

In Chapter 2, we will introduce the Abacus method which similarly splits the force into a short-range and long-range component, but without any overlap and thus no need to compute a compensated kernel.

Trees

Another method of reducing the amount of $\mathcal{O}(N^2)$ direct force work is based on the observation that most particles pairs occur at large separation in a quasi-homogeneous distribution. Thus, it would be useful to be able to represent distant particles as a monopole or low-order multipole since the details of distant mass distributions matter very little. The challenge is implementing an efficient space partitioning scheme that facilitates decision making about whether a given particle should interact with another directly or as part of a monopole.

Tree structures are one such partitioning scheme. As famously implemented by Barnes & Hut (1986), an "oct-tree" recursively partitions space into 8 equal sub-volumes until the smallest sub-volume contains only one (or a few) particle(s). Each sub-volume is a "node" in the tree; the whole volume is the root node, and nodes with no children are leaf

nodes. Every node contains the total mass and center of mass of all particles beneath it (or a multipole expansion of same). Each particle may then interact with nearby particles directly and with distant particles via their parent node. These decisions are made by "walking" the tree starting from the root node and determining the highest-level node in each branch that satisfies the opening-angle criterion $l/D < \theta$, where l is the distance to the node and D is its extent. θ thus sets the accuracy of the method, allowing for a clear accuracy-time trade-off.

Tree construction takes $\mathcal{O}(N \log N)$ time, since N particles must be inserted and each insertion requires a decision at each of $\mathcal{O}(\log N)$ levels. Similarly, force evaluation takes $\mathcal{O}(N \log N)$ time. Trees are usually reconstructed at each time step for maximum accuracy.

Modern CPU architecture does have implications for the efficiency of tree methods. The compute patterns of tree construction and walking are not naturally amenable to SIMD ("same-instruction, multiple-data") vectorization, and the data access patterns are not amenable to high cache utilization. Similar complications apply to GPU architectures; developing efficient tree algorithms is an active area of research (e.g. Teyssier 2010; Warren 2013; Potter et al. 2016).

Aside from computational complications, tree codes introduce an additional physical complication: they do not readily provide a way to sum over the infinite replicas implied by the periodic boundary conditions. This is handled naturally by Fourier methods, so most tree-based codes (e.g. Dubinski et al. 2004; Springel 2005; Teyssier 2010; Habib et al. 2013; Potter et al. 2016), use a particle mesh or other Ewald summation technique (Ewald 1921) for the large-scale force and the tree for the intermediate and small-scale force. These are so-called "TreePM" codes.

1.3.4 Beyond Cosmological N-body

Cosmological simulations are not the only application of N-body. For example, planetary dynamics (both solar system and exoplanet) relies on numerical modeling of resonant capture and migration, requiring nearly machine-precision stability over billions of orbits (Chambers & Migliorini 1997; Lissauer et al. 2011; Rein & Tamayo 2015; Rein & Spiegel 2015; Hernandez & Bertschinger 2015). Similarly, star clustering modeling requires resolution of the two- and three-body relaxation processes that exchange cluster binding energy for binary binding energy or kinetic energy in ejected stars (Perryman et al. 1998; Baumgardt & Makino 2003; D'Ercole et al. 2008). Both of these cases are quite different from cosmological N-body. The stochastic nature of the cosmological density field makes exact resonance modeling and machine-precision energy conservation less important, and the collisionless nature of dark matter means we do not need to resolve binary orbits of dark matter particles. Indeed, we do not want to resolve such orbits, as the dark matter particles should be acting as collisionless tracers of the 3D phase-sheet formed by the cold dark matter (Abel et al. 2012); this is why the exact $1/r^2$ force law is typically softened at small r. The distinction between collisional and collisionless N-body simulations is key, as we are not actually seeking to model the behavior of the system of N particles but instead appealing to some underlying, continuum Vlasov-Poisson distribution function that a putative WIMP dark matter particle would follow. The extent to which N-body is useful model of this behavior is explored in Chapter 4.

Dark-matter-only N-body simulations ignore the physics of galaxy formation which is clearly a poor approximation on small scales. However, because hydrodynamical forces are local and cannot "coordinate" on large scales, ignoring them is an acceptable approx-

imation for the largest scales of interest (e.g. BAO). Indeed, baryons and dark matter trace each other very well on large scales (Angulo et al. 2013), so the standard method for setting up N-body initial conditions is to use as the initial power spectrum the combined baryon and DM power spectrum at $z = 0$ and simulate a single, combined species rather than two separate species. The late-time clustering will thus be representative of both species, even if the early clustering is not. In this sense, "dark-matter only" simulations are not truly that but instead "matter only" or perhaps "gravity only", since the mass and linear power spectrum from two species is included in the initial particles.

Still, the matter clustering on scales as large as tens of Mpc can be affected by AGN feedback and other energetic processes (van Daalen et al. 2011; Mohammed et al. 2014; Schneider & Teyssier 2015). Unfortunately, numerical hydrodynamics remains prohibitively expensive, with the largest realizations (e.g. the $200h^{-1}$ Mpc box of Illustris TNG; Springel et al. 2018) only reaching a fraction of the multi-Gpc^3 volumes required for systematic error tests for DESI-like surveys. Thus, N-body simulations remain the primary tool for structure formation, upon which observationally- or hydro-inspired galaxy models can be painted for comparison with observations.

Perhaps the most well-known of these models is the halo occupation distribution (HOD; Berlind & Weinberg 2002; Kravtsov et al. 2004; Zheng et al. 2005, 2007; Zehavi et al. 2011), in which a number of galaxies is probabilistically assigned to each halo based on the halo mass (often distinguishing between "central" and "satellite" galaxies). This is main "galaxy painting" method we will use throughout this thesis. Other methods include abundance matching (AM) or subhalo abundance matching (SHAM), parametrized stellar mass/halo mass relation (SHMR), the conditional luminosity function (CLF), and semi-analytic modeling (SAM), all of which have been recently reviewed in Wechsler & Tinker

(2018). Aside from galaxy-painting models, other methods include direct prescriptions of the effect of hydrodynamics and feedback on the matter power spectrum (e.g. Mohammed et al. 2014; Dai et al. 2018); these could be viewed as a "pre-conditioning" step before painting galaxies.

Although N-body simulations simulate a cosmic volume, the General Relativistic effects are extremely small. Strictly speaking, Newtonian simulations incorrectly ignore the finite "speed of gravity" and the presence of the Hubble horizon $c/H(z)$, but for reasonable cosmologies, these have very little impact on observables. This is largely due to the fact that dark matter velocities remain non-relativistic (at most $\sim 0.01c$) and the density perturbations remain small ("weak field" GR). This result has been confirmed theoretically and with direct GR simulations (Chisari & Zaldarriaga 2011; Jeong et al. 2012; Green & Wald 2012; Adamek et al. 2013, 2014; Fidler et al. 2015; Adamek et al. 2016). Thus, it is sufficient to employ Newtonian gravity in a comoving frame; a highly convenient result from a computational perspective.

1.3.5 Why N-body?

It is worth considering why N-body has emerged as the primary model of non-linear matter clustering. It certainly has enough difficulties to give one pause: finite box size effects, particle discreteness effects, and computational expense, to name a few (all of which are problems even with an exact N-body solver!). One answer is that there are not many alternatives: 6D Vlasov-Poisson solvers are strictly memory-bound to small problems due to the curse of dimensionality (Yoshikawa et al. 2013; Tanaka et al. 2017);

even all 4600 nodes of the flagship Summit supercomputer[4] cannot hold a 300^6 phase-space mesh. Eulerian methods are severely limited due to their inability to track dynamics past shell crossing (when the velocity becomes multi-valued). Phase-sheet simulations are a promising recent development (Abel et al. 2012; Hahn et al. 2013; Hahn & Angulo 2016) but still have difficulties in dense regions and have not yet undergone the stringent validation for dark energy survey science that N-body has.

Another answer is that N-body does have many nice properties: it is automatically spatially adaptive, in the sense that particles end up in areas of interesting density contrasts and voids are relatively sparsely sampled[5]. It is extremely simple, due to its Newtonian nature, and thus possible to reason about (and optimize computationally). It automatically allows for multi-streaming, to the extent that particle-particle relaxation can be suppressed. It is Lagrangian, in the sense that mass is not tracked on a static mesh and is thus robust to Galilean boosts (or equivalently coherent flows of large patches of the universe). Likewise, one never has to worry about losing mass—a concern in Vlasov solvers. Particles can also be tagged with their initial Lagrangian locations, enabling direct exploration of the Lagrangian-Eulerian mapping and deterministic tracking of mass elements through time. And N-body does indeed converge to the Vlasov-Poisson solution in the limit $N \to \infty$.

Ultimately, N-body is a tool for helping us answer questions about cosmology through large-scale structure. However, it is also a window into interesting aspects of modern high-

[4] `https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/`

[5] Perhaps overly so; in response to the results of Chapter 4, one senior scientist said "It's probably the case that no void has ever been properly simulated."

performance computing, which we will explore throughout this thesis.

1.4 Thesis Outline

The remainder of this thesis is organized as follows. In Chapter 2, we introduce our N-body code Abacus, with special focus on the force solver and the code design it enables. In Chapter 3, we directly compare the performance and accuracy of Abacus with other state-of-the-art N-body codes by evolving a canonical set of initial conditions from the *Euclid* code comparison project. In Chapter 4, we use Abacus's exceptional force accuracy to explore the validity of common assumptions in generating initial conditions and use the results to generate better initial conditions. We also derive an implementation of 2LPT from direct force evaluations. In Chapter 5, we use these initial conditions to generate a suite of over 150 N-body simulations and data products spanning 40 cosmologies and two mass resolutions. We release them publicly as the "Abacus Cosmos" simulations. In Chapter 6, we develop a method of changing the background cosmology of a simulation—focusing on small changes with high fidelity—thus enabling re-blinding of a simulation's cosmology between epochs of a blind challenge. We validate the methodology on the Abacus Cosmos simulations. We conclude in Chapter 7.

We also offer five appendices: in Appendix A, we present preliminary results of scale-free simulations with Abacus as a route towards analytic control on N-body systematics in the deeply non-linear regime. In support of this, in Appendix B we derive numerically stable expressions for evaluation of σ_8 in power-law cosmologies. In Appendix C, we present a method of accelerating numerical convergence of an important class of integrals used in evaluating the Gaussian covariance of the two-point correlation function (2PCF).

In Appendix D, we present a pedagogical derivation of Lagrangian perturbation theory as it relates to initial conditions (essentially a pre-text to Chapter 4). In Appendix E, we identify a small bias that can arise in computation of the 2PCF from sets of particles.

Chapter 2

Abacus

2.1 Overview

ABACUS is a code for massive cosmological N-body simulations with high force accuracy based on an exact decomposition of the near-field and far-field force. In this method, developed in Metchnik (2009), the near-field and far-field force are analytically disjoint such that the near-field has no overlap with the far-field. In other words, the near-field force is exactly Newtonian gravity with open boundary conditions and not the "compensated" form arising from Green's function matching in P^3M methods (Section 1.3.3). The method is exceptionally accurate, with only one parameter (the order of the multipole expansion) controlling the accuracy of the entire method, and fast, with GPU acceleration of the simple $1/r^2$ near-field computation and vectorized CPU acceleration of the far-field multipole computation.

We will introduce the force solver in more detail below and discuss consequences for

data flow, code design, and hardware. The full mathematical details of the Metchnik (2009) method as implemented in the modern version of ABACUS will be presented in Pinto et al. (in prep.).

In the following, we refer to the number of cells per dimension in the ABACUS domain decomposition alternately as K or `CPD`. We usually use the former in mathematical contexts and the latter in software contexts.

2.2 Force Solver

The ABACUS periodic domain is decomposed into K^3 cubic cells. We organize particles into *cells*, and cells into planar *slabs*. Particles in cells separated by fewer than near-field radius R cells (typically 2) interact via the near-field force which we compute with direct pairwise summation. Particles in more distant cells interact via their multipoles (far-field force). This is illustrated in Figure 2.1 using $R = 1$.

The far-field force operates as a convolution over cell multipoles. Thus, it requires a global view of the box: at least one full dimension must be in memory at a time because we implement the convolution as a multiplication in Fourier space. This requires a forward FFT, cell-by-cell multiplication, then a inverse FFT. To accomplish the FFT along a given dimension, we thus need that full dimension in memory. But the near-field force doesn't share that requirement: only a window of $-R$ to $+R$ slabs must be in memory.

This leads to the idea of a *slab pipeline* (Section 2.4): we load a rolling window of slabs into memory, compute forces and update particles on the central slab, write out the trailing slab, and then load a slab at the leading edge. Only the central slab can

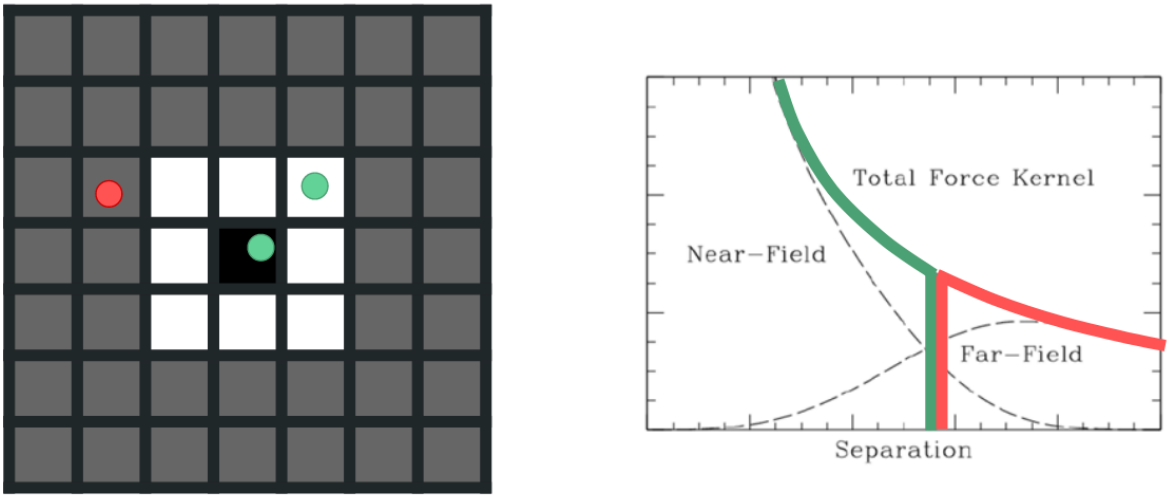

Figure 2.1: A schematic illustration of the ABACUS near-field/far-field force decomposition for near-field radius $R = 1$. Forces on particles in the black cell may come from the near field (white cells) or far field (shaded cells). Particles in the near field (green particles) interact via direct $1/r^2$ Newtonian gravity; particles in the far field (red) use a high-order multipole approximation to same. The separation is exact: there is no leakage from far to near field as is common in particle-mesh methods (right panel, dashed lines). Adapted from Metchnik (2009).

be processed at a given time because R slabs must be present on either side to compute the near force. For $R = 2$, this means 5 slabs must be in memory. In practice, we allow ABACUS to read ahead by a few slabs, so we typically have 7 slabs in memory. Typical values of K are a few hundred to a few thousand, chosen so that we have around 30 particles per cell (this is a performance tuning parameter; the optimal value will vary from system to system and the expected clustering of the problem).

The thinness of the slab pipeline is a substantial opportunity. Since not all particles have to be in memory, we don't need a large computer cluster—we can instead use a

single node and store the slabs on hard drives, reading and writing them in an ordered sweep. The raw compute power can be provided by GPUs for the near-field force and fast vectorized CPU implementations for the far-field force. Of course, one may still use multiple nodes, and here the thin slab pipeline also helps: almost all of the computation can be done node-locally, with no tightly-coupled inter-node communication or "ghost" (padding) zones. We will discuss the parallel implementation in more detail in Section 2.9.

During the slab pipeline, the multipole moments of the particle positions are computed in every cell and written to disk. After a single pipeline sweep through the volume, we thus have K slabs of the cell multipoles. Now the far-field convolution is ready to run: a "derivatives" tensor is convolved with the cell multipoles to produce a Taylor series approximation to the force in every cell. We dub these the "Taylors". The derivatives tensor is so called because it uses the derivative of the gravitational potential from the multipole moments to produce the acceleration. It encodes the mapping of multipole coefficients in one cell to Taylor series coefficients in all other cells. This tensor is fixed for a given K, R, and multipole order p and is pre-computed in a small amount of time.

The convolution is performed in Fourier space as a multiplication, so in detail we perform the 2D yz-FFT during the slab pipeline while we have the whole slab in memory (a slab spans all y & z cells for a single x). Thus, the convolution's task is to do the cross-slab x-FFT, apply the derivatives tensor in Fourier space, and do the inverse x-FFT to produce the Taylors. The inverse yz-FFT is done while applying the Taylors during the slab pipeline.

Every time step thus consists of two sub-steps: the slab pipeline and the convolution.

In the ABACUS implementation, these are separate executables called `singlestep` and `convolution`. The time step loop consists of invoking these two in a tick-tock fashion. `singlestep` dominates the runtime, with `convolution` taking about 10% of the total time.

Multipole order $p = 8$ is our usual choice that balances performance and accuracy. One way we test our accuracy is with the "Ewald test", in which we compute the forces on a random distribution of 65K particles with a brute-force Ewald summation (Ewald 1921) in quad-double precision and compare the result with Abacus's. We find that Abacus's 99% and median fractional errors are 1.6×10^{-4} and 1.2×10^{-5}, respectively. We also use a "homogeneous lattice" test, in which a uniform grid of particles is set up such that the forces should be zero everywhere. For $p = 8$, the maximum deviation is 2.6×10^{-5}, in units of the displacement that would produce that force under the Zel'dovich Approximation (Zel'dovich 1970), expressed as a fraction of the inter-particle spacing.

One reason we choose the multipole order to give such high force accuracy is that our domain decomposition is a structured mesh. When computing forces on such a repeating structure, the force error patterns are likely to not be homogeneous and random: they will vary based on position in the cell and approximately repeat in every cell. Such a spatially repeating error could readily appear clustering statistics which are some of the primary quantities we wish to measure from these simulations.

The simplicity of the near-field computation offers a substantial performance and accuracy opportunity. Due to the compact force split, if we compute the near-field force with brute force N^2 summation, then it is exact (up to machine precision) and any force inaccuracy must arise from the far-field. To increase total force accuracy, one thus only

needs to increase the far-field multipole order p. The challenge then becomes offsetting the computational load of doing so. This is where the GPU performance helps: we can decrease K to shift work from the far field into the near field, balancing the performance for the choice of p. Modern GPUs excel at the kind of work N-body requires: compute-dense kernels consisting of a few simple mathematical operations repeated many times on a small amount of data (Section 2.5). In Abacus, this performance translates directly into increased accuracy: the faster the near field becomes, the smaller the optimal K becomes, allowing us to increase p at fixed wall-clock time.

Although Abacus employs single precision (32-bit floats) by default for particle kinematic data, positions are stored as offsets relative to cell centers. This gains us an extra 9–10 bits of mantissa beyond the nominal 23 in IEEE 754. Multipole and Taylor data is stored on disk as 32-bit floats, but all internal far-field computations (Multipoles, Taylors, and FFTs) are performed in double precision to avoid potential buildup of round-off error.

The primary Abacus code paper is in preparation (Garrison et al.), of which this thesis chapter will form the core. Chapters 3, 4, & 5 contain published results from Abacus, including validation tests against analytic theory, against other codes and emulators, and in internal convergence tests.

2.3 Intuition: Abacus vs. PM/Tree

The central advance of Abacus is the exact near-field/far-field split. To understand how Abacus "carves out" the near field region from the far-field force kernel, a tempting

comparison is P^3M which also splits the force into a long-range and short-range component (Section 1.3.3). However, P^3M bins all particles onto a monolithic mesh on which the long-range force is calculated, offering neighboring particles no opportunity to "exclude" each other. ABACUS never bins particles on a mesh, instead computing multipole moments in cells. To see how this helps, though, a more instructive comparison may be a tree method.

In a tree method such as Barnes-Hut (Barnes & Hut 1986), the domain is decomposed into hierarchical cells in which multipole moments are computed. A given cell is free to interact with another cell directly, computing all pairwise interactions, or with its multipoles. If we consider direct interaction "near field" and multipole interaction "far field", then we can say that a tree method has an exact near-field/far-field split. ABACUS works the same way, with the near field given by direct interaction and the far field by multipoles, except that multipoles are computed on a Cartesian grid instead of a tree. This rigid structuring of the cells allows us to re-phrase the problem as a convolution over cells instead of many separate interactions of pairs of cells. The convolution is amenable to acceleration with Fourier methods which also offers a chance to include periodic boundary conditions for "free".

It is worth emphasizing again that the mesh size K has no bearing on the force accuracy. The opening angle to a cell in the far-field (and thus the far-field accuracy) is fixed by the near-field radius R, not K. This is quite different from PM, where a finer mass mesh means better force accuracy.

The one minor exception to this is that a higher K means a smaller near-field region and thus smaller near-field forces. In a quasi-homogeneous scenario (such as the initial time) where the near-field and far-field forces must nearly cancel each other out, this

means less "catastrophic cancelation" of two large, opposing forces at the boundary of the near field. In practice, this is a small effect.

2.4 Slab Pipeline

The particle update cycle—load, compute forces, kick, drift, store—in ABACUS is expressed as a *slab pipeline*. A slab is a plane of $1 \times K \times K$ cells (one cell thick in the x direction). The pipeline is implemented as an event loop with a set of `Dependency` objects, each with preconditions and an action. The preconditions express data dependencies between pipeline actions; a given action can only execute after its preconditions are fulfilled. For example, `Kick:2`, the velocity update for slab 2, can only execute once `NearForce:2` and `TaylorForce:2`, the near-field and far-field forces on slab 2, have completed. Dependencies may also cross slabs: `NearForce:2` must wait for `LoadSlab:0` through `LoadSlab:4`, for example (the near-field radius must be in memory).

This event-driven model makes it easy to incorporate asynchronous events, such as completion of I/O, GPU kernel execution, or MPI communication. With a properly expressed set of dependencies, the maximum amount of work possible can execute while waiting for external events. Each dependency will run exactly `CPD` times, and the pipeline finishes when the `FinishAction` has run on all slabs.

We require that dependencies execute slabs sequentially (e.g. `TaylorForce:1` must always execute before `TaylorForce:2`), even if preconditions are satisfied out-of-order (this is rare but could happen due to out-of-order I/O completion, for example). Out-of-order execution is safe but potentially wastes memory since the pipeline effectively gets

wider (more slabs in memory). For the same reason, we manually specify the first slab for each dependency rather than letting them choose to start on whichever slab first becomes available.

ABACUS has several versions of the slab pipeline: notably, on-the-fly group finding introduces a number of new dependencies. These significantly lengthen and widen the pipeline (the pipeline *length* is the number of dependencies a slab must pass through; the pipeline *width* is the number of slabs required to be in memory before the first slab can finish). Toggling between the two versions of the pipeline is as simple as stubbing the group-finding dependencies with dummy preconditions that always pass and no-op actions that do nothing. The event loop can remain exactly the same.

ABACUS also contains several simplified pipelines for various tasks, such as loading the initial conditions, multipole recovery, or standalone group finding. The initial conditions pipeline simply reads particles, computes multipoles, and generates state files. The multipole recovery is used to generate multipoles an existing state when the multipole files are missing; this would typically happen when restarting from a backup. The standalone group finding is used when one wishes to run a state or output through the "on-the-fly" group finder. Each of these is conveniently expressed with a short event loop and a small set of dependencies (many of which can be reused from the primary pipeline).

2.5 GPU Data Model

To compute the near-field force at each time step, every cell in ABACUS must interact with its 125 nearest neighbor cells (for near-field radius $R = 2$) using open-boundary-condition

Newtonian gravity. We process one slab at a time, but must have 5 in memory: the central slab is the "sink" slab that receives forces from itself and its neighbor slabs, and the neighbor slabs are "source" slabs. The amount of work is formidable for CPUs but the mathematical operations are rather simple; thus, we leverage the massive parallelism of GPUs to accelerate the computation. In this model, the efficiency of the simple N^2 approach outweighs the gains of a more complicated (e.g. tree) approach for all but the most clustered problems. The architecture of the GPU demands some care in arranging the particle data and interaction sets for efficient computation; we discuss our GPU data model here. We will use the case $R = 2$ for concreteness, so all instances of "5 cells" or "5 slabs" may be replaced by $2R+1$ cells or slabs.

Once 5 slabs are loaded into memory, we are ready to compute accelerations on the central slab. All particles in the central slab must act as sink and source particles; particles in the neighboring slabs must only act as sources. For simplicity, we do not exploit the pair-wise symmetry in the force calculation; thus, particles in the central slab appear both in the sink lists and the source lists.

We arrange the computation as "pencil-on-pencil" interactions, where every pencil is a linear block of 5 cells. Sink pencils run in the z direction, while source pencils run in the x direction (across slabs). A sink pencil centered at cell (i, j, k) will interact with the 5 source pencils centered at $(i, j+b, k)$ for $b \in [-2, 2]$. Graphically, one can think of each sink pencil being acted on by the plane of 5 source pencils that intersects its center perpendicularly. By having each sink cell appear in 5 sink pencils, centered at $(i, j, k+c)$ for $c \in [-2, 2]$, we thereby include every needed pairwise interaction exactly once. Each cell accumulates 5 partial accelerations, one for each of its parent pencils.

All of the sink pencils for a given i and range of j are indexed in a `SetInteractionCollection` (`SIC`) object along with the corresponding source pencils. Upon pencil construction, particle positions are adjusted from cell-centered to pencil-centered: this allows the GPU to be agnostic to cell divisions and seamlessly handles the periodic wrap. Sink particles are arranged into blocks with a `BlockSize` of 64 particles (or some other a multiple of the atomic CUDA warp size of 32 threads). The membership of blocks in pencils defines which sink blocks must interact with which source blocks—this is how the GPU views the pencil-on-pencil model: interactions of blocks rather than pencils.

Pencils are essentially virtual indexing structures until the particles are loaded for staging to the GPU. At that point, all pencils are explicitly constructed (and thus 5 copies of every sink particle are made). Each copy is offset to pencil coordinates as discussed above; thus, the sink and source coordinate systems differ by at most a y offset. The y offset is stored for every pencil interaction and is also passed to the GPU, where it is applied on the fly.

One CUDA kernel launch is executed for each `SIC`. Each CUDA kernel launches with `NSinkBlocks` thread blocks, each containing `BlockSize` threads. On the GPU, each thread is responsible for one sink (this is why the particle block size must be a multiple of the thread warp size). Each thread loads its sink, and then the work loop begins: each thread loads one source into shared memory and pauses at a barrier, such that all threads have access to `BlockSize` sources at a time. Then each thread loops over sources and computes the $1/r^2$ interaction, or a softened form thereof.

Why go through all this trouble to construct pencils? The simplest data model would be to compute cell-on-cell interactions, but that would be substantially less efficient. An

NVIDIA Volta GPU has 900 GB/s GPU memory bandwidth, or 75 Gsources/s. But the compute rate is 15 TFLOPS, or $\sim$ **650** Gforces/s. Thus, each source should be used at least 10 times per load to avoid being bandwidth-limited. Packing the sinks into pencils that fill thread blocks ensures 64 uses per load. Furthermore, NVLink can only transfer data from host memory to the GPU[1] at 35 GB/s, so we would like to use each source at least 250 times per transfer. With each source acting on at least 5 sink pencils (possibly more across different j), this means we only need 10 particles per cell which is achievable outside of sparse voids.

The `SetInteractionCollection` construction happens as soon as the positions and cell info for 5 slabs are loaded. No particle copies happen at this time; the `SIC` instead constructs the necessary indexing information for a later copy. The `SIC` is pushed to a work queue that is monitored by several CPU threads; when a thread is free, it pops a `SIC` from the queue and begins executing it. First, the thread constructs pencils by copying particles from the slabs to pinned memory, applying coordinate offsets on-the-fly. Then, it launches the CUDA copies, the main work kernel, and the acceleration copy-back. The thread then blocks while waiting for the results. Finally, once all the accelerations are back in host memory, the 5 partial accelerations are combined into one total acceleration; this reduction is performed directly into the acceleration slab. The result is the final near-field force for every particle that was part of the `SIC`.

We use CUDA "pinned memory" as the staging area where we construct pencils to send to the GPU and receive accelerations back from the GPU. Pinning memory locks RAM pages such that they have a guaranteed physical address (not just a virtual address).

[1] As measured by us on Summit using NVIDIA's bandwidth tool

This enables direct memory access (DMA) transfers between host RAM and GPU memory with no CPU intervention.

The copy of particles into pinned memory does apply extra memory pressure, but it ensures optimal data transfer rate to the GPU and allows the CUDA stream to proceed unblocked. The initial pinning of memory is slow, however—at least a few seconds, depending on the amount of pinned memory. This is a noticeable overhead in the Abacus model where a new process is invoked for every time step. While some work can begin without access to pinned memory, the GPU work (often the rate-limiting factor at late times) cannot. In the near-term, we plan to overlap convolution work with the GPU startup; long-term, this (and other overheads) may push us to a single-invocation model where all time steps are executed within a single process call.

We typically have three CPU threads per GPU. Each thread manages one CUDA stream and executes the pencil construction and acceleration co-addition (essentially all pinned memory work). Using three streams per GPU ensures overlap of host-device transfers and GPU compute.

This careful data packaging is all for naught if our force kernel is slow to compute. Abacus offers a number of softening options, but our preferred one is a "spline softening", in which the force law is regularized at small r but explicitly switches to $1/r^2$ at a finite radius (Section 3.4.3). From a computational perspective, the key aspects are the small number of floating-point operations (FLOPs) per interaction (about 22, plus a reciprocal square root) and the implementation of the hand-off to the $1/r^2$ form with a `min` instead of a conditional. A conditional such as an `if` statement runs the danger of triggering a conditional jump instead of a conditional move thus causing code path branching—this

is costly on the CPU and even more so on the GPU with its lock-step hardware model. Hopefully a smart compiler could infer this optimization on its own, but it is far safer to implement it explicitly, as this is the innermost loop of the entire near-field computation.

We use the NVIDIA's CUDA programming language for our GPU implementation, as it is the most mature GPU programming language and the NVIDIA GPUs have wide adoption in the supercomputing community. A port to another architecture such as Intel's Xeon Phi would be straightforward, but the hardware is still a factor of a few slower than the NVIDIA GPUs.

2.6 Top-level Interface

The top-level ABACUS interface is written in Python. The Python layer serves several purposes: it provides programmatic and script interfaces to set up a new simulation, it parses the parameter file and does preamble work, and it executes the simulation. The main time step loop is actually in the Python layer, since a new process in invoked for every time step (indeed, two new processes, since `singlestep` and `convolution` are both invoked for every time step).

This is a convenient model as it allows for flexible logic based on the state of the simulation: if no state files exist, then we know this must be an initial conditions step. If no initial conditions exist, we know we must invoke the IC generator. If no Taylors exist, then we must invoke the `convolution`. If no derivatives exist, then we must create them. If no multipoles exist, then we must invoke multipole recovery. And if none of the above are true, then we may invoke `singlestep`. These file-oriented tasks are readily

accomplished in Python and is one reason why the time step loop is not implemented in C++.

`singlestep` loads a "read state" and outputs a "write state" (a "state" is a directory containing all slab files). Upon successful completion of `singlestep`, the Python layer deletes the read state and moves the read state to the write state. In this sense, `singlestep` is idempotent (unless the user has requested that the write state overwrite the read state to save space).

The interface to run a new simulation is quite simple: typically, one creates a new parameter file called `abacus.par2` (second-level parameter file), fills it with the desired cosmology, output, softening, and other code parameters, and calls `abacus.run('abacus.par2')` in Python. The `abacus.par2` file is processed into a `abacus.par` file with various substitutions from the shell environment (and with simple math like `NP = 64**3` processed into `NP = 262144`).

Paths such as working directories and output directories are read from environment variables by default. These are set up by Abacus upon installation into the user's `.bashrc` file or a module file (on systems that support modules).

The Python layer also sets up the environment variables that the executables will see. Notably, some OpenMP settings (such as thread affinity, Section 2.8) have no runtime interfaces by which they can be controlled and must be configured with environment variables.

The executables produce a number of log files that the Python layer checks after each step. It then writes some simple information about timing and status to a global log, from which the overall simulation progress can be monitored.

Checkpoint backups are incredibly simple with ABACUS: the state directory *is* the checkpoint. Thus, making a backup is as simple as copying a directory, usually to some network file system. The Python layer handles this as well.

Different machines have different code tuning values related numbers of cores and thread bindings, so we store tuning values for our commonly-used machines in the ABACUS repository as "site files". These are loaded when parsing the `abacus.par2` file in order to insulate users from optimization minutae, but they may be overridden (or skipped entirely) as desired.

ABACUS supports several I/O modes, including sloshing, striping, and overwriting. Sloshing reads from one disk system and writes to another; striping divides even and odd state files between disk systems; overwriting writes in-place. The I/O mode may be specified separately for the state and the multipoles/Taylors. The optimal choice will depend on the performance and capacity of local disk systems; we sloshed the state and striped the multipoles/Taylors for maximum performance in the *Euclid* simulation of Chapter 3, for example.

2.7 Memory Allocation

Each "logical slab" in ABACUS has many different types of associated data, be it particle data like positions and velocities or cell data like cell offsets and sizes. Every slab in ABACUS thus has about 20 different associated "slab types", such as `PosSlab`, `VelSlab`, or `CellInfoSlab`. When requesting a slab from the ABACUS "slab buffer", one thus specifies both a slab number (an integer) and a slab type (an enum).

The ABACUS slab buffer interface is a high-level wrapper around a low-level "arena allocator". Arenas are thin wrappers around large, contiguous memory allocations. The arenas manage metadata about whether an allocation is present and how many bytes have been allocated; they also include a few "guard bytes" at either end of each slab to help detect out-of-bounds writes. The details of slab numbers and slab types are abstracted away from the arena allocator which uses a flattened "slab ID".

ABACUS puts a large amount of pressure on the memory allocator (especially with group finding, which allocates per-group temporary workspace). Memory allocation (`malloc()` and the like) is well known to be one of the slowest low-level operations on Linux systems. Indeed, we reached a point in 2018 where 15% of the total simulation time was spent just calling `free()` on arenas! The implementation in the Linux kernel is not particularly optimized for the ABACUS use-case with our mix of small and large allocations and heavy multi-threading. Furthermore, the kernel memory manager tries to ensure that memory freed in one process is available for allocation in another process; this requires remapping physical pages to new virtual address spaces. This is solving a harder problem than ABACUS needs—we can safely assume that only one (memory-hungry) process is running at a time.

Our approach to reduce kernel memory allocator pressure was twofold: reuse allocations when possible, and use a different `malloc` implementation. Implementing arena reuse was fairly straightforward within our allocator: when we discard a slab, we do not always free it but instead mark it as a "reuse slab". The next time the arena allocator receives a allocation request, it first checks if the reuse slab is present and is large enough to satisfy the allocation. To facilitate a higher hit-rate, we over-allocate arenas by a few percent. To avoid running out of memory, we only retain at most one reuse slab per slab

type (in detail, it is implemented as the $(K+1)$-th slab). In medium-sized simulations (e.g. the $N = 2048^3$, $K = 693$ *Euclid* simulation of Chapter 3), this reduces the number of fresh allocations from >10K to a few hundred.

We also replaced the built-in GNU allocator with Google's `tcmalloc`[2] ("thread-cache malloc"), a user-space allocator with high performance under multi-threaded workloads. As the name suggests, every thread keeps a cache of recently released memory, such that the allocator can often immediately satisfy small requests out of thread-local cache rather than a central store (thus no locks are required). For large requests, `tcmalloc` does use a central store, but typically does not release memory back to the kernel. Thus, all allocations after a short burn-in period can be satisfied in user-space without an expensive kernel call.

`tcmalloc` was extremely successful in handling ABACUS's memory pressure. The 15% time spent calling `free()` disappeared, but more surprisingly, it accelerated about 6 independent areas of the code that we did not even realize were affected by background memory management issues. ABACUS saw an immediate 60% performance boost as a result.

2.8 Thread Affinity and NUMA

Modern multi-socket platforms often have certain memory banks associated with certain CPU sockets. All memory remains addressable by all CPUs but at different rates. This model is known as "non-uniform memory access" or NUMA.

[2] `https://gperftools.github.io/gperftools/tcmalloc.html`

ABACUS is a NUMA-aware code: we try to ensure that CPUs, GPUs, and disks (where applicable) only access memory on their NUMA node. We accomplish this in two parts: binding threads to cores, and scheduling threads over memory consistently. The goal of the thread binding is NUMA consistency: that the socket that first touches a region of memory is the only socket that works on that region, as physical memory pages are allocated on first touch. Binding also prevents unnecessary jumping of threads among cores and flushing of caches. This generally helps ABACUS `singlestep` performance by 20%.

There are a few thread pools in ABACUS—namely, the OpenMP threads, the GPU threads, and the I/O threads. The later two are implemented with POSIX `pthreads` and are bound to cores in a user-customizable manner via the ABACUS parameter file. One typically wants to assign these threads to the CPU socket to which the corresponding hardware device (GPU or hard drive) is attached.

The OpenMP thread affinity is controlled via the OpenMP "places" mechanism (the `OMP_PLACES` and `OMP_PROC_BIND` environment variables). Each place is a set of one or more cores to which a thread may be bound; the binding parameter controls how to distribute threads to places. In ABACUS, we typically assign one place per core, so the binding is trivial.

We typically find it advantageous to give I/O threads their own cores and let each GPU spread its threads over a handful of cores. This seems to affect I/O and GPU communication at the 20% level. The OpenMP threads use the rest of the available cores. On an older six-core machine like `ted` this would not be feasible, but on our latest `hal` hardware with 28 cores, spending about 8 on GPU and I/O threads is acceptable (see

Section 2.16.4 for more on `ted` and `hal`). The OpenMP work doesn't scale perfectly with cores anyway, especially in memory-bandwidth-limited operations like the kick and drift, so losing 8 cores is a worthwhile tradeoff for more GPU performance (especially at late times). On many-core platforms like Summit (42 cores), this is an even better tradeoff.

`tcmalloc`, the memory allocator used by Abacus (Section 2.7), also has nice NUMA properties. Since every thread keeps a cache of recently released memory, a thread will likely receive memory that it recently released in response to a new allocation request. Since `tcmalloc` usually does not release memory back to the kernel, this means that the memory will still be on the same NUMA node.

2.9 Parallel Implementation

The origin of Abacus as an "out-of-core" code designed to operate on thin slices of a simulation lends itself to parallelization. Each node can be responsible for a chunk of slabs and can begin processing them immediately at the beginning of a time step without any neighbor communication. To a node, there is little difference between a slab being on disk versus being on another node.

Eventually, a node will need neighbor communication in order to process slabs near its local domain boundaries—the near-field force needs R neighbor slabs on each side, and the group finding needs about 10 h^{-1} Mpc of neighbor slabs. We implement this communication as a one-time burst transfer: once the first slab has finished on a node, all preceding slabs—positions, velocities, accelerations, cell groups, etc—are detached from the current node and transferred to the previous node. Every node thus receives slabs

from its upstream neighbor in a 1D toroidal fashion.

For concreteness, consider a node n that has slabs j through $j+N$ in memory initially: the first slab to finish will be slab k, which will be some 10 or 15 slabs higher than j due to cross-slab dependencies such as the near-force and group-finding. Slabs j through $k-1$ will then be transferred to node $n-1$, and slabs upwards of $j+N$ will be received from node $n+1$. Node n will eventually finish slabs k through $k+N$. Thus, the domain decomposition cyclicly rotates across nodes.

Received slabs are directly installed into arenas. The local dependencies are updated, too, so the slab pipeline is none the wiser about a slab having been processed on another node versus locally. Information about incoming arenas and dependencies is packaged into a *parallel manifest* which contains the types and sizes of incoming data so allocations may be performed.

The 1D parallel decomposition is ultimately not as scalable as a 2D or 3D decomposition. The implementation, however, is much simpler given ABACUS's slab-oriented nature and limits communication overheads. The main limitation of the 1D decomposition is one of memory: each node must have enough RAM to hold 10 or 20 h^{-1} Mpc worth of slabs to support group finding. Platforms like Summit with 512 GB of RAM per node are well-suited to this parallel strategy; we have tested 7000^3 simulations using 64 nodes successfully, with perfect weak scaling across a factor of 100 in problem size. The perfect weak scaling is unsurprising, given the lack of any tightly-coupled communication, but non-trivial, since the slabs themselves are getting much fatter.

We have implemented this parallel scheme using MPI. One hurdle we did not anticipate is that MPI does not support individual transfers larger than 2 GB ($2^{31}-1$ bytes,

the 32-bit signed integer maximum). In theory, one can create larger datatypes (of 1 MB in size, for example) and transfer $2^{31}-1$ of those, but we have no divisibility guarantees for many of our transfers. Thus, it is ultimately simpler to break arenas into a safe size like 1 GB. The performance overhead is minimal.

For the convolution, every node has some range of x in memory for all y and z; we need all x in memory for some y and z so we can do the x-FFT. Rather than use a canned MPI FFT routine, we opt to do a "manual" MPI transpose and a then apply a local FFT. This allow us to overlap the MPI transpose work with other parts of the code, such as GPU initialization (which is somewhat slow on Summit and must be done for every time step).

2.10 On-the-fly Group Finding

Abacus is designed for massive simulations where post-processing is expensive—one often does not one to save full particle data from more than a few epochs. Some data products, such as halo catalogs, we would prefer to have at more epochs than we have full particle outputs, especially for analyses like merger trees. Thus, on-the-fly analysis is desirable when it will not horribly slow down the simulation. With Abacus, we have a further requirement: the on-the-fly analysis must be posed in a manner that does not require all particles in memory at once. In other words, it must be implementable in the slab pipeline.

We have developed an on-the-fly friends-of-friends (FoF, Davis et al. 1985) halo finder that is fully integrated with the Abacus cell and slab structure. Aside from generating

a useful data product, we have an ulterior motive: our microstepping (adaptive time stepping) scheme uses halos to identify regions of small dynamical time that are candidates for sub-cycling. This work will be presented in Maksimova et al.; see Section 2.18 for more discussion.

Our FoF pipeline begins with independent FoF in every cell. Each cell group is bit-tagged with any cell faces of which it is within one linking length. These groups are then linked across cell boundaries within a slab, then with the slab behind them.

Cell FoF groups are formed by reordering particles in cells; the "group" is then recorded with a start index and count. "Global groups" consist of lists of cell groups. In this way, we form the FoF groups only with permutations and with no temporary copies until the final group is ready to be gathered (for convenient computation of halo properties and microstepping). Only the particle positions participate in the initial reordering in cells; velocities and PIDs are rearranged to match once we have the final permutation of particles.

We support three levels of group finding:

- Level 0 ("groups"), linking length $b = 0.2 - 0.25$: the outermost FoF groups. Used for microstepping.
- Level 1 ("halos"), $b = 0.15 - 0.2$: the traditional virialized structure. May be FoF or another algorithm such as spherical overdensity operating withing L0 groups.
- Level 2 ("cores"), $b = 0.1 - 0.15$: the innermost group. Used for tagging particles for merger trees; allows for robust reconstruction of mergers and flybys.

Not every level must be found at every time step. In microstepping simulations, only L0

must be found every time.

Finding groups in cells before proceeding to global groups accelerates the FoF considerably. The cell structure effectively prunes possible particle linkages at large distances, much as a tree structure would in a traditional FoF code. Within cells, we also apply a core-skin partitioning algorithm to accelerate group finding and use AVX to accelerate distance computations.

The group finding dependencies add significant width to the slab pipeline. Slabs cannot be released from memory until all groups containing particles in that slab are closed. Groups are closed when they cannot have any more linkages. FoF, with its extensive percolation, thus extends the physical pipeline width to 10 h^{-1} Mpc or more, independent of the slab width. It is difficult to replace FoF as the L0 group finder, however; it has strong properties of local decidability not shared by SO and most other algorithms. Local decidability is key for integration with the cell and slab structures of Abacus.

We output a number of products for L1 and L2 groups: halo properties, particle subsamples, and tagged PIDs; see Section 2.12.

Our initial group finding implementation was quite slow at early times. In the initial, near-lattice state, almost no groups are present, but all particles near faces still need to be considered as part of potential global groups that might link across a cell boundary. This caused an excessive amount of work for a regime with no groups!

We decided to augment our group finding with a local density estimate from the GPU: when computing forces, one is already computing r^2 to nearby particles. To estimate the local density, one can just increment a counter if a neighbor particle is found to be within

b^2. By setting b to the FoF linking length, we know if a particle can possibly have any neighbors. This decreased the amount of group-finding work at early times by orders of magnitude.

To carry around the density estimate, we promote triplets of near-field accelerations to quads. In detail, the density estimate is not an integer counter but an floating point accumulator of $1 - r^2/b^2$; this density estimator is smoother than the "top hat" estimator and thus gives a lower variance estimate. This is particularly useful for applications like spherical overdensity L1 group finding. On the GPU, we compute $1 - r^2/b^2$ as $b^2 - r^2$ to save the cost of a division.

2.11 In-Memory Operation (Ramdisk)

Abacus is designed to operate on problem sizes that do not fit into memory by buffering particle data on hard drives and only loading a few slabs into memory at a time. However, since 2018, GPU hardware (combined with several software engineering improvements in Abacus) has put the compute rate substantially out of balance with the disk I/O rate. Unfortunately, the I/O load is not tunable in the same way that the balance of CPU and GPU work is tunable by changing `CPD`—one can see this by recognizing that every particle must be read and written once per time step. Eventually, microstepping will substantially reduce the number of global time steps per simulation, thus easing the I/O burden, but for our flagship ambitions in the near future it became apparent that even NVMe drives would struggle to keep up with Abacus's compute rate. Thus, we developed an in-memory version of Abacus using a ramdisk.

A ramdisk is a file system that exists in RAM. It is a Linux kernel service that is accessible as a normal directory; by default, it is available under the directory `/dev/shm/` ("shm" stands for "shared memory"). By default, half of a system's RAM is available as a ramdisk. This does not limit the size of problems we can run, however, since about half of our memory allocations (e.g. accelerations and group finding data) are ephemeral anyway.

Files written to a ramdisk will stay there even after the writing process exits. They can then be read by a new process. The ramdisk files have "kernel persistence": they stay in memory as long as the Linux kernel is active.

The ramdisk model offers many advantages over a pure in-memory version of Abacus (in which the executable would be invoked once and run all time steps before exiting):

1. The `singlestep` executable can be responsible for exactly one time step, meaning global variables and objects can assume cosmology values, tuning parameters, and bookkeeping values relevant to the current step.

2. The code can remain almost identical to the non-ramdisk version, so we can continue to support normal disk and ramdisk with almost no top-level logic changes.

3. Writing/restoring a backup is as simple as copying files to/from the ramdisk directory.

4. The information flow between steps is strictly confined to information read from state files; this is a strong protection against bugs.

The shared-memory interface is POSIX-specified[3], so the ABACUS ramdisk functionality should be robust across any platform on which we might reasonably run.

In `singlestep`, ramdisk functionality is coordinated through the arena interface. Arena allocations are automatically performed in shared memory if the file path for that arena is on ramdisk (i.e. starts with `/dev/shm/`). "Reading" an arena consists of attaching a pointer using `mmap()` ("memory map"), so the "I/O" is effectively instantaneous. There are no memory copies involved.

In `convolution`, the multipoles are automatically mapped from shared memory if they exist on the ramdisk. The Taylors can be written directly into the multipoles memory, and the convolution driver renames the multipoles files to the corresponding Taylors name at the end of a successful convolution.

The ramdisk has been successful in supporting our most massive simulations (e.g. 7000^3 test runs on 64 nodes of Summit, each with 512 GB of RAM and thus 256 GB used for ramdisk), but an unexpected performance bottleneck has emerged. The `mmap()` interface has been unexpectedly slow; it is likely not designed to deal with the many massive allocations that ABACUS is performing, instead being designed for lightweight inter-process communication. Thus mapping and un-mapping memory is a noticeable overhead, of order 10% of the step time (at early, non-GPU-limited times). Preliminary investigation shows that `mmap()` is implemented in the Linux kernel with a global lock; it is possible other kernel operations we perform (such as normal memory allocations) are fighting for this lock. Possibly too the process of mapping kernel pages into user address space is causing a costly CPU context switch, or a TLB flush, or a number of other complications.

[3] `http://man7.org/linux/man-pages/man7/shm_overview.7.html`

Thus, despite the nice properties of the ramdisk model, Abacus may eventually support a full in-memory model. While the software engineering overheads of this will be larger than the ramdisk model, it will also offer the chance to eliminate other per-process overheads that Abacus incurs right now, such as pinning CUDA memory at every time step. This will also retire a risk in the parallel model which currently requires a new MPI invocation for every time step but would only require a single invocation in the full in-memory version.

2.12 Outputs

Abacus can output a number of data products. The simplest is a *time slice* output, or a snapshot, of all particles at a given redshift. The particles are written in `CPD` slabs for convenience and as a consequence of the slab pipeline (Section 2.4)—the "output" pipeline dependency is activated when a time slice is requested.

To ensure a synchronous output of positions and velocities, the drift of the step before an output is shorted such that the positions land exactly at the target redshift. The full kick during the output time step "overshoots", but the appropriate half-unkick factor is applied during the output.

Our time slice outputs are typically written in a bit-packed format called `pack14` (14 bytes per particle). The positions and velocities are stored as 12-bit offsets from cell centers, with 40 bits for particle ID. Cells usually span a few h^{-1} Mpc, so a 12 bit offset is (pessimistically) 1 h^{-1} kpc precision. Velocities are scaled to the maximum box velocity which rarely get above 6000 km/s, so 12 bit precision is 1–2 km/s. These are stored

as redshift-space displacements so that the positions and velocities use the same units. `pack14` takes less than half the space required for full 32 byte outputs but is more than sufficient for most cosmological analyses.

Abacus can also generate coarse density grids for on-the-fly power spectra. Usually, we use these to monitor simulation progress/sanity and instead use our post-processing pipeline for science power spectra as we don't want the memory cost of holding a large FFT mesh in memory during a simulation. Of course, we could disk-buffer this in a slab-oriented fashion, much like the IC code already does. We will likely explore this before executing our proposed simulation campaign on Summit (Section 2.19.2), or just repurpose the cell monopoles from the multipoles files.

Abacus can also produce particle light cones. A light cone is a simulation output in which the box is placed some distance from an imaginary $z = 0$ observer and a spherical surface sweeps inwards towards the observer at the speed of light; particles are output when their world lines intersect this surface. This produces a "synthetic observation" that takes into account the finite speed of light.

Light cones are currently implemented in Abacus with particle tags to record which particles have intersected the light cone. This prevents duplication, but creates "ragged edges" of the box during the second time a particle's world line intersects the light cone (and complicates stacking of observers to generate a longer light cone). We are exploring alternatives that do not use tags or schemes in which the tag are reset to allow for multiple world-line intersections.

Of course, Abacus generates many log files. The primary log file is a verbose record of all slab pipeline operations along with timestamps and copious debugging information.

It is mostly useful when looking for a specific pathology or when using a log parser. The timing file is the other key log file: it contains timing breakdowns of each pipeline step and sub-step along with I/O and GPU performance metrics. This quickly lets us determine if a simulation is out-of-tune. Other log files include an I/O log for each thread and convolution logs.

On-the-fly group finding generates several outputs. First are the halos: binary records of about a dozen halo properties (mass, velocity dispersion, and so on). Next are the halo particle subsamples: a 10% (user-configurable) sample of the particles in groups, ordered such that group membership can be reconstructed with indexing information from the halo records. The subsamples are selected based on particle ID and are thus consistent across time slice. These are useful for constructing crude merger trees and as sites of satellite galaxies in the HOD framework. The particle IDs are output separately from the particle subsample positions and velocities, but in the same order. HOD users will likely skip downloading this data product, while this may be the only product merger tree users need. Finally, "tagged" and "taggable" particle are output. In our hierarchical FoF scheme, particles are tagged if they are part of the innermost level—a halo core. This allows robust tracking of halos during flybys and mergers.

2.13 Analysis Tools

Abacus has a robust suite of analysis tools for post-processing of massive simulations. Compared with existing public tools, the primary consideration of our analysis chain is that the particle data may not fit in memory but that we have a guaranteed spatial segmentation of our outputs (slabs). The tools are designed to operate on one node

(compared to the MPI design of a package like NBODYKIT; Hand et al. 2018), given the provenance of ABACUS as a code designed for massive simulations on one node. We will briefly discuss our power spectrum code, our correlation function code, and a few other important utilities.

Our power spectrum code consists of three modules: a 3D particle binning module, a Fourier transform module, and a 1D histogramming module for gridded data. The 3D binning module is based on an parallelized implementation of triangle-shaped cloud (TSC) interpolation using Numba[4], an LLVM-based just-in-time compiler for Python. Numba compiles pure Python into machine code and thereby accelerates C-like loops (a classic anti-pattern in Python). This is particularly useful for cases where Numpy's array-broadcasting model of vectorization fails (usually due to memory constraints or loop-carried dependencies). We have measured our Numba implementation of TSC to be at least as fast as our original C implementation. We support processing one file at a time so that the whole particle set never has to be in memory.

Our FFT module is based on pyfftw[5], a Python interface to FFTW. Our module facilitates parallel in-place FFTs with de-convolving of the TSC window function using Numba.

Our 1D histogramming module likewise uses Numba. The module's task is to produce a 1D power spectrum from the 3D FFT. It is parallel and exploits the grid structure of the data for fast histogram bin lookups.

The efficiency of this full power spectrum pipeline is convenient for post-processing

[4] `http://numba.pydata.org/`

[5] `https://pyfftw.readthedocs.io/en/latest/`

simulations, but more importantly, it was highly enabling for the optimization procedure in Chapter 6 which had to execute evaluate a power spectrum thousands of times during a 10-dimensional non-linear optimization procedure.

Our correlation function code is backed by the `Corrfunc` code (Sinha & Garrison 2017a). `Corrfunc` applies a rectangular mesh to spatially partition the domain and prune distant particle pairs; the actual distance computations are accelerated with vector instructions. For out-of-core analysis, we operate in chunks of width $r_{\max}$ (the maximum requested correlation function radius): we compute the auto-correlation on the current chunk and the cross-correlation with the next chunk, then discard the current chunk and proceed one chunk down.

We have a custom version of `Rockstar` (Behroozi et al. 2013) that supports our `pack14` file format and outputs halo catalogs with particle subsamples in HDF5 format. We also have a standalone FoF code, but it is deprecated in favor of the ABACUS on-the-fly FoF. We support running ABACUS on-the-fly FoF in post-processing mode[6] using a redacted version of the slab pipeline. Using a redacted pipeline rather than writing new hooks into the FoF code has proven to be a fortuitous design, as later improvements to the group finding using local density estimates computed by the GPU were easily incorporated by re-including the GPU dependencies from the primary pipeline.

We provide simple Python interfaces to load `Rockstar` and FoF catalogs; these are part of the Abacus Cosmos release (Garrison et al. 2018, Chapter 5 of this thesis). They optionally employ the `halotools` package (Hearin et al. 2017) which is backed by the `astropy` Table package (Astropy Collaboration et al. 2018). `astropy` Tables has proven

[6] Off-the-fly?

an excellent package for high-performance analysis tasks given its column-oriented internal layout and support for multi-dimensional columns, compared to `Numpy` structured arrays which lack the former (van der Walt et al. 2011) and `pandas` which lacks the latter (McKinney 2010).

We provide a Python interface called `ReadAbacus` for I/O on the half-dozen particle file formats in the Abacus software ecosystem. The more complicated formats are backed by C/C++ code with a `ctypes` or `cffi` interface; the simpler ones that can be parsed directly by `Numpy` are implemented in pure Python. As many of our analysis tasks are I/O limited, the `ReadAbacus` interface also provides an asynchronous interface that transparently loads the next slab in the background while the current one is being processed.

2.14 Build System

The Abacus compilation system uses an Autoconf + Make toolchain. The user runs a `configure` script which checks for the presence of necessary libraries and sets any compile-time options for the code, such as single or double precision. Running `make` will then build the code with those options.

The `configure` script is generated from an Autoconf template file (`configure.ac`). The compiler and library dependencies of Abacus are expressed here, as are user-selectable compile-time options. The `configure` script is a portable way of checking the capabilities of the C++ compiler, libraries, and hardware in a given environment; it also provides a simple way for the user to look at the available compile-time options with `./configure`

`--help`. The result of the `configure` script is a `common.mk` file which is loaded by the Makefile, and a `config.h` file which is included by the ABACUS source code. The former includes compiler flags and library paths, and the latter includes macro definitions that can be used to enable or disable code features based on library availability or user choice.

The `configure` script also outputs a summary of the options the user has selected (output has been truncated for brevity):

```
-----------------------------------------------------
Abacus has been configured with the following options:
Double precision:                no
Near field max radius:           2
Near field block size:           64
AVX-512 directs:                 no
AVX-512 multipoles:              yes
AVX FOF:                         yes
Spherical Overdensity:           no
GPU directs:                     yes
Compute FOF-scale density:       yes
MPI parallel code:               no
Near-force softening technique:  single_spline
Maximum GPUs:                    2
Cache line size:                 64
CXX:                             icc
MKL FFT:                         yes
```

This Autoconf-based approach to the build system was inspired by ATHENA (Stone et al. 2008).

If a non-essential library is missing, the configure process will still continue but the absence is noted in the `config.h` file so that the code can compile appropriately. If CUDA is missing, for example, the user will be warned but the compilation will still proceed with only CPU directs.

Each submodule of ABACUS has its own `Makefile`; notably, `singlestep`, `convolution`, and the GPU module. Each of these loads `common.mk` from the ABACUS root directory (produced from `common.mk.in` by `configure` with the appropriate substitutions). Our old model used separate `Makefile.in` files in each directory; this led to lots of repetitious code and inability to run maintenance commands like `make clean` if ABACUS was not already configured. The `common.mk` approach centralizes inclusion of `configure` output, although some narrow `configure` options are broadcast to all `Makefile`s as a result. This approach to `Makefile` organization was inspired by CORRFUNC (Sinha & Garrison 2017b).

This model also avoids recursive `Makefile`s, which is highly desirable because otherwise a submodule's compilation might be different depending on whether `make` was called from a parent directory (where the parent `Makefile` has the chance to edit variables before the child `Makefile` sees them).

2.15 Cluster Commissioning

Abacus has been commissioned on a number of supercomputer clusters; in particular, the El Gato GPU cluster of the University of Arizona[7] which was used to run the Abacus Cosmos suite (Garrison et al. 2018, Chapter 5 of this thesis) and Summit (the number one computer on the Top500 as of November 2018[8]) at Oak Ridge National Lab[9]. Each has brought unique challenges, but overall Abacus has been very successfully ported to a wide range of architectures. Summit, in particular, is notable as a non-Intel architecture!

The port to El Gato happened in the very early days of Abacus. El Gato was one of the first clusters we commissioned on that we had not built ourselves, and thus we had to deal with lack of root access. In particular, the build system required an overhaul to support the flags and paths used by the El Gato compilers consistently across all Abacus modules. El Gato also was our first cluster with a centralized network file system and queue system; thus, we added new functionality to organize outputs, launch simulations on through a job scheduler, and re-queue interrupted simulations. For environment setup, we developed an Abacus "modulefile" that is set up upon installation in cluster environments that support the module system. These and the other tools we developed to manage suites of simulations continue to be used in our latest cluster deployment on Summit.

We commissioned Abacus on Summit in 2018. Summit is unusual as an IBM PowerPC-based system; thus all of our Intel AVX optimizations were unusable and we

[7] `elgato.arizona.edu`

[8] `https://www.top500.org/lists/2018/11/`

[9] `https://www.olcf.ornl.gov/summit/`

had to revert to plain-C code in several places. The far-field AVX code was notably slower as a result, as it involves a triple nested "diagonal" loop over multipole orders that a compiler can't unroll. Using Python meta-code to write a plain-C unrolled version was very successful, achieving nearly AVX speeds with no explicit vectorization. For even better performance we may revisit this with AltiVec instructions in the future.

The port to Summit required a rewrite of the Abacus build system; the compiler setup was simply too different from anything we had used before. Even familiar compilers like `gcc` would not accept some common arguments on Intel platforms like `-march=native`. The result was the build system described in Section 2.14; we can now detect both the compiler and architecture and select compiler flags based on the combination of the two.

Summit is extremely fast; we can sustain 50 Mp/s per node without much tuning. This speed makes it difficult for any disk system to keep up, even the node-local NVMes. As such, we developed the full ramdisk interface described in Section 2.11.

2.16 Abacus Hardware

2.16.1 Overview

Abacus was designed for massive simulations on modest hardware, accessible to a department or lab budget instead of a national supercomputer facility. As a development environment and proof of concept, we have built a number of machines in a computer lab at the Harvard-Smithsonian Center for Astrophysics—building, maintaining, and tuning these machines has been an important part of this thesis. We will discuss the hardware

choices and trade-offs that go into building a computer for ABACUS. The performance characteristics of ABACUS and the available hardware has evolved over the years, so we will also discuss how the design balance has changed.

2.16.2 Disk

For massive single-node simulations, the only "unusual" ABACUS hardware requirement is a fast array of disk. Consider the I/O demands: 32 bytes per particle—12 bytes for positions, 12 for velocities, and 8 for auxiliary/particle ID—and 356 bytes per cell—$4(p+1)^2$ bytes for multipoles (where p is the multipole order, usually 8) and 32 bytes of indexing information. For a typical value of 50 particles per cell, we thus have 2 TB of particle data and 0.4 TB of multipole data for a 4096^3 simulation. To sustain a rate of 20 million particles per second (Mp/s), the total I/O demand (read + write) is thus 1300 MB/s for the particle data.

We usually supply this with hardware RAID ("redundant array of independent disks") which distributes files over multiple disks to provide some combination of redundancy, performance, and capacity. We typically use RAID 5 which maximizes performance and capacity while still providing one disk's worth of redundancy (state redundancy is not too important, as it is straightforward to write a simulation checkpoint to another file system). A single hard drive provides about 200 MB/s under favorable conditions, so with a 10 disk RAID 5 system we could expect 1800 MB/s peak performance (one disk is lost to redundancy). In practice, we usually achieve 1400 MB/s sustained from 10 disks; at least some of the loss appears to be due to system load (that is, disappears with blocking I/O). The precise mechanism by which this operates (slower I/O threads? memory bandwidth

pressure? read/write patterns?) is unknown, but 1400 MB/s is enough to support a compute rate 20 Mp/s. We have not noticed any appreciable difference between the XFS and EXT4 file systems, even with manual RAID tunings for the latter.

Spinning hard drives read and write more slowly towards the center of their platters. Hard drives consist of several metallic disks (much like small, metal DVDs) with constant areal bit density. Thus, more bits pass under the read head per rotation on the outer edge than the inner edge. And since the drives rotate at a fixed rate (typically 7200 RPM), this translates to faster I/O on the outer portion of the platter.

This can be leveraged for better performance. Hard drives can be "partitioned" into logical segments for use by different file systems; this logical partitioning corresponds different physical regions of the hard drive. By simply creating two partitions per hard drive, one thus segments each drive into a inner, slow partition and an outer, fast partition. The fast partitions can be linked together in RAID, as can the slow partitions. This is a convenient split for ABACUS, where we have state files to which we want fast access and output files where performance is not critical. In practice, the fast partition is consistently 20% faster than the slow partition which translates directly to 20% increase performance in our large, I/O limited sims. Keeping a "clean" partition for the state files also has the benefit of minimizing file fragmentation from small files like logs.

The ABACUS slab I/O pattern of large, bulk reads and writes is quite amenable to RAID with large stripe sizes (the stripe size is the atomic unit of RAID operations). The exception is the convolution: we most hold cross-slab pencils of cells in memory in order to do the x-FFT which requires touching all files in small chunks at a time. Thus, we prefer to use SSDs (solid-state drives) which have nearly no I/O latency and are thus

better than hard drives at handling small files (or many files in small chunks). However, with enough RAM, one can load very large chunks of every multipole file into memory at once, so the cost of using an HDD instead of SSD is not so great.

We note that modern NVMe SSDs (solid-state drives) can provide well over 2 GB/s sustained from a single drive. However, they are $10\times$ more expensive per GB than HDDs and are only rated for about 1 PB of write—easily achieved in a single large simulation! The largest drives are still only around 1 TB so scaling beyond 4096^3 in a single node is also not easy. We have used NVMes successfully in more modest, 2048^3 simulations on `hal`—those results are presented in Chapter 3.

In almost all cases, we use direct I/O to bypass operating system caching of files. Our slab-oriented I/O means that we are performing exactly one read and write of each file per time step, meaning we would not benefit from caching. One exception where we do not use direct I/O is when performing I/O on network file systems. These often do not respond well to direct I/O when they support it at all.

2.16.3 GPUs: Tesla vs GeForce

In all Abacus machines that we have built ourselves (see below), we have used consumer-level NVIDIA GeForce cards instead of the HPC-marketed NVIDIA Tesla cards. The primary difference is that the double-precision performance is crippled on the GeForce cards, but our use of cell-centered particle coordinates ensures that our precision is much better than box-centered single precision anyway (see the end of Section 2.2). The price difference is considerable: a Tesla P40 cost about $6K at launch, while the GeForce 1080 Ti—the equivalent consumer card—cost only $700. Our performance when porting to

Tesla cards on clusters has not been appreciably different.

2.16.4 The Abacus Development Computer Cluster

Production Machines

The first generation of ABACUS machines (`ted` and `charlie`, built before the start of this thesis) relied heavily on commodity hardware designed for the computer enthusiast community (gamers). `ted` and `charlie` each contained a single six-core Intel i7 CPU that was overclocked to about 3.6 GHz, 64 GB of RAM, sixteen 1 TB hard drives in RAID 5, and dual NVIDIA GeForce GTX 690 graphics cards. The 690s were outliers of their generation, as they each contained two Kepler GPUs for a peak theoretical compute rate of over 5 TFLOPs. This would not be surpassed in a single card for four more generations with the advent of the 1080 (although cards did become cheaper and less power-hungry in the intervening years). `ted` and `charlie` proved that, in 2012, ABACUS could run large, 4096^3 particle simulations at rate of 8 million particles per second on a single machine costing less than $10K. `ted` and `charlie` are still useful production machines today.

The next few generations of computer hardware brought incremental improvements. We attempted to build another machine shortly after `ted` and `charlie`, but the disk system was very unstable, possibly due to hardware RAID or backplane issues. We eventually re-purposed the GPUs (TITAN Xs) for `charlie`.

In 2016, we built our first Intel Xeon-based machine, called `franklin`. In contrast to the six 3.6 GHz cores of `ted` and `charlie`, `franklin` had dual 12-core processors clocked at 2.2 GHz. To see a speedup, we thus had to improve the parallelism of many areas of

the code; some of our efforts are documented in Section 2.17. `franklin` had more (and faster) disk than our previous builds which made it suitable for even larger simulations. The large amount of RAM (256 GB) has proven useful for a few analysis tasks, such as power spectrum computation, where it is convenient to hold large arrays in memory.

The rise of many-core CPUs as a route to increased performance instead of increasing clock rates is unsurprising in some ways. The amount of power dissipated by a CPU scales as frequency cubed, while adding more cores is a linear cost[10]. Of course, to see a speedup, applications must be capable of utilizing many cores efficiently. Abacus, somewhat by luck and somewhat by design, is such a code. The computation of multipoles in cells is embarrassingly parallel and compute-dense, and the memory access patterns are predictable and highly local.

In 2017, we built `hal`, our latest machine. `hal` was different than previous generations in a few important ways: it was our first machine with substantially faster GPUs than `ted` and `charlie`; it was our first AVX-512 machine; and it was our first build with NVMe drives. `hal`'s NVIDIA 1080 Ti GPUs were nearly twice as fast as the previous generation, and the Abacus GPU rate scaled almost perfectly in line with expectations. This was a major validation of the scalability of our GPU data model (Section 2.5).

The AVX-512 processors were an opportunity to revisit legacy AVX assembly code in the far-field multipoles and Taylors. AVX-512 allows for processing of 16 floats at a time, instead of the 8 floats of AVX. Thus, the promise is a two-fold speedup, although this is rarely realized in practice[11]. This is partly because processors down-clock while executing

[10] `https://software.intel.com/en-us/blogs/2014/02/19/why-has-cpu-frequency-ceased-to-grow`

[11] See, for example, `Corrfunc` (Sinha & Garrison 2017b), where a 1.6× speedup was considered very

the power-hungry AVX-512 instructions, but instruction latencies can be different as well. We only saw a 30% speedup from rewriting the assembly code with AVX-512 intrinsics, but the new Python meta-code used to generate the AVX-512 C++ code for different multipole orders turned out to be reusable for unrolled pure-C multipoles when we ported to the Summit platform that supported neither AVX nor AVX-512 (Section 2.15).

The two Samsung 960 and two 970 Pro 1 TB NVMe drives of `hal` were a major success. We initially had several worries: that they would wear out (they are only rated for about 1 PB of write); that they would throttle under sustained I/O; and that the interior temperature of `hal` would cause them to burn out more quickly (the M.2 slots were right under the GPUs on the motherboard). While we have some evidence for thermal throttling of the 960s, the 970s were extremely stable and provided 2.8 GB/s sustained read and 2.5 GB/s sustained write in the *Euclid* simulation of Chapter 3—by far the fastest disk system we have ever used (RAID or single drive). The drives are still stable and in use today, even after 2 PB written.

Our initial impetus to supplement the 960 NVMes with the 970s was a mysterious performance decline in the write speed of the 960s: over the course of a week, their performance declined from 2 GB/s to 350 MB/s, nearly HDD speeds. This crippled the convolution rate in particular. We initially attributed it to throttling or hardware failure, until we realized that the drives needed to have the TRIM command issued to recover free blocks from deleted files. This immediately restored the full drive speed; we now have scheduled TRIM enabled.

---

successful.

2.17 Notable Simulations

In addition to the *Euclid* code comparison and Abacus Cosmos simulations presented in Chapters 3 & 5, we have completed a few other notable simulations. These have served as proofs-of-concept and helped us explore specific scientific goals.

In Zhang et al. (2017), we presented our most massive single-node simulation to date[12]: 5120^3 particles (130 billion) in a $250h^{-1}$ Mpc box, for a particle mass of $1 \times 10^7 h^{-1}\,\mathrm{M}_\odot$. The simulation was designed to explore the detectability of the clustering of the first galaxies with a JWST 13 arcmin deep-field survey. The simulation was evolved from $z = 200$ to 8 in eight weeks on the `franklin` hardware. In our analysis, we found that the extreme bias factors (5–30) of massive halos at this epoch lend themselves to detection of clustering with only 500–1000 objects, assuming that the detected galaxies occupy the most massive halos.

This was our first simulation at this scale and exposed many bugs and inefficiencies. In the far-field, this was the first time where $\texttt{CPD}^3$ exceeded 2 billion—the maximum value representable by a 32-bit signed integer. The legacy far-field code (Multipoles, Taylors, and Convolution) had to be carefully updated and re-validated to be 64-bit safe. We also updated the far-field code to store Multipoles and Taylors in 32-bit floats on disk instead of 64-bits, halving the I/O load. This had no measurable impact on the accuracy of the forces but did expose an implicit $\texttt{CPD}^3$ scaling in the amplitude of the Taylors which caused the value to exceed 10^{38}—the maximum representable 32-bit float.

Our initial rate was around 8 million particle updates per second (Mp/s), driven

[12] Aside from a few test steps of 7000^3 on Summit

entirely by CPU work—a combination of Multipoles, Taylors, CPU pencil construction, and the Kick. This was our first major simulation on `franklin` and thus our first with more than 20 cores which exposed the limits of some of our parallelization schemes. The improvements implemented for this simulation include:

1. Multiple I/O threads (one per disk system)
2. Parallelization of the GPU pencil construction indexing
3. A queue system for GPU work units
4. Thread affinity functionality for better NUMA awareness
5. New multi-threaded partition and merge-sort implementations for the insert list (by exploiting our knowledge of the sort key distribution, we can outperform the Intel TBB parallel merge sort)
6. Multi-threaded insert list insertion based on "gap tracking"
7. Multi-threaded construction of merge slabs
8. Multi-threaded I/O in the convolution driver
9. A parallel wrapper to the N-body Shop's friends-of-friends halo finding code[13] with a 2D domain decomposition, suitable for out-of-core analysis on massive sims.
10. Commissioning with the Intel compiler suite (in particular using the Intel MKL FFT; about 20% faster than FFTW)

[13]https://faculty.washington.edu/trq/hpcc/tools/fof.html

11. "Spin detection": tracking of which dependencies are blocking the pipeline from proceeding

12. New GPU performance reporting mechanism

13. New pipeline step to immediately transpose particle positions upon load (matches the GPU data model)

These improvements brought the simulation speed from 8 to 20 Mp/s, just brushing the disk speed.

Our next large simulation on `franklin` followed shortly thereafter: a more traditional, BAO-oriented simulation dubbed "FigBox" of 4096^3 particles in a $3.2h^{-1}$ Gpc box, for a particle mass of $4 \times 10^{10} h^{-1}$ $M_\odot$. After the JWST simulation, it was apparent that the GPU pencil construction model was a weakness, as it involved two copies: packing the particles from slabs into pencils, and from pencils into pinned memory for staging to the GPU. Large copies are expensive operations, particularly on Intel platforms where the bandwidth to main memory is only about 10 GB/s per core (under ideal conditions). Thus, we introduced the "deferred copy" GPU pencil model described in Section 2.5. The salient part is that a `PencilPlan` is constructed for each source and sink pencil that contains indexing information but waits to copy any particles until the GPU work unit comes up for execution. At that point, the particles are packed directly from the slabs into the pinned memory.

This model was very successful, with the overall CPU work running about 30% faster. Unfortunately, the disks could only supply 22 Mp/s, so the wall-clock time to completion was still about 8 weeks.

FigBox has been an important testing ground for on-the-fly group finding. With its large volume, it finds rare peaks and filaments in the cosmic density field that might be missed in a smaller box, and thus helps us understand the percolation properties of various algorithms. This is particularly important for on-the-fly group finding, where the largest filament sets the number of slabs we must hold in memory, and thus the requisite amount of RAM per node. A FoF linking length of 0.2, for example, finds a $20h^{-1}$ Mpc group that that is actually a string of 9 or 10 visually obvious halos embedded in a filament.

The improvements implemented for these simulations were important preparation for the *Euclid* simulation to be presented next in Chapter 3.

2.18 Current Limitations

The current version of ABACUS computes the near field with brute-force N^2 summation. While GPUs are quite good at this sort of compute-dense operation, even they can be overwhelmed by the "Coma Clusters" of a simulation (halos with millions of particles). Since the near field is compact, any open-boundary-condition Newtonian gravity solver may be swapped in, however. We currently have a prototype tree implementation for accelerating the force computation in dense clusters. The trees are constructed and walked on the CPU and the interaction sets are executed on the GPU. Our choice of when to use a tree (and what leaf opening criteria to employ) is determined by the usual efficiency-accuracy trade-off; this work will be presented by Maksimova et al.

All particles in ABACUS currently share a global time step. The time step changes throughout the course of the simulation; we set it based on the smallest dynamical time

anywhere in the box (usually at the center of a dense cluster), so the method is very accurate (see Section 3.4.2). However, we waste a large amount of effort integrating trajectories of low-acceleration particles (e.g. in voids) with ~ 1000 steps when they could be accurately represented with ~ 150. To address this, we have developed a prototype of a leapfrog multi-stepping scheme (in the spirit of Quinn et al. 1997) in which all particles are kicked with a large global time step and then individual halos are sub-cycled with a smaller time step. The scheme preserves the pair-wise of particle interactions and is thus momentum-conserving. This work will also be presented in Maksimova et al.

Microstepping does require identification of halos on-the-fly to identify accurately regions of small dynamical time (Section 2.10); this is time-consuming compared to the nominal force computation, even with our slab-oriented optimizations. The global-step savings of microstepping will likely make this a worthwhile tradeoff, but we are still investigating group finding optimizations or faster algorithms that can still be implemented in a slab pipeline.

As discussed in Section 2.11, the overheads of invoking two processes for every time step is starting to become noticeable on fast platforms like Summit and `hal`. Ramdisk memory is slow to map (although still far faster than disk I/O), and pinning GPU memory takes at least a few seconds. Eventually, we may thus consider a single-process model for Abacus.

2.19 Looking Forward

2.19.1 Public Release

We plan on a public release of Abacus in 2019. The major outstanding task before this can happen is documentation. Most of Abacus is documented, especially the top-level interfaces, but the documentation is spread out among several locations (source code comments, standalone notes describing interfaces, publications, and a PDF user guide) and written in non-standard formats across at least two programming languages (Python and C++). We will need to collect this documentation into a single user-friendly interface, likely using a Doxygen + Breathe + Sphinx + ReadTheDocs pipeline. We are currently testing a prototype of this procedure. A fair amount of new documentation will need to be written too, particularly detailing the installation process and examples of end-to-end analysis for new users.

We will also have to consider how to distribute the code. Given its Python top-level interface, it is tempting to host the project on PyPI or Anaconda (in addition to GitHub). However, the current build system requires the user to recompile the code in order to change certain physics options (such as softening and local density computation). This is orthogonal to the Python package model. We will have to decide whether enough of these options can be efficiently implemented as runtime settings that it makes sense to distribute a "compile-once" Python package, or whether users will be required to build from source (or even a hybrid model where the Python layer triggers a recompilation based on user settings).

2.19.2 Simulation Campaigns

We have submitted a proposal for a major ABACUS simulation campaign in support of DESI to the U.S. Department of Energy's "ASCR Leadership Computing Challenge". We have proposed an ambitious set of 97 simulations in $2h^{-1}$ Gpc boxes of 7000^3 particles each, for a particle mass of $2 \times 10^9 h^{-1}$ $\mathrm{M}_\odot$, to be run on the DOE's Summit computer with parallel ABACUS (see Section 2.15). We have requested 300K node-hours to complete this set of 33 trillion particles, with a total of 1.6 PB of data products.

The proposed simulations span a range of cosmologies: 25 boxes in one cosmology and 6 in five others, with single-box excursions to 34 additional cosmologies for interpolation/emulation and 8 additional for warping using the technique of Chapter 6.

If awarded, these simulations will form a core sample of catalogs by which DESI analysis will be validated. ABACUS's simultaneous computational economy and unprecedented accuracy will ensure that DESI analysis can be declared to be as robust as the methodology, not merely as robust as the simulations.

We are well-situated to execute this simulation campaign, having already executed a similar campaign (at much smaller scale) with the Abacus Cosmos simulations. The pipelines developed there for configuring, launching, and monitoring simulations in a cluster environment will immediately reusable here, even if our data management pipeline has to be scaled up. Perhaps most importantly, we have learned many lessons about the ways running a suite of simulations can go wrong. In one memorable incident, I accidentally launched a set of simulations that I thought had different initial condition seeds but instead had identical seeds, such that we ended up simulating the same box many times over! This was not a total loss, as it provided the chance to check the end-to-

end reproducibility of Abacus (which was quite good; the small-scale cross correlation was perfect within 10^{-6} at the softening scale).

Chapter 3

A High-Fidelity Realization of the Euclid Code Comparison N-body Simulation with Abacus

This thesis chapter was originally published as

Garrison, Lehman H., Daniel J. Eisenstein, Philip A. Pinto *2019, MNRAS, 485, 3370*

Abstract

We present a high-fidelity realization of the cosmological N-body simulation from the Schneider et al. (2016b) code comparison project. The simulation was performed with our Abacus N-body code, which offers high force accuracy, high performance, and minimal particle integration errors. The simulation consists of 2048^3 particles in a $500\,h^{-1}\mathrm{Mpc}$ box,

for a particle mass of $1.2 \times 10^9\ h^{-1}\mathrm{M}_\odot$ with $10\ h^{-1}\mathrm{kpc}$ spline softening. ABACUS executed 1052 global time steps to $z = 0$ in 107 hours on one dual-Xeon, dual-GPU node, for a mean rate of 23 million particles per second per step. We find ABACUS is in good agreement with RAMSES and PKDGRAV3 and less so with GADGET3. We validate our choice of time step by halving the step size and find sub-percent differences in the power spectrum and 2PCF at nearly all measured scales, with $< 0.3\%$ errors at $k < 10\ \mathrm{Mpc}^{-1}h$. On large scales, ABACUS reproduces linear theory better than 0.01%. Simulation snapshots are available at `http://nbody.rc.fas.harvard.edu/public/S2016`.

3.1 Introduction

Cosmological N-body simulations are the primary tool for forward modeling the theory of large-scale structure to observable quantities like the spatial distribution of galaxies. As observations improve, the comparison of the forward model with observations becomes increasingly sensitive to systematic errors in the N-body simulations. Some systematics can be checked analytically, such as the recovery of linear theory on large scales, but most rely on "convergence testing", in which a parameter of the simulation (such as the time step) is moved towards the continuum value until the answer stops changing (to some tolerance). Such tests can be prohibitively expensive (see DeRose et al. 2018 for a recent exhaustive effort) and are not guaranteed to converge to the physical answer.

A common additional check is to compare the "converged" results from multiple, independent codes. While not a guarantee of physical accuracy, agreement indicates control over systematics related to the numerics, to the extent that different codes use different numerical techniques. This is the approach of code comparison projects like

Heitmann et al. (2008) and Schneider et al. (2016b, hereafter S2016). The latter presents the code comparison project from the Euclid Cosmological Simulations Working Group, which compared the matter power spectrum from the PKDGRAV3 (Potter et al. 2017), RAMSES (Teyssier 2001), and GADGET3 (Springel 2005) codes.

A third path to assessing code accuracy in the non-linear regime is through scale-free simulations. In these tests, a power-law power spectrum is used in an expanding $\Omega_M = 1$ background, such that the clustering on small scales should be a rescaling of the clustering on large scales at a later time. Any deviation from this self-similarity must be due to finite box size, finite particle mass, or inaccurate numerics. The breakdown of this self-similarity can be used to identify halo mass resolution limits and other complex non-linear systematics; this will be our approach in an upcoming paper (Joyce et al., in prep.).

In this work, we contribute ABACUS's result to the S2016 code comparison project. ABACUS is a GPU-accelerated code for cosmological N-body simulations; it offers excellent force accuracy and minimal integration errors of the particle trajectory due to the small global timestep used. It also employs a compact spline softening kernel, which minimizes leakage of force softening to large scales. ABACUS's speed allows us to perform convergence tests at scale; we do not need to sacrifice volume or mass resolution to complete the tests in a reasonable amount of time with modest computational resources.

The paper is organized as follows. In Section 3.2, we discuss the ABACUS code and performance in the context of the S2016 simulation. In Section 3.3, we compare the ABACUS matter-field clustering results with RAMSES, GADGET3, and PKDGRAV3. In Section 3.4, we show validation tests for various ABACUS code parameters. We discuss

our findings in Section 3.5.

3.2 Abacus

The details of ABACUS code are presented in Chapter 2. In this section, we will focus on the performance of ABACUS on the S2016 problem and discuss the accuracy in Section 3.4.

3.2.1 Performance: Design

We present the performance of ABACUS for the S2016 2048^3 simulation on one node. The performance and low memory requirements enabled by the exact force split mean that ABACUS does not need a computer cluster to complete large simulations in a reasonable amount of wall-clock time; indeed, ABACUS presently only supports single-node operation.

We built the node used in this work, called `hal`, specifically for ABACUS using commodity computer hardware. `hal` is a dual-socket Intel platform with two 14-core Intel Xeon Gold 6132 @ 2.60 GHz, 128 GB DDR4-2666 RAM, and two NVIDIA GeForce GTX 1080 Ti GPUs. HyperThreading is disabled and the CPU frequency scaling governor is set to `performance`. `hal` is equipped with four 1 TB Samsung NVMe SSDs (two 970 Pro, and two 960 Pro). We used the Intel compiler `icc` 17, and NVIDIA CUDA 9.2. `hal` cost about $13000 and consumes approximately 1 kW under load.

The NVMe drives store the particle and convolution data. Since we only hold 1% of slabs in memory at a time, and since the CPU and GPU compute rate is so high, the drives holding the particles must be similarly fast. For this work, we used the two

970 Pros for the particle data (440 GB), and the two 960 Pros for the convolution data (multipoles and Taylors, 102 GB total). While we could have fit the whole 2048^3 problem on a single SSD, we rely on multiple SSDs to provide the throughput to keep up with the GPU and CPU. For larger simulations, we employ RAID arrays of HDDs, which offer lower performance but a better price/GB ratio.

During `singlestep`, since we can compute the near field on GPUs and the far field on CPUs, we can overlap their computation. Typically, a few CPU cores are dedicated to GPU communication, a few are dedicated to IO, and the rest are used for far-field forces and other CPU work. Thus, we have the IO, GPU, and CPU operating in parallel. For this work, we used 6 cores to prepare GPU work, 1 core for IO, and 21 for CPU work.

We carefully control the assignment of threads to cores (both OpenMP threads and our own GPU and IO threads). This is to prevent threads from switching cores and interfering with each other and to maintain NUMA locality. In most contexts, we statically schedule the OpenMP threads over y-rows, so particles will largely stay on their own NUMA node.

For this simulation, we use $K = 693$, multipole order $p = 8$, and near-field radius $R = 2$. This yields 25.8 particles per cell. K was chosen *post hoc* to optimize the trade-off between the late-time N^2 work and the increased CPU work and IO for all time steps.

We note that the S2016 particle mass ($1.2 \times 10^9\ h^{-1}\ \mathrm{M}_\odot$) is smaller than would be optimal for the current version of Abacus on the `hal` hardware. At late times, the N^2 work from the largest halos dominates the runtime; the largest cell has over 200,000 particles by the end of the simulation. To counteract this, we were forced to choose a relatively large K, which tends to under-fill the AVX-512 vectors in most cells and increase

the FFT work. This decreases far-field performance. Furthermore, the multipole/Taylor data volume increases as K^3, thus slowing down the convolution step (which is strongly IO limited). Even a factor of 2–3 increase in particle mass would cause the GPU work to be subdominant to the CPU work, increasing the total speed of the simulation.

The small force softening also leads to larger particle accelerations in the centers of halos. Since ABACUS is a globally time-stepped code, this forces us to take one or two thousand time steps to $z = 0$, instead of one or two hundred, as is common in codes with adaptive time stepping. Future versions of ABACUS will address this with on-the-fly identification of halos and refined time stepping within those halos; we call this scheme "microstepping". This will allow us to take larger global time steps and have the benefit of increasing the compute load for every time we load the particles. This should bring the compute performance back in line with the disk performance. The former currently outstrips the latter, except at late times for the low particle masses considered here.

For truly massive simulations, we are developing a parallel version of ABACUS based on the existing slab decomposition suitable for parallelization over a few dozen nodes.

While it may seem that the `hal` hardware is specialized (e.g. the combination of fast local disk and several GPUs on a single node), we note a trend in supercomputing towards this "fat node" design. For example, the Summit[1] supercomputer (number 1 on the Top500[2]) has 1.6 TB of NVMe SSD and six NVIDIA Volta V100 GPUs per node. We expect Abacus to be well-suited to Summit and Summit-like architectures.

[1] `https://www.olcf.ornl.gov/summit/`

[2] `https://www.top500.org/list/2018/11/`

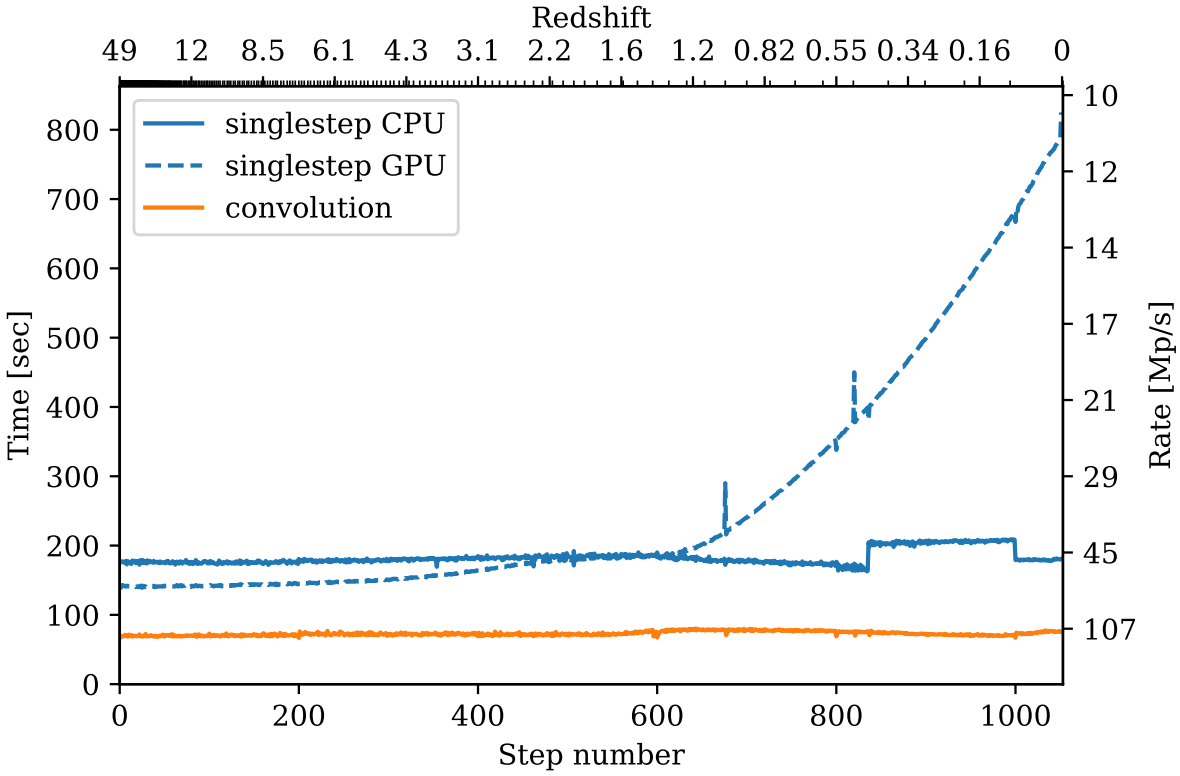

Figure 3.1: ABACUS runtime per step. The `singlestep` GPU and CPU work is overlapped, so the wall-clock time is the maximum of the two. The convolution occurs as a separate step between `singlestep` invocations. The spikes in GPU runtime are the output steps, where the CPU is too busy to prepare work for the GPU at full speed. Minor ticks on the redshift axis are steps of $\Delta z = 0.1$.

3.2.2 Performance: Results

Using the exact particles provided by the Euclid Cosmological Simulations Working Group[3] [i.e. no corrections to the initial conditions in the style of Garrison et al. (2016)], ABACUS executed 1052 time steps from $z_{\text{init}} = 49$ to $z = 0$ in 107 hours (4.5 days) for a

[3] `https://www.ics.uzh.ch/~aurel/euclid.htm`

mean rate of 23 million particles per second per step (Mp/s). The `singlestep` work took 87 hours, and the convolution 21 hours. In `singlestep`, the GPU work (near-field force) was fully masked by the CPU work until about redshift $z = 1.5$ (step 600), after which it quickly became dominant (Fig. 3.1). `singlestep` started at 46.0 Mp/s and ended at 10.8 Mp/s due to the increased GPU work. Including the convolution, ABACUS started at 33.6 Mp/s and ended at 9.9 Mp/s. See Fig. 3.2 for a visualization of the final state.

A timing breakdown of the first ABACUS time step is given in Table 3.1. This timing is representative of all time steps, except for the increased GPU work towards late times, as noted in the table.

In Fig. 3.3, we show the measured GPU wall-clock performance in terms of number of pairwise spline interactions computed per second. The performance increases as the compute density (interactions per particle) increases, peaking around step 700 ($z = 1$) or 1.2×10^4 interactions per particle. Afterwards, the performance declines, possibly due to worsening load balancing from the increasing density contrasts between cells.

We also give a rough estimate of the theoretical peak performance of our two NVIDIA 1080 Ti GPUs. We assume 10.6 TFLOPS per GPU (see above), which assumes all operations are fused multiply-add (FMA). In our spline kernel, we count 22 additions and multiplies (not all of which are FMA), a reciprocal square root (`rsqrt`), and a `min`. We count the `rsqrt` as one FLOP and ignore the `min`, even though we expect these are poor approximations. We thus derive a conservative 23 FLOP estimate, yielding a theoretical peak of 920 billion direct interactions per second (GDIPS).

We measure a peak ABACUS performance of 485 GDIPS, which is 52% of our estimated theoretical peak. We consider this excellent performance. This measurement uses

wall-clock time while at least one Abacus GPU thread is running and thus includes PCIe bus transfer overheads and load imbalancing.

A "notch" of 10% slower CPU performance is visible between steps 837 and 1000 in Fig. 3.1. After step 836, the simulation was manually paused for several minutes to run `fstrim` on the SSDs to ensure consistent write performance (we had observed catastrophic write performance decreases in the recent past that were fixed with TRIM). Upon resuming, the `singlestep` performance was slightly slower in memory-bandwidth-bound operations like Transpose Positions. At step 1000, the simulation automatically paused for several minutes to write a backup state. Upon resuming, the bandwidth issue disappeared. We do not have compelling explanation for this issue, but it also had no impact on the wall-clock time, since it was completely masked by the GPU compute time.

Table 3.1. Wall clock timing for the first ABACUS time step of the S2016 box, $z = 49$. Units of "Mp/s" mean millions of particles per second. "DP-FLOPS" means double-precision floating-point operations per second. Only rates for the dominant sub-steps are shown. Percentages are relative to their parent step. "Non-blocking" means other CPU actions can proceed while that action is running.

Action	Time [s]	%	Rate	Notes
Total	255	100 %	33.6 Mp/s	9.9 Mp/s at $z = 0$
singlestep	187	73 %	46 Mp/s	
CPU Work	187	100 %	46 Mp/s	
CUDA Initialization	7.3	3.9%		Pinning memory
Check Slab Integrity	1.6	0.9%		
Transpose Positions	3.0	1.6 %		
Prepare Near Force	7.6	4.1 %		
Taylor Force	85.2	45.5 %	100 Mp/s	
Kick	10.1	5.4 %		
Drift	7.8	4.2 %		
Multipoles	48.3	25.8 %	177 Mp/s	
Finish	14.7	7.9 %		
Waiting for GPU or IO	1.0	0.5 %		
GPU Near Force (non-blocking)	137	73 %	62 Mp/s	11 Mp/s at $z = 0$
Disk IO (all non-blocking)	.	.		
Particle Data Read	100	53 %	2.8 GB/s	285 GB read
Particle Data Write	112	60 %	2.5 GB/s	285 GB written
Taylors Read	30.6	16 %	1.7 GB/s	108 GB read

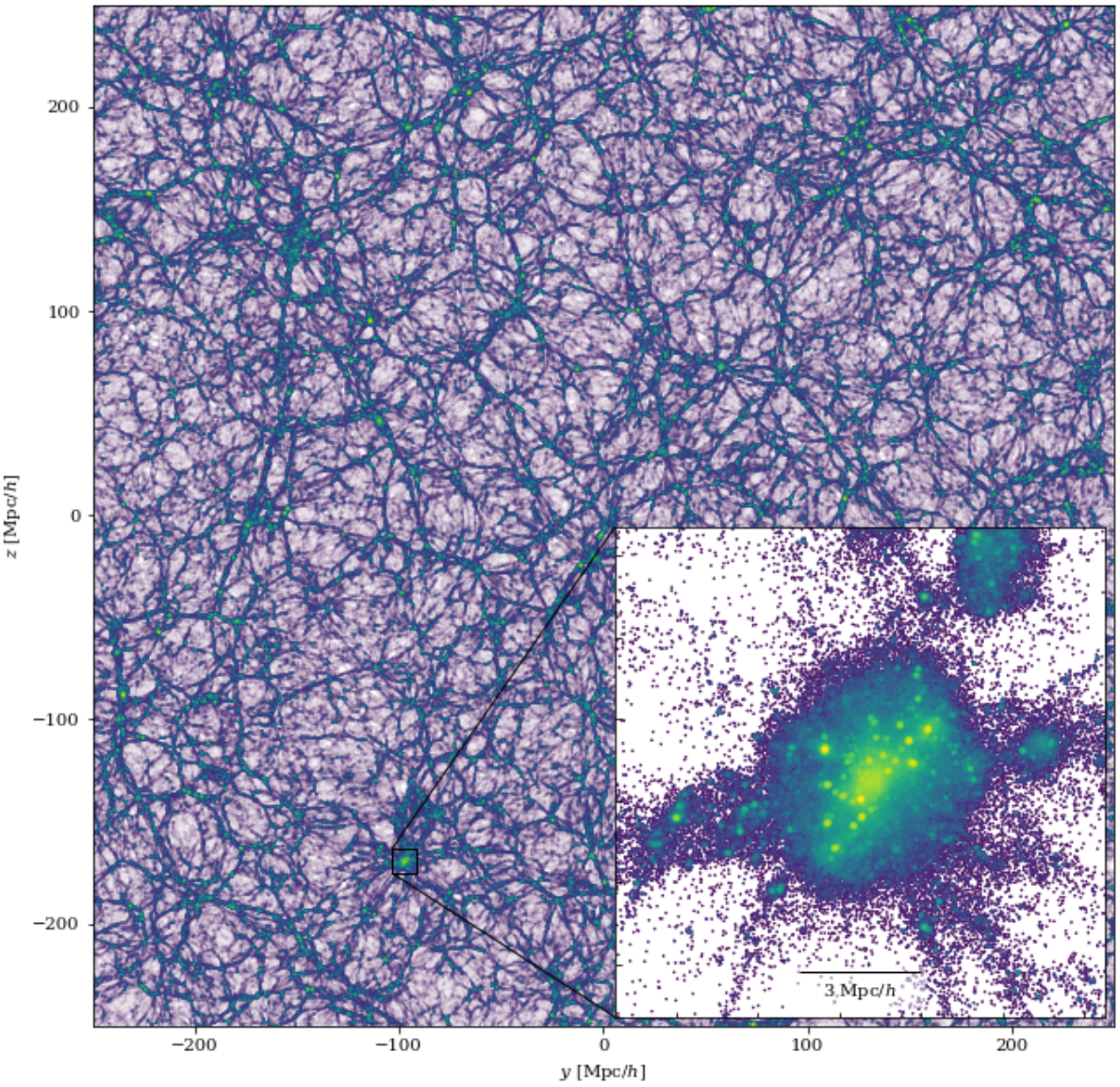

Figure 3.2: A 0.7 h^{-1} Mpc thick slice through an ABACUS realization of the S2016 box at $z = 0$. Color indicates projected surface density.

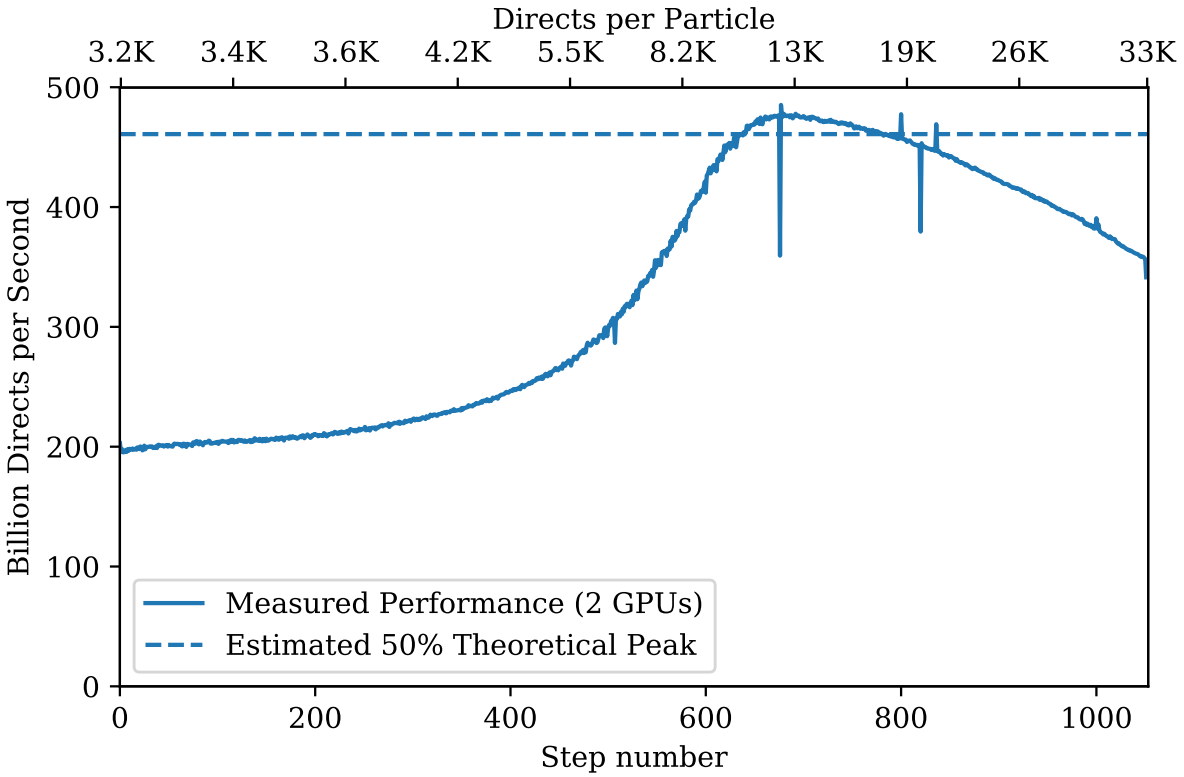

Figure 3.3: GPU performance for the near-field pairwise force computation. The theoretical maximum is computed assuming 10.6 TFLOPS per GPU and 23 FLOP per direct interaction (spline kernel). The latter is a conservative lower limit and we surmise that our peak performance is actually larger than the 52% of the ideal maximum shown here.

3.3 Code Comparison Results

3.3.1 Power Spectrum

We repeat the power spectra tests of S2016 on the $z = 0$ and $z = 2$ particles (except for GADGET3 $z = 2$ particles, which were unavailable) and add ABACUS's results. We use our own power spectrum code, which uses triangle-shaped (TSC) cloud mass assignment and TSC-alias window deconvolution (Jing 2005). We use a 3500^3 FFT mesh and find excellent agreement with the previously reported results. The ABACUS result lies between RAMSES and PKDGRAV3 at both $z = 2$ & $z = 0$ (Figures 3.4 & 3.5).

As observed in S2016, the codes do not agree even on the largest scales at both redshifts at the 0.5% level. This motivates us to check the analytic linear theory prediction of the power spectrum on these scales in Section 3.4.1. The largest disagreements are on the smallest scales, however. This motivates our checks of the effects of time step and softening in Sections 3.4.2 & 3.4.3, and our exploration of scale-free simulations in Appendix A.

3.3.2 Two-Point Correlation Function

We extend the analysis of S2016 to the small-scale two-point correlation function (2PCF). We use the CORRFUNC code (Sinha & Garrison 2017a) to measure the auto-correlation of the matter density field out to 1 h^{-1} Mpc. We first downsample the particles by a factor of two to reduce the pair-counting runtime.

The same trends that are visible in the small-scale power spectrum are visible in the

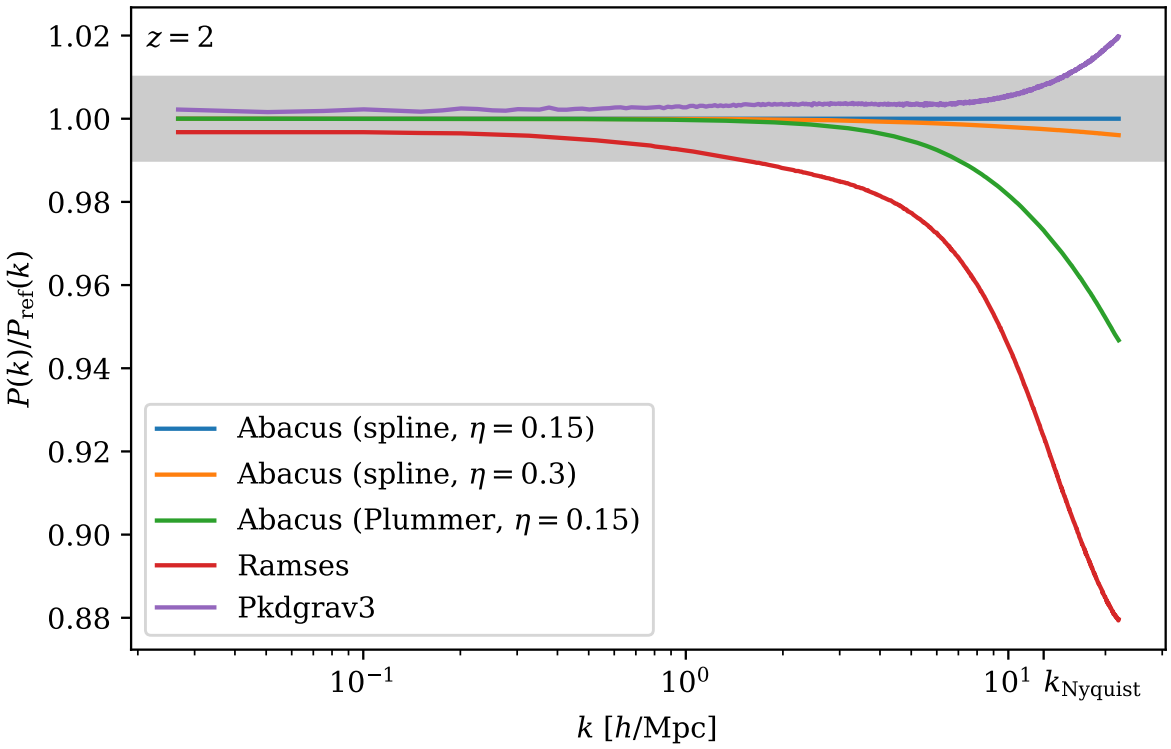

Figure 3.4: Comparison of power spectra at $z = 2$. GADGET3 particles were not available for this redshift.

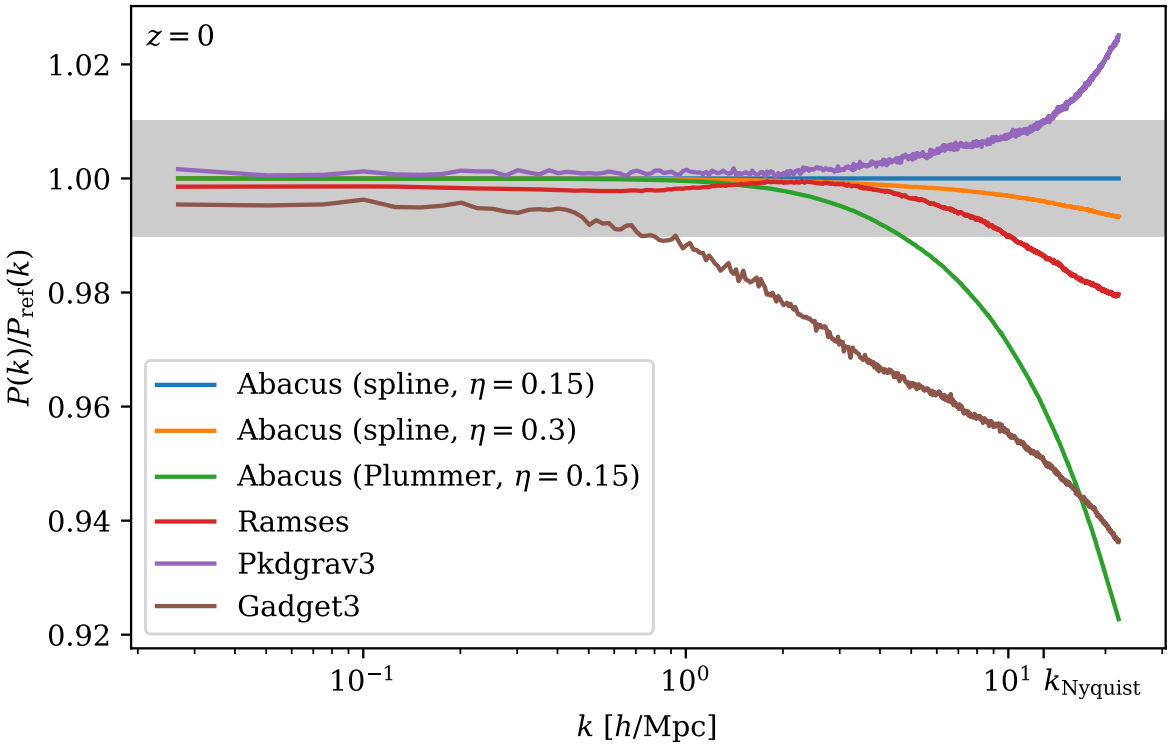

Figure 3.5: Same as Fig. 3.4 (comparison of power spectra) but at $z = 0$.

2PCF analysis (Figures 3.6 & 3.7). The main trend that is qualitatively different from the power spectrum analysis is that the RAMSES clustering amplitude exceeds that of ABACUS on the smallest scales. This may be related to differences in the softening model.

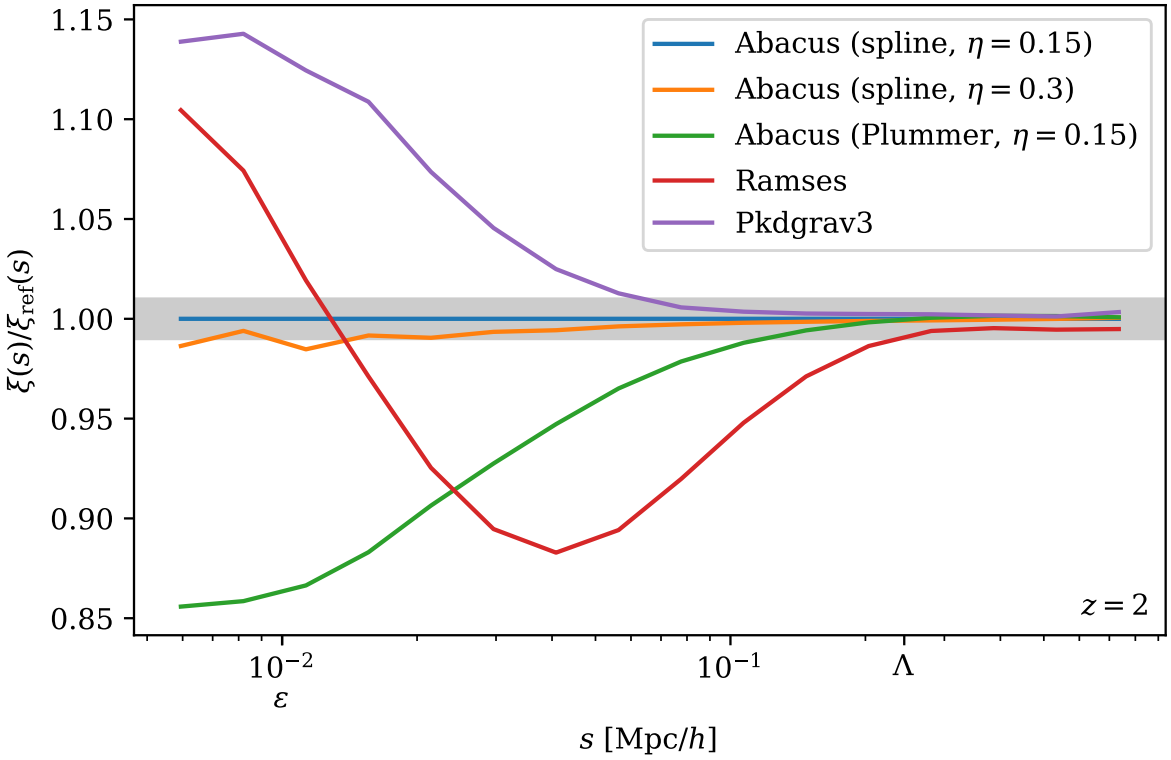

Figure 3.6: Comparison of 2PCF at $z = 2$. ε marks the softening length, and Λ marks the mean particle spacing. Only compressed outputs were retained for ABACUS particles at this redshift, so some noise is apparent on small scales. A larger binning was chosen to mitigate this effect. GADGET3 particles were not available for this redshift.

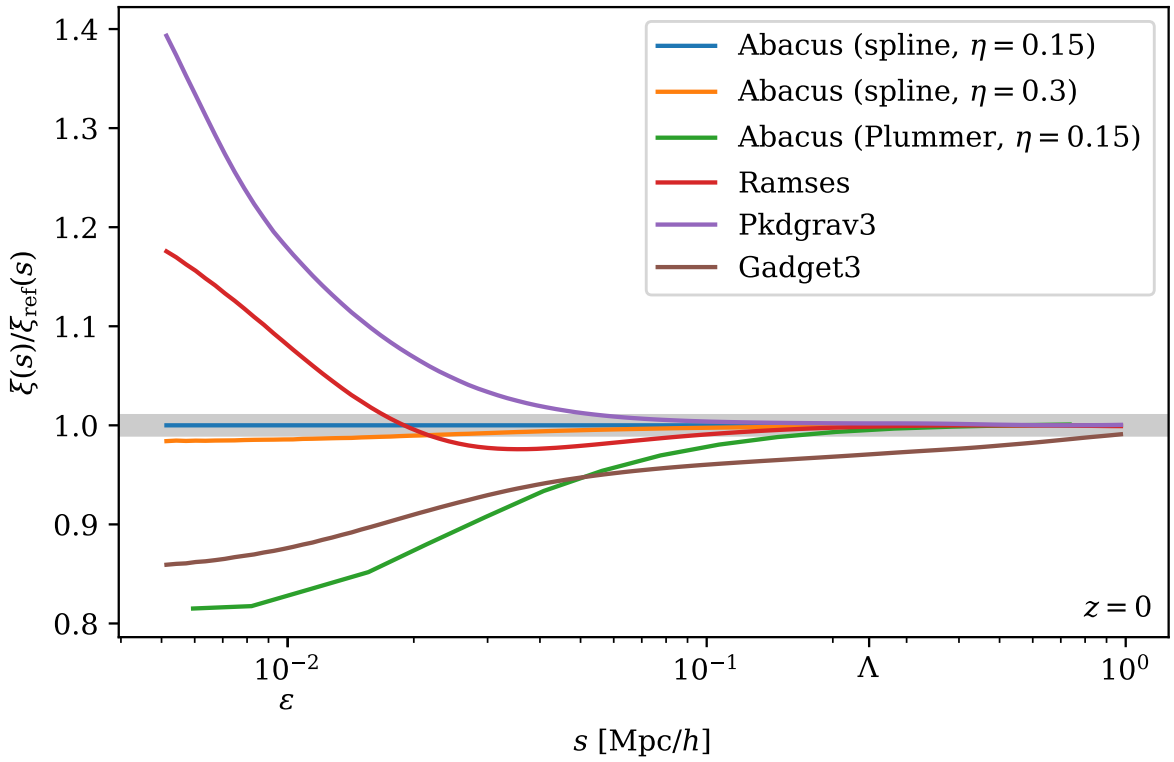

Figure 3.7: Same as Fig. 3.6 (comparison of 2PCF), but at $z = 0$. Full-precision outputs ABACUS outputs were available for all but the Plummer softening line, hence the coarser binning in that case.

3.4 Validation

In addition to the end-to-end, standalone tests described in Section 3.2 (the Ewald and homogenous lattice tests), we validate the accuracy of Abacus specifically in the context of the S2016 simulation. We test the recovery of linear theory, time step parameters, force softening model, and far-field and near-field accuracy.

3.4.1 Linear Theory

The S2016 codes do not agree on the power spectrum on the largest scales in the simulation. Most notably, RAMSES produces a 0.35% power deficit compared to ABACUS at $z = 2$, and GADGET3 produces a 0.45% deficit at $z = 0$. Motivated by this failure to agree on linear theory, we test ABACUS's ability to recover the analytic result in the strongly linear regime. We set up a 1024^3 particle simulation with a $\sigma_8 = 0.817/200$ at $z = 0$ and otherwise the same parameters as the S2016 simulation. In particular, we hold fixed parameters that could plausibly affect the accuracy, like the particles per cell and the multipole order. To best mimic S2016, we used the ordinary Zel'dovich Approximation [i.e. no corrections following Garrison et al. (2016)].

We evolve the simulation from $z_{\rm init} = 49$ to $z = 0$ using 137 time steps and compare the power spectrum at $z = 2$ and $z = 0$ to the linear power spectrum at those redshifts. In both cases, we find better than 0.01% agreement on the largest scales (Fig. 3.8). We find a deficit of power on smaller scales, towards $k_{\rm Nyquist}$. This is expected. An N-body system with finite particle mass should see a suppression of linear growth rate towards the Nyquist wavenumber of the particle sampling, independent of force softening or integration errors (Marcos et al. 2006; Garrison et al. 2016). The wavenumber scaling of this effect is an excellent match to the analytic prediction (dashed line, Fig. 3.8), using the $k \ll k_{\rm Nyquist}$ approximation of Marcos et al. (2006). We see that $z = 2$ consistently shows less suppression of power, as expected: the suppressed quantity is the growth *rate*, so a higher redshift gives less time for the suppression to accumulate in an absolute sense.

3.4.2 Time Stepping

Parametrization

ABACUS is presently a globally-stepped code; all particles share the same time step. At the beginning of each step, the time step Δa is chosen based on three criteria:

1. the step size in $\Delta \log(a)$ units must not exceed `TimeStepDlna`;
2. the step size must be less than `TimeStepAccel` times the maximum of $v_{\rm rms}/a_{\rm max}$ computed within each cell;
3. the step size times the maximum velocity must be less than 80% of a cell width.

The first criterion, set by `TimeStepDlna`, usually limits the step size at the beginning of the simulation before particle accelerations become large. It ensures integration accuracy even in the linear regime. We use a value of 0.03, or about 33 steps per e-fold of scale factor. The successful linear theory test in Section 3.4.1 uses the same value and thus validates this choice.

The second criterion, controlled by `TimeStepAccel` (also called η), becomes the limiting factor as soon as two particles anywhere in the simulation come into a close orbit. We nominally use a value of $\eta = 0.15$; we also try a value of $\eta = 0.3$ in Section 3.4, since we expect that our nominal choice is extremely conservative, given that we take 2200 global steps to $z = 0$ as a result.

In detail, for the second criterion we also compute a global $v_{\rm rms}/a_{\rm max}$ and use the maximum of the global and cell-based values. This protects us from taking catastrophically small time steps as a result of abnormally cold cells.

The third criterion simply ensures that particles drift by at most one slab per time step. This is necessary in order to keep the rolling window of slabs in memory small. In practice, we rarely trigger this criterion.

A large number of global steps is undesirable because (1) it increases the IO load, and (2) it wastes a large amount of computational effort integrating motions of low-acceleration particles. This problem is particularly noticeable in the S2016 simulation, which has a small softening length and thus a high contrast in the dynamical time scale of halo and void particles. We intend to address this in a future version of ABACUS with our "microstepping" scheme (see Section 3.2.1). A large number of global steps does have the benefit of minimizing integration errors in the particle dynamics, however, which is useful in the context of checking code convergence.

Validation

We investigate the effect of varying the time-stepping parameter η, or `TimeStepAccel`, on the matter field power spectrum and 2PCF. We try both $\eta = 0.15$ and $\eta = 0.3$. ABACUS takes 2206 and 1052 time steps, respectively, to $z = 0$. We find sub-percent differences in the power spectrum to the smallest scales we measure ($k = 22\ h\ \mathrm{Mpc}^{-1}$); the differences fall to 0.4% above $k_{\mathrm{Nyquist}} = \pi/\Lambda = 12.9\ h\ \mathrm{Mpc}^{-1}$, where $\Lambda = L/N^{1/3}$ is the mean interparticle spacing. The differences are even smaller at $z = 2$.

Similarly, we find very small differences of less than 1.6% across the whole measured range of the 2PCF at $z = 0$, which extends down to $\epsilon/2$, or $5\ h^{-1}\ \mathrm{kpc}$. This decreases to 1% at 2ϵ. Again, the differences are even smaller at $z = 2$.

In both metrics, the error caused by increasing the time step to $\eta = 0.3$ is smaller

by about an order of magnitude than the disagreements among the different codes. Thus, we consider our choice of $\eta = 0.3$ to be sufficiently accurate.

3.4.3 Softening

Parametrization

We investigate the effect of different force softening laws on our results. Our nominal results use spline softening, but we also perform a simulation with Plummer softening. In both cases, we use a Plummer-equivalent comoving softening length of $10\ h^{-1}\ \mathrm{kpc}$ (see below). We use a timestep parameter of $\eta = 0.15$, which is the finer of the two timestep criteria investigated above.

In Plummer softening (Plummer 1911), the $\mathbf{F}(\mathbf{r}) = \mathbf{r}/r^3$ force law is modified as

$$\mathbf{F}(\mathbf{r}) = \frac{\mathbf{r}}{(r^2 + \varepsilon_p^2)^{3/2}}, \tag{3.1}$$

where ε_p is the softening length. This softening is very fast to compute but is not compact, meaning it never explicitly switches to the exact r^{-2} form at any radius (in contrast with spline softening). This affects the growth of structure on scales much larger than ε_p, as we will see below.

Spline softening is an alternative in which the force law is softened for small radii but explicitly changes to the unsoftened form at large radii. Traditional spline implementations split the force law into three or more piecewise segments (e.g. the cubic spline of Hernquist & Katz 1989); we split only once for computational efficiency and to avoid code path branching[4]. We derive our spline implementation by considering a Taylor expansion

[4]We implement our split as a single `min` operation which compiles to a conditional move rather than

in r of Plummer softening (Eq. 3.1) and requiring a smooth transition at the softening scale up to the second derivative[5]. This gives

$$\mathbf{F}(\mathbf{r}) = \begin{cases} \left[10 - 15(r/\varepsilon_s) + 6(r/\varepsilon_s)^2\right] \mathbf{r}/\varepsilon_s^3, & r < \varepsilon_s; \\ \mathbf{r}/r^3, & r \geq \varepsilon_s. \end{cases} \tag{3.2}$$

This was first presented in Garrison et al. (2016).

The softening scales ε_s and ε_p imply different minimum dynamical times (an important property, as this sets the step size necessary to resolve orbits). We always choose the softening length as if it were a Plummer softening and then internally convert to a softening length that gives the same minimum pairwise dynamical time for the chosen softening method. For our spline, the conversion is $\varepsilon_s = 2.16\varepsilon_p$.

Comparison

In Figures 3.4 & 3.5, we see that Plummer softening produces a significant suppression of small-scale power. The range is notable too: 1% effects extend even to k below k_{Nyquist}, which itself is 24 times larger than the softening scale.

We see the same trend in the two-point correlation function in Figures 3.6 & 3.7: the suppression of clustering extends to many times the softening length.

In both the power spectrum and the 2PCF, using spline softening brings us into qualitatively better agreement with the other codes. Due to its compact nature, we

a costly conditional jump.

[5]A Taylor expansion in r^2 is also possible, but we discard that solution due to a large plateau of constant angular frequency near $r \sim 0$ that we worry might excite dynamical instabilities.

consider spline softening the more physically accurate of our two softening models.

3.4.4 Far-field Force

The main source of error in the far-field force is the finite multipole order p. Our nominal value is $p = 8$, which gives excellent force accuracy for near-field radius 2 (see Section 3.2). To quantify this in the context of S2016, we re-ran the final state with $p = 11$ and compared the far-field forces in the first 6 slabs to the $p = 8$ result. Measuring the fractional error as $|\mathbf{f_8} - \mathbf{f_{11}}|/|\mathbf{f_{11}}|$, we find the median error is 6.8×10^{-7}, with only 0.28% of forces worse than 1×10^{-4}.

3.4.5 Near-field Force

The near-field force is essentially exact, since it is computed via brute-force N^2 summation (i.e. no tree structures or other approximations are used). The main source of uncertainty is use of single-precision (32-bit) floating point values for the positions and accelerations. This mainly enters as round-off error in the accumulation of the accelerations, but there are other intermediate steps, like the re-centering of particle positions before they are sent to the GPU, that may suffer similarly. To quantify these effects, we make a copy of the final state in double precision, evaluate the forces, and compare the forces to the single-precision answer. We find that only 0.038% of force errors are worse than 1×10^{-4}, which is an order of magnitude fewer than in the far-field. The median fractional error is 7.5×10^{-7}.

3.5 Discussion

We have presented a realization of the S2016 code comparison simulation using our ABACUS N-body code. ABACUS has excellent force accuracy properties that give us confidence that we are recovering the correct answer on most scales, especially in the linear regime where other codes disagree on the answer. Indeed, our linear evolution tests show better than 0.01% recovery of linear theory growth. We have validated our time step and force accuracy parameters and found them to be conservative.

On small scales, the answer still depends on the choice of softening model. Even matching dynamical times, we find that Plummer softening produces a significant suppression of small-scale power. We consider this non-physical and prefer our compact spline softening, which brings our small-scale results closer to that of RAMSES and PKDGRAV3. GADGET3 is still somewhat of an outlier, generally missing power across a broad range of scales. Part of this could well have to do with softening model differences, given the large effect we saw when switching from Plummer to spline. It may be illuminating to compare these results to those of other codes like HACC (Habib et al. 2013) and 2HOT (Warren 2013) to determine which differences arise from the force-solving technique and which arise from the softening model.

Another approach to determining the correct solution in the non-linear regime is through scale-free simulations. In the scale-free framework, a simulation is executed with a power-law initial power spectrum and an $\Omega_m = 1$ background cosmology. The matter clustering is expected to be self-similar—early-time, small-scale clustering should be a rescaling of late-time, large-scale clustering. Deviation from this behavior is non-physical, but it is also expected since N-body simulations include non-physical scales such as soft-

ening. We explore this behavior in Appendix A.

In this work, we have also demonstrated ABACUS's performance, which exceeds 30 million particles per second per step until $z = 1.1$. Afterwards, the near-field computation slows down due to the amount of clustering at this particle mass. Overall, we achieve a mean rate of 23 Mp/s and measure GPU performance of over 50% of the peak theoretical FLOPS. Future enhancements to ABACUS will substantially increase our performance at this force and mass resolution.

Acknowledgements

We would like to thank Doug Ferrer and Marc Metchnik as co-authors of ABACUS, and Nina Maksimova for assistance in building `hal`, the computer used to run these simulations. We also would like to thank Aurel Schneider and Doug Potter for providing the S2016 particle snapshots, and the referee for helpful comments. This work has been supported by grant AST-1313285 from the National Science Foundation and by grant DE-SC0013718 from the U.S. Department of Energy. DJE is further supported as a Simons Foundation investigator.

Table 3.1—Continued

Action	Time [s]	%	Rate	Notes
Multipole Write	27.6	15 %	1.9 GB/s	108 GB written
convolution	68	27 %		All work is CPU
Array Swizzle	19	28 %		
Convolution Arithmetic	19	28 %	5.8×10^9	DP-FLOPS/core
z-FFT	10	15 %		
Inverse z-FFT	10	15 %		
Wait for IO	10	15 %		

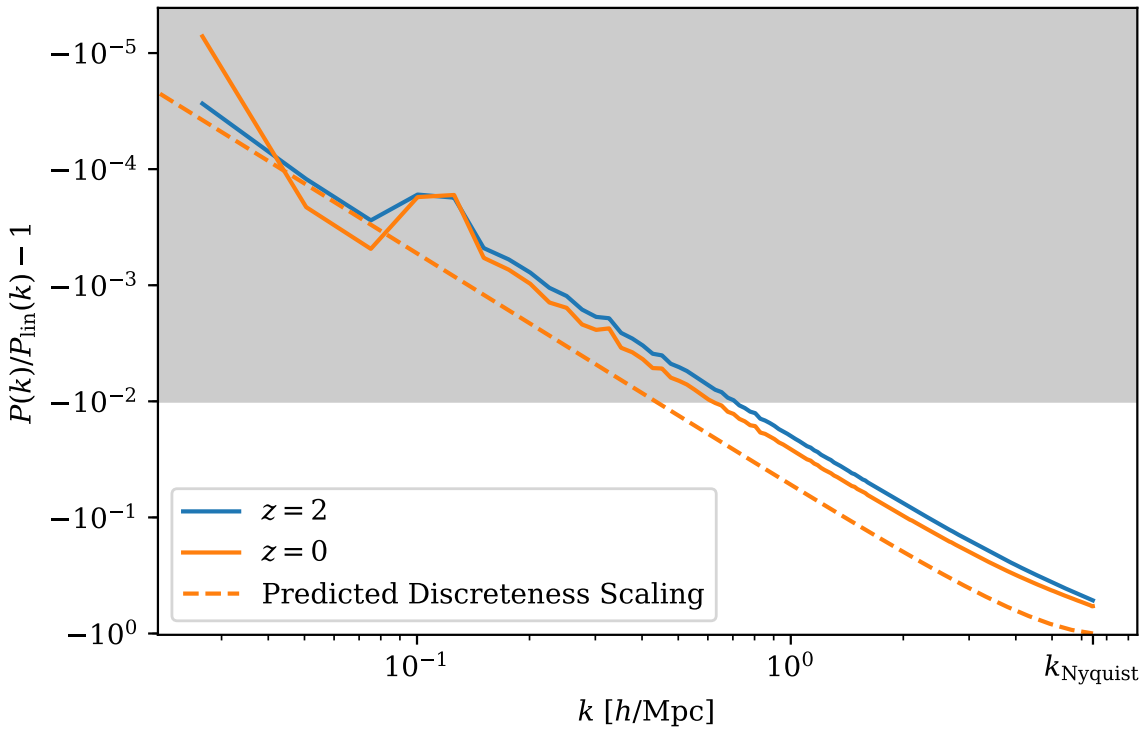

Figure 3.8: Test of evolution the deeply linear regime ($\sigma_8 = 0.817/200$ at $z = 0$). ABACUS executed 137 time steps from $z_{\rm init} = 49$ to $z = 0$ and recovers the analytic linear theory prediction with excellent accuracy. The dashed line shows the predicted scaling of the suppression of growth rate due to discreteness, or finite particle mass, on the power spectrum.

Chapter 4

Improving Initial Conditions for Cosmological N-Body Simulations

This thesis chapter was originally published as

Garrison, Lehman H., Daniel J. Eisenstein, Douglas Ferrer, Marc V. Metchnik, Philip A. Pinto *2016, MNRAS, 461, 4125*

Abstract

In cosmological N-body simulations, the representation of dark matter as discrete "macroparticles" suppresses the growth of structure, such that simulations no longer reproduce linear theory on small scales near $k_{\rm Nyquist}$. Marcos et al. demonstrate that this is due to sparse sampling of modes near $k_{\rm Nyquist}$ and that the often-assumed continuum growing modes are not proper growing modes of the particle system. We develop initial conditions that respect the particle linear theory growing modes and then rescale the mode

amplitudes to account for growth suppression. These ICs also allow us to take advantage of our very accurate N-body code Abacus to implement 2LPT in configuration space. The combination of 2LPT and rescaling improves the accuracy of the late-time power spectra, halo mass functions, and halo clustering. In particular, we achieve 1% accuracy in the power spectrum down to $k_{\rm Nyquist}$, versus $k_{\rm Nyquist}/4$ without rescaling or $k_{\rm Nyquist}/13$ without 2LPT, relative to an oversampled reference simulation. We anticipate that our 2LPT will be useful for large simulations where FFTs are expensive and that rescaling will be useful for suites of medium-resolution simulations used in cosmic emulators and galaxy survey mock catalogs. Code to generate initial conditions is available at `https://github.com/lgarrison/zeldovich-PLT`.

4.1 Introduction

Cosmological N-body simulations are the state-of-the-art tool for predicting dark matter halo clustering and masses for a given cosmology. In most cosmological models, a large fraction of mass is in the form of dark matter and thus behaves as a collisionless "fluid" well described by the coupled Vlasov-Poisson equations. N-body simulations take a discrete "macroparticle" sampling of this underlying fluid and then treat the evolution of the particle system as tracing the evolution of the fluid system (for an alternative phase-space formulation of this problem, see Hahn et al. 2013). The applicability of results derived from N-body simulations thus assumes correspondence between the particle system and the fluid system.

This assumption has a number of well-documented violations (collectively known as "discreteness effects") at very early and very late times, such as correlations induced

by the initial particle sampling (see Joyce & Marcos 2007b), and two-body relaxation (e.g., Binney & Knebe 2002; El-Zant 2006). A third, intermediate regime has received less attention, however: the evolution of the N-body system from its initial configuration up to the mildly non-linear regime. The work of Marcos et al. (2006) showed that in this regime, the assumptions of fluid linear theory are strongly violated on small scales. Based on their work, we seek to correct these small-scale effects by modifying our initial conditions to respect the proper growing modes of the simulation and compensate for missing growth. This directly addresses the improper growth of modes on small scales that Warren (2013) identified as the dominant systematic error in precision halo mass functions.

The underlying theory, developed by Marcos et al. (2006), is called *particle linear theory* (PLT). PLT is an analytical description of the evolution of a grid-like particle system that self-interacts under a $1/r^2$ force law. It is a perturbative solution to the fully discrete cosmological N-body problem, derived from a linearization of the force from a perfect cubic periodic lattice[1] using the *dynamical matrix* formalism well known in solid state physics (see §4.2). As long as its perturbative assumption is satisfied, PLT fully describes the particle positions and velocities as a function of time (or redshift). This allows analysis of discreteness effects by comparing the particle behavior for finite N to the limit $N \to \infty$.

The authors of PLT have used their theory to quantify discreteness effects from the linear and weakly non-linear regimes (Joyce & Marcos 2007a) to the fully non-linear regime

[1]It is not limited to this case; the framework is equally valid for any Bravais lattice, such as body-centered and face-centered lattices. We will focus on the simple cubic case, however.

(Joyce et al. 2009). However, PLT has not yet been used to improve the initial conditions of simulations. In this work, we develop PLT-based corrections to the initial conditions that eliminate transients due to the initial grid configuration of the particles. Additionally, we develop a fast and powerful new approach to second-order Lagrangian perturbation theory (2LPT) that does not rely on large Fourier transforms, and we demonstrate its accuracy by performing the actual particle evolution from $z = 4999$ with our extremely precise N-body code Abacus. We compare our answer to that of a well-known 2LPT code and find excellent agreement on all but small scales, where we expect differences due to the different assumptions inherent in our approaches.

Broadly speaking, simulations must produce power spectra and halo properties accurate to 1% to support current and upcoming galaxy surveys (e.g. Tinker et al. 2008; Weinberg et al. 2013). Specifically, projects like the DES (Frieman & Dark Energy Survey Collaboration 2013), LSST (LSST Dark Energy Science Collaboration 2012), and *Euclid* (Cimatti et al. 2009; Laureijs 2009) are projected to require 1% accuracy in the matter power spectrum to $k = 10h\ \mathrm{Mpc}^{-1}$ (Schneider et al. 2016a). These stringent demands are our motivation for careful examination and improvement of the initial conditions on small scales.

In §4.2, we review the formalism of PLT. Then, in §4.3, we discuss our application of PLT to the initial conditions of cosmological simulations and quantify the improvements. In §4.4, we develop our new approach to 2LPT and test its accuracy. Finally, in §4.5, we discuss the implications of our findings for cosmological measurables derived from N-body simulations (halo masses, clustering, and power spectra), and summarize our results in §4.6.

4.2 Particle Linear Theory

Here, we review particle linear theory (PLT) as developed by Marcos et al. (2006) (see also Joyce & Marcos 2007a; Joyce et al. 2005). PLT gives the analytical evolution of a slightly perturbed lattice of self-gravitating particles, which is precisely the initial configuration of many cosmological simulations. We will emphasize the ways in which this evolution diverges from that of the corresponding fluid system.

4.2.1 PLT formalism

Consider a simple cubic lattice of N equal-mass particles in a box of side length L with periodic boundary conditions. In an expanding universe, the equation of motion is

$$\ddot{\mathbf{x}}_i + 2H(t)\dot{\mathbf{x}}_i = -\frac{1}{a^3}\sum_{i\neq j}\frac{Gm_j(\mathbf{x}_i-\mathbf{x}_j)}{|\mathbf{x}_i-\mathbf{x}_j|^3}, \tag{4.1}$$

where $\mathbf{x}_i$ is the comoving position of particle i, m_i is its mass, and G is the universal gravitational constant. $\mathbf{x}_i$ is related to the physical position by $\mathbf{r}_i = a(t)\mathbf{x}_i$, where $a(t)$ is the cosmological scale factor and $H(t) = \dot{a}/a$ is the corresponding Hubble factor.

If we label the original lattice site corresponding to particle i with its comoving position $\mathbf{R}_i$, then we may write the displacement of particle i from $\mathbf{R}_i$ as $\mathbf{u}(\mathbf{R}_i)$. Thus, the full comoving position of a particle is given by $\mathbf{x}_i(t) = \mathbf{R}_i + \mathbf{u}(\mathbf{R}_i)$. Following the convention in PLT, we generally drop the subscript i from $\mathbf{R}_i$.

The right side of Eq. 4.1 can be expanded at linear order in the relative displacements

of particle pairs[2] to yield

$$\ddot{\mathbf{u}}(\mathbf{R},t) + 2H\mathbf{u}(\mathbf{R},t) = -\frac{1}{a^3}\sum_{\mathbf{R}'} \mathscr{D}(\mathbf{R}-\mathbf{R}')\mathbf{u}(\mathbf{R}',t). \tag{4.2}$$

The matrix $\mathscr{D}(\mathbf{R})$ is known in solid state physics as the *dynamical matrix* (Pines 1964). For a given $\mathbf{R}$, $\mathscr{D}(\mathbf{R})\mathbf{u}(\mathbf{R})$ is the force induced at the origin by a particle at $\mathbf{R}$ displaced by $\mathbf{u}(\mathbf{R})$. Specifically, the dynamical matrix can be written as

$$\mathscr{D}_{\mu\nu}(\mathbf{R}\neq\mathbf{0}) = Gm\left(\frac{\delta_{\mu\nu}}{R^3} - 3\frac{R_\mu R_\nu}{R^5}\right) \tag{4.3}$$

$$\mathscr{D}_{\mu\nu}(\mathbf{0}) = -\sum_{\mathbf{R}\neq\mathbf{0}} \mathscr{D}_{\mu\nu}(\mathbf{R}), \tag{4.4}$$

where $\delta_{\mu\nu}$ is the Kronecker delta. The second equation is a statement of Newton's third law. $\mathscr{D}$ cannot be computed as simply as these expressions suggest, however, because there is an implicit sum over infinite periodic copies. Ultimately, this means one must either compute $\mathscr{D}$ using an Ewald-type summation (as in Marcos et al. 2006) or with a very precise N-body force solver, as we use (see §4.2.3).

In Eq. 4.2, $\mathscr{D}$ acts as a convolution kernel acting on the displacements, and thus it is not surprising that it has a natural action in Fourier space. If we define the discrete Fourier transform and its inverse[3] as

$$\tilde{\mathbf{u}}(\mathbf{k},t) = \sum_{\mathbf{R}} e^{-i\mathbf{k}\cdot\mathbf{R}}\mathbf{u}(\mathbf{R},t) \tag{4.5}$$

$$\mathbf{u}(\mathbf{R},t) = \frac{1}{N}\sum_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{R}}\tilde{\mathbf{u}}(\mathbf{k},t), \tag{4.6}$$

[2]When expanding the force in a Taylor series, one finds that each term in the sum over lattice sites $\mathbf{R}'$ has its own convergence criterion: $|\mathbf{R}-\mathbf{R}'| > |\mathbf{u}(\mathbf{R})-\mathbf{u}(\mathbf{R}')|$. This lends some robustness to the expansion, because even as particles move and some particle pairs start to violate this condition, many others may continue to satisfy it and thus still produce a useful approximation of the total force.

[3]See Marcos et al. (2006) for subtleties regarding the summation limits.

then we may write the equation of motion (Eq. 4.2) as

$$\ddot{\tilde{\mathbf{u}}}(\mathbf{k},t) + 2H(t)\dot{\tilde{\mathbf{u}}}(\mathbf{k},t) = -\frac{1}{a^3}\tilde{\mathscr{D}}(\mathbf{k})\tilde{\mathbf{u}}(\mathbf{k},t). \tag{4.7}$$

We define $\tilde{\mathscr{D}}$ as the Fourier transform of $\mathscr{D}$, in analogy with Eq. 4.6. From the symmetry properties of $\mathscr{D}(\mathbf{R})$, $\tilde{\mathscr{D}}(\mathbf{k})$ must be a real, symmetric operator with three orthogonal eigenvectors $\mathbf{e}_n(\mathbf{k})$ and eigenvalues $\omega_n^2(\mathbf{k})$.

Because the eigenvectors of $\mathscr{D}$ form a complete basis at every $\mathbf{k}$, we can project an arbitrary displacement field onto the basis $\hat{\mathbf{e}}_n(\mathbf{k})$. Or, as we discuss in §4.3.1, we can construct a displacement field that consists of one eigenmode at every $\mathbf{k}$. For now, we will discuss the evolution of an arbitrary displacement field from initial conditions $\mathbf{u}(\mathbf{R},t_0)$ and $\dot{\mathbf{u}}(\mathbf{R},t_0)$.

We can represent the Fourier space evolution of $\tilde{\mathbf{u}}(\mathbf{k},t)$ as a sum of the independent evolution of each eigenmode[4]:

$$\begin{aligned}\mathbf{u}(\mathbf{k},t) = \sum_{n=1}^{3} & U_n(\mathbf{k},t)\left[\hat{\mathbf{e}}_n(\mathbf{k})\cdot\tilde{\mathbf{u}}(\mathbf{k},t_0)\right]\hat{\mathbf{e}}_n(\mathbf{k}) \\ & + V_n(\mathbf{k},t)\left[\hat{\mathbf{e}}_n(\mathbf{k})\cdot\dot{\tilde{\mathbf{u}}}(\mathbf{k},t_0)\right]\hat{\mathbf{e}}_n(\mathbf{k}).\end{aligned} \tag{4.8}$$

The functions $U_n(\mathbf{k},t)$ and $V_n(\mathbf{k},t)$ can be found by substituting the above equation into Eq. 4.7, with the boundary conditions

$$\begin{aligned} U_n(\mathbf{k},t_0) &= 1, & \dot{U}_n(\mathbf{k},t_0) &= 0, \\ V_n(\mathbf{k},t_0) &= 0, & \dot{V}_n(\mathbf{k},t_0) &= 1. \end{aligned} \tag{4.9}$$

Replacing $\tilde{\mathscr{D}}\hat{\mathbf{e}}_n$ by $\omega_n^2\hat{\mathbf{e}}_n$, one finds

$$\ddot{f} + 2H\dot{f} = -\frac{\omega_n^2(\mathbf{k})}{a^3}f, \tag{4.10}$$

[4]Recall that modes at different wavevectors evolve independently in linear theory.

to which $U_n(\mathbf{k},t)$ and $V_n(\mathbf{k},t)$ are the solutions.

The exact form of U_n and V_n depends on the cosmology. In ΛCDM, the early universe is well described by an Einstein-deSitter ($\Omega_m = 1$) cosmology, with scale factor $a(t) \propto t^{2/3}$ and Hubble constant $H(t) = 2/3t$. Since the small-displacement (and thus early-time) regime is exactly what we are considering here, EdS is a good approximation to ΛCDM. Thus, we have

$$U_n(\mathbf{k},t) = \tilde{\alpha}(\mathbf{k}) \left[\alpha_n^+(\mathbf{k}) \left(\frac{t}{t_0}\right)^{\alpha_n^-(\mathbf{k})} + \alpha_n^-(\mathbf{k}) \left(\frac{t}{t_0}\right)^{-\alpha_n^+(\mathbf{k})} \right] \tag{4.11a}$$

$$V_n(\mathbf{k},t) = \tilde{\alpha}(\mathbf{k}) t_0 \left[\left(\frac{t}{t_0}\right)^{\alpha_n^-(\mathbf{k})} - \left(\frac{t}{t_0}\right)^{-\alpha_n^+(\mathbf{k})} \right] \tag{4.11b}$$

where

$$\tilde{\alpha}(\mathbf{k}) = \frac{1}{\alpha_n^-(\mathbf{k}) + \alpha_n^+(\mathbf{k})} \tag{4.12}$$

and

$$\alpha_n^-(\mathbf{k}) = \frac{1}{6}\left[\sqrt{1+24\varepsilon_n(\mathbf{k})} - 1\right] \tag{4.13a}$$

$$\alpha_n^+(\mathbf{k}) = \frac{1}{6}\left[\sqrt{1+24\varepsilon_n(\mathbf{k})} + 1\right]. \tag{4.13b}$$

In these expressions, $\varepsilon_n(\mathbf{k})$ are the normalized eigenvalues, given by

$$\varepsilon_n(\mathbf{k}) \equiv -\frac{\omega_n^2(\mathbf{k})}{4\pi G\rho_0}. \tag{4.14}$$

This completes the PLT description of the evolution of an arbitrary displacement and velocity field in an EdS universe, up to the numerical computation of the eigenmodes of $\mathscr{D}$ (see §4.2.3).

4.2.2 Discreteness effects and the fluid limit

We now have a quantitative framework in which to compare particle lattice evolution to the evolution of the equivalent fluid system. Namely, we can compare the behavior of wave modes on the lattice to wave modes in the fluid system. We discuss two ways in which discreteness manifests: deviation of the eigenvalue spectrum from unity, and deviation of the longitudinal eigenvectors from $\hat{\mathbf{k}}$.

In Fig. 4.1, all three eigenvalues are plotted for every $\mathbf{k}$. If the lattice behaved as a fluid, two of the eigenvalues would be 0 and one would be 1 at every $\mathbf{k}$. The two null eigenvalues correspond to transverse modes, or modes with zero divergence and non-zero curl that do not source forces in fluid theory, so their deviation from 0 in Fig. 4.1 is purely an artefact of discreteness. The eigenvalue of 1 corresponds to a longitudinal mode that produces density perturbations that source a force directly proportional to the overdensity. The presence of $\varepsilon_n > 1$ corresponds to an "overdriven" mode that collapses faster than the fluid limit, while $\varepsilon_n < 1$ is an "underdriven" mode that collapses more slowly. A mode with $-1/24 < \varepsilon_n < 0$ is purely decaying; $\varepsilon_n < -1/24$ is oscillatory. Note that the eigenvalues converge to either 1 or 0 as $|\mathbf{k}| \to 0$, which is a reflection of the fact that we recover the fluid behavior in the limit of a well-sampled mode.

The orientation of modes explains why some modes are overdriven and some are underdriven for the same $|\mathbf{k}|$. Modes aligned with the grid axes collapse faster than those skew to them. This orientation dependence is a direct violation of isotropy.

Due to the fact that some modes are consistently underdriven and some overdriven, we would expect accumulation of this effect over time. In Fig. 4.2, we plot the magnitude of this effect, averaged over mode orientations. This plot illustrates one of the most surprising

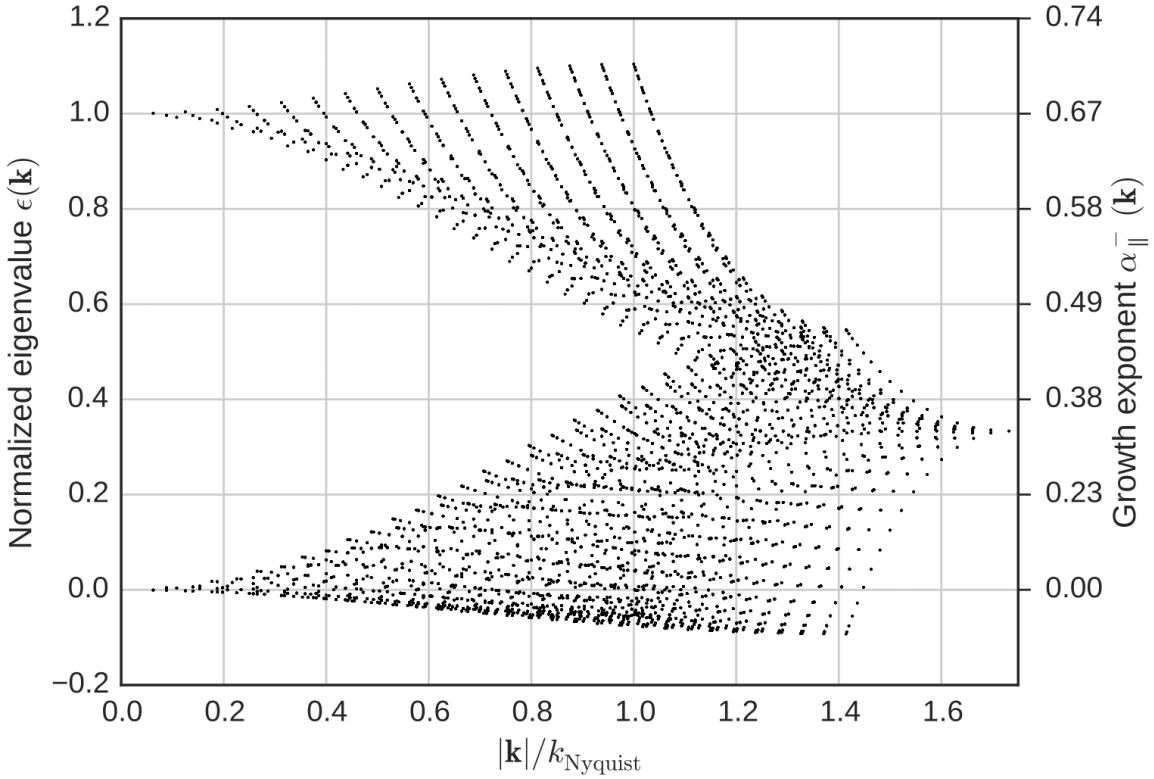

Figure 4.1: Eigenvalue spectrum for a 32^3 particle simple cubic lattice. Eigenvalues of 1 and 0 correspond to fluid behavior for longitudinal and transverse modes, respectively (Eq. 4.14). The corresponding growing mode exponent is labeled on the right axis (Eqs. 4.13 & 4.18), where $2/3$ is the nominal fluid linear theory value. Compare with Joyce & Marcos (2007a) fig. 1.

and important results of Marcos et al. (2006), which is that the power spectrum at a fixed redshift of a particle system *diverges* from the fluid limit as $z_{\rm init} \to \infty$, because an earlier starting time means more time for these non-fluid effects to build up[5]. This divergence is particularly important to note because standard practice is to increase $z_{\rm init}$ and claim that the results are representative of the desired fluid behavior. While this achieves the goal of reducing higher-order effects, the fact that the first-order results diverge from the fluid limit is often neglected. Increasing the redshift as a test of convergence is only a safe procedure when considering scales much larger than the interparticle spacing. The correct way, then, to test for convergence on intermediate and small scales is to increase the particle density while keeping the initial redshift fixed and compare the results at the same wavenumber. This is the procedure we employ throughout this work. Additionally, we attempt to make a correction for this effect by modifying the initial conditions, which we describe in §4.3.3.

The second manifestation of discreteness in PLT is in the eigenvectors of the dynamical matrix. In fluid theory, only longitudinal (compressional) modes feel forces because they are the only modes that produce density contrasts. This corresponds to the eigenmodes

$$\begin{aligned} \hat{\mathbf{e}}_1 &= \hat{\mathbf{k}}; & \varepsilon_1 &= 1; \\ \hat{\mathbf{e}}_2 &= \hat{\mathbf{k}}_{2\perp}; & \varepsilon_2 &= 0; \\ \hat{\mathbf{e}}_3 &= \hat{\mathbf{k}}_{3\perp}; & \varepsilon_3 &= 0; \end{aligned} \tag{4.15}$$

where $\hat{\mathbf{k}}_{2\perp}$ and $\hat{\mathbf{k}}_{3\perp}$ are chosen such that $\{\hat{\mathbf{e}}_n\}$ forms an orthogonal basis. In PLT, all

[5]This prediction of suppression of small-scale power with increasing initial redshift has been borne out in empirical tests, e.g. L'Huillier et al. (2014).

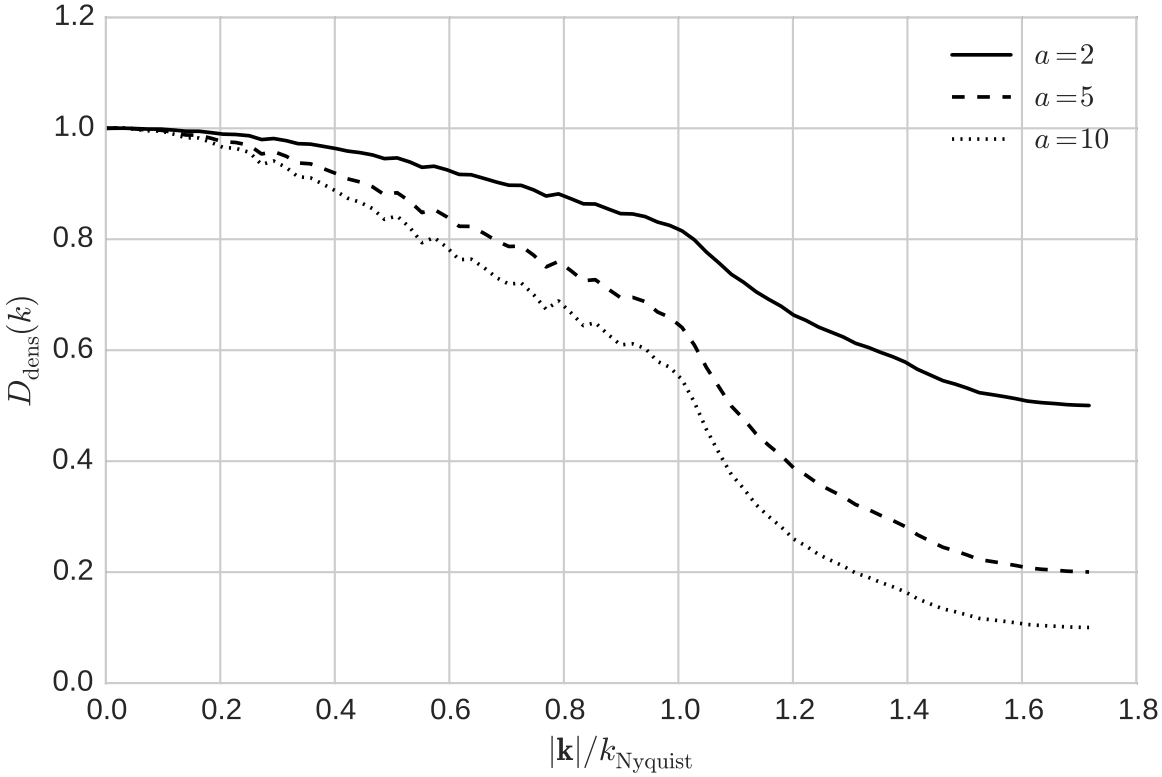

Figure 4.2: Discreteness factor D_{dens} for a 64^3 particle simple cubic lattice, averaged in bins of $|\mathbf{k}|$. This gives the ratio of the density power spectrum in PLT to fluid theory as a function of scale factor (for $a_{\mathrm{init}} = 1$) and wavenumber. Compare with Joyce & Marcos (2007a) fig. 3, but note the differences due to our definition of D_{dens}.

three eigenvectors generically have non-zero eigenvalues and the longitudinal eigenvector $\hat{\mathbf{e}}_1 \neq \hat{\mathbf{k}}$. Since simulations are nearly always initialized with purely $\hat{\mathbf{k}}$ modes, this manifests as forces misaligning with displacements. This introduces vorticity that should eventually decay relative to the growing mode, but such effects do not disappear quickly, especially in higher-order statistics (Scoccimarro 1998). See §4.3.1 for our correction of this effect. The nearly-perfect alignment of forces and displacements that we achieve is also important for our implementation of second-order Lagrangian perturbation theory corrections (see §4.4).

4.2.3 Numerical computation of dynamical matrix

Eq. 4.8 gives the analytical particle evolution in PLT but depends on knowing the eigenvectors $\hat{\mathbf{e}}_n(\mathbf{k})$ and eigenvalues $\omega_n^2(\mathbf{k})$ of $\tilde{\mathscr{D}}(\mathbf{k})$, which must be calculated numerically. Marcos et al. (2006) compute the spectrum with a custom Ewald summation method, which works by decomposing the gravitational potential into near-field and far-field components. The former is summed in configuration space and the latter in Fourier space, since they converge quickly in those respective spaces. The sums are truncated when the series is determined to have converged. The potential yields $\mathscr{D}(\mathbf{R})$, which is then is Fourier transformed to $\tilde{\mathscr{D}}(\mathbf{k})$. Recall that $\tilde{\mathscr{D}}(\mathbf{k})$ is a 3×3 matrix at every $\mathbf{k}$, so the determination of the $3N$ eigenvectors and eigenvalues reduces to N 3×3 matrix diagonalizations, which can be done with any numerical linear algebra package.

Rather than build a custom Ewald summer, we take advantage of the high force accuracy of our N-body code ABACUS (see §4.2.4) and calculate $\mathscr{D}$ in the following manner:

1. Generate a uniform grid of N particles;

2. Displace one particle by a small fraction of the interparticle spacing (10^{-5} is sufficiently small) along the x-axis;

3. Measure the force induced on all other particles by this displaced particle and call this field $\mathbf{F}_{+x}(\mathbf{R})$;

4. Displace the particle by the same amount in the $-x$ direction;

5. Measure the force and call this field $\mathbf{F}_{-x}(\mathbf{R})$;

6. Add the forces to cancel second-order effects: $\frac{1}{2}(\mathbf{F}_{+x}(\mathbf{R}) - \mathbf{F}_{-x}(\mathbf{R}))/10^{-5}$ is one row of $\mathscr{D}(\mathbf{R})$;

7. Repeat steps (ii) – (vi) for the y- and z-axes.

Having formed $\mathscr{D}(\mathbf{R})$, we can now proceed exactly as before to calculate $\tilde{\mathscr{D}}(\mathbf{k})$ and its eigenmodes. In practice, we do not displace the particle along the y- and z-axes. Instead, we permute the indices of the x result to obtain the y and z results. Furthermore, the eigenvalues and eigenvectors vary smoothly below k_{Nyquist}, so rather than generate new eigenmodes for every different N, we generate one $N = 128^3$ grid and trilinearly interpolate to finer grids on-the-fly.

4.2.4 ABACUS: N-body cosmology to machine precision

Throughout this work, we employ the N-body code ABACUS, described in Chapter 2 of this thesis. In this application, the most enabling aspect of ABACUS is its ability to compute near-machine-precision forces while maintaining competitive speeds.

In Table 4.1, we define two sets of parameters that we will refer to as "normal precision" and "high precision" throughout the rest of this work. We use high precision for evaluation of the dynamical matrix in §4.2.3 and most of the tests of the correctness of our methods in §4.3 & §4.4 (that is, the 64^3 and 256^3 particle simulations). We note the exceptions as they occur, which are generally for simulations to low z. In high precision, the maximum force error reaches nearly machine precision, but is more expensive to evaluate (and the lack of softening makes it unsuitable for low-z applications). In our 720^3 and 1440^3 cosmology simulations in §4.5, we use normal precision. In computing the dynamical matrix, the high precision multipole `Order` and `Precision` are both necessary for an accurate determination of the eigenmodes.

The exceptional force accuracy of Abacus enables us to carry out the precise testing of the initial conditions in the following sections.

4.3 Corrections to Initial Conditions

In this section and the next, we discuss four applications of the above theory to improve the initial conditions of cosmological simulations: (i) initializing every mode in the simulation to a single eigenmode at every $\mathbf{k}$; (ii) correcting the velocities of every mode to eliminate decaying-mode transients; (iii) rescaling the initial displacement amplitudes such that the power spectrum will match the linear prediction at a later time; and (iv) calculating second-order Lagrangian perturbation theory corrections using a novel in-place scheme. The last correction we defer to the next section, as its derivation is independent of PLT.

Table 4.1:: ABACUS code options

Parameter	Normal precision	High precision
`SofteningLength`	1/8 particle spacing	0
Softening technique	Plummer	None
`NearFieldRadius`	2	3
Multipole `Order`	8	16
`Precision`	32-bit	64-bit
Max force error	1×10^{-4}	2×10^{-8}
Median force error	2×10^{-6}	4×10^{-11}

NOTES – The force error is the maximum fractional error on a set of 2^{16} uniformly random distributed particles, compared to the true $1/r^2$ forces computed with an Ewald summation in 256-bit precision. The other parameters are described in §4.2.4.

4.3.1 Spatial transients

As we know from our consideration of the eigenmodes of $\tilde{\mathscr{D}}$, every $\mathbf{k}$ has a three orthogonal eigenvectors $\hat{\mathbf{e}}_n$: one "longitudinal" and two "transverse" eigenvectors. The longitudinal eigenvector is most closely parallel to $\hat{\mathbf{k}}$ (thus we label it $\hat{\mathbf{e}}_\parallel$), and it converges to $\hat{\mathbf{k}}$ as $|\mathbf{k}| \to 0$. Below the Nyquist frequency, the longitudinal eigenmode also always has the largest eigenvalue, meaning it is the strongest growing mode on the grid. This has the following implication. Consider a mode $\tilde{\mathbf{u}}(\mathbf{k})$ oriented along $\hat{\mathbf{k}}$. Generically, this mode will have non-zero components along all three PLT eigenvectors. No matter the relative magnitudes of these components, the one with the largest eigenvalue will dominate after some time, because of the power-law behavior of Eq. 4.11. Until then, the excitation of the transverse eigenmodes can be seen as a transient that is purely dependent on the time since initialization. This is a discreteness effect that we can eliminate by initializing each mode in the longitudinal eigenmode $\hat{\mathbf{e}}_\parallel$ instead of $\hat{\mathbf{k}}$. We will call this mode $\tilde{\mathbf{u}}_\parallel(\mathbf{k})$, since $\tilde{\mathbf{u}}_\parallel(\mathbf{k}) \propto \hat{\mathbf{e}}_\parallel$.

What amplitude do we choose for $\tilde{\mathbf{u}}_\parallel(\mathbf{k})$? There are two reasonable choices:

$$|\tilde{\mathbf{u}}_\parallel(\mathbf{k})| = |\tilde{\mathbf{u}}(\mathbf{k})| \quad \text{or} \quad |\tilde{\mathbf{u}}_\parallel(\mathbf{k})| = \frac{|\tilde{\mathbf{u}}(\mathbf{k})|}{\hat{\mathbf{e}}_\parallel \cdot \hat{\mathbf{k}}}. \tag{4.16}$$

The former is simply a rotation of the old mode into the new direction, while the latter preserves the projection of the new mode onto $\hat{\mathbf{k}}$. We choose the latter gauge because it preserves the density power spectrum. In other words, the divergence remains unchanged, but we add a small curl component.

As always, we only excite modes below the Nyquist wavenumber, defined as $k_{\text{Nyquist}} = \pi/\Delta x$, where Δx is the particle spacing. This is the maximum wavenumber at which one can inject power without aliasing to lower wavenumbers. This corresponds to modes that

are sampled by at least two particles per cycle.

4.3.2 Temporal transients

The above prescription guarantees that the displacements start in the longitudinal eigenmode of the grid; now we must turn to the growing mode. This corresponds to choosing the initial velocities such that the decaying terms in Eqs. 4.11 cancel each other when substituted into Eq. 4.8. For an initial displacement field $\tilde{\mathbf{u}}_{\parallel}(\mathbf{k},t_0)$, the velocity field that cancels the decaying terms is

$$\tilde{\mathbf{v}}_{\parallel}(\mathbf{k},t_0) = \frac{\alpha^{-}(\mathbf{k})\tilde{\mathbf{u}}_{\parallel}(\mathbf{k},t_0)}{t_0}, \tag{4.17}$$

which, in combination with our choice above to use only $\hat{\mathbf{e}}_{\parallel}$, causes Eq. 4.8 to simplify to

$$\mathbf{u}(\mathbf{k},t) = \left(\frac{t}{t_0}\right)^{\alpha_{\parallel}^{-}(\mathbf{k})} \mathbf{u}(\mathbf{k},t_0). \tag{4.18}$$

In other words, the displacements evolve in the pure growing mode.

Note that this velocity choice is a significant departure from the Zel'dovich approximation (Zel'dovich 1970) in which the velocities are always parallel to the displacements. In our prescription, the parallel property still holds true in Fourier space, but the $\mathbf{k}$ dependence of α^{-} means it will not hold in configuration space.

We have now shown how to establish displacement and velocity fields $\mathbf{u}_{\parallel}(t_0)$ and $\mathbf{v}_{\parallel}(t_0)$ for an arbitrary input power spectrum that will eliminate transients to linear order. Two issues remain: the systematic under-/over-growth of modes on the grid, and non-linear transients. We first turn to the former.

4.3.3 Growth rates and rescaling

As we discussed in §4.2.2, every eigenvalue in Fig. 4.1 not equal to 1 will grow faster or slower than fluid theory predicts. The effects can be significant: the average undergrowth of a mode at $k_{\rm Nyquist}/2$ is about 15% after a factor of 10 increase in scale factor (see Fig. 4.2). To achieve 1% accuracy in the power spectrum, we would have to limit ourselves to wavenumbers below $\sim k_{\rm Nyquist}/10$ (if the problem were purely linear). Systematic small-scale undergrowth could also impact non-linear clustering, which we investigate in §4.5. Furthermore, the growth rates are dependent on the orientation relative to the grid, which could imprint preferred axes on the clustering.

Can we correct for this effect? Fluid linear theory gives us the expected displacement power spectrum as a function of time, and Eq. 4.8 gives the *actual* power spectrum that will be produced in a simulation. Thus, we can try rescaling the initial power by the ratio[6]

$$\begin{aligned} D_{\rm dens}(\mathbf{k},t) \equiv \frac{P^{\rm PLT}(\mathbf{k},t)}{P^{\rm fluid}(\mathbf{k},t)} &= \frac{|\tilde{\mathbf{u}}^{\rm PLT}(\mathbf{k},t)\cdot\hat{\mathbf{k}}|^2}{|\tilde{\mathbf{u}}^{\rm fluid}(\mathbf{k},t)\cdot\hat{\mathbf{k}}|^2} \\ &= \frac{\left|\left(\frac{t}{t_0}\right)^{\alpha_\parallel^-(\mathbf{k})} \frac{\tilde{u}^{\rm fluid}(\mathbf{k},t_0)\hat{\mathbf{e}}}{\hat{\mathbf{e}}\cdot\mathbf{k}}\cdot\hat{\mathbf{k}}\right|^2}{|\tilde{\mathbf{u}}^{\rm fluid}(\mathbf{k},t)\cdot\hat{\mathbf{k}}|^2} \\ &= \left(\frac{a(t)^{3\alpha_\parallel^-(\mathbf{k})/2}}{a(t)}\right)^2, \end{aligned} \tag{4.19}$$

where in the second line we have used Eqs. 4.16 & 4.18. This requires selecting a "target redshift" $z_{\rm target}$ at which time the simulation power spectrum and fluid power spectrum will match in linear theory. This redshift should be early enough that the displacements are still perturbative and PLT is still valid, but close enough to the onset of non-linear

[6]In practice, we are rescaling displacement amplitudes, not power, so we rescale by $\sqrt{D_{\rm dens}(\mathbf{k},t)}$.

evolution that the non-linearities will be seeded with the correct power spectrum. In our tests in §4.5, we try $z_{\rm target} = 12$ and 5. One still expects these "growth rate" effects to be present during non-linear evolution, but PLT loses descriptive power in that regime, so we can no longer apply rescaling.

One major concern (and indeed the concern that Joyce & Marcos (2007a) raise) with this "rescaling" is the accumulation of non-linearities while evolving from $z_{\rm init}$ to $z_{\rm target}$ due to the larger self-interaction of the displacements, since the displacement field is offset from the "true" fluid value during this time. In the strongly linear regime, this is a demonstrably negligible effect. To quantify this, we ran ABACUS in a high-precision mode (see Table 4.1) from $z_{\rm init} = 4999$ to $z_{\rm final} = 24$, a factor of 200 in scale factor, for 64^3 particles in a $50h^{-1}$Mpc box. Furthermore, we decreased σ_8 (the normalization of the power spectrum) by a factor of 1000 to decrease the displacement amplitudes, such that PLT should fully describe the evolution of the system. We tested two initial conditions: one with rescaling and one without (both started from the Zel'dovich approximation in the PLT growing mode) and compare both to the linear theory prediction at $z = 24$. The results are shown in Fig. 4.3, which demonstrates the remarkable success of rescaling. In configuration space, the particle displacements and velocities match linear theory to 0.006% on average[7], versus 6% without rescaling. In Fourier space, rescaling fully restores a 60% power deficit at $k_{\rm Nyquist}$. Besides being a strong confirmation of the correctness of PLT and rescaling within their regime of applicability, this is a remarkable testament to ABACUS's ability to evolve a system with displacements of order 10^{-6} of the interparticle

[7]Our definition of particle-averaged fractional error is the average absolute error over the mean magnitude, or $\langle|\mathbf{a}-\mathbf{b}|\rangle/\langle|\mathbf{a}+\mathbf{b}|/2\rangle$ We adopt this definition to avoid large numerical scatter due to the tiny magnitudes of some of the displacements.

spacing.

Of greater interest is rescaling in the weakly non-linear regime, where we intend to apply it in practice (for example, starting a simulation at $z_{\rm init} = 49$ with $z_{\rm target} = 5$). We anticipate that non-linearities may arise from two sources: fluid non-linearities that are present in the true physical problem and non-physical non-linearities due to the offset evolution of the rescaled field before $z_{\rm target}$. Thus, we must test whether the latter are sufficiently small. To that end, we run a simulation identical to the above, but with $z_{\rm init} = 49$ and $z_{\rm target} = 5$ and 2LPT initial conditions (see §4.4), and compare it to a simulation oversampled by a factor of 4. Specifically, we increase the particle count to 256^3 and truncate the power spectrum at $k_{\rm Nyquist}/4$. By only adding power below this wavenumber, we are oversampling the existing modes in the 64^3 box which thus reduces the requisite amount of rescaling on those modes. Thus, we expect non-linearities in the 256^3 simulation to represent fluid non-linearities, not rescaling non-linearities.[8]

The results of this test are shown in Fig. 4.4, where we have also shown the results of $z_{\rm target} = 12$. Rescaling to $z_{\rm target} = 5$ completely restores the lost power at all scales (up to 40% at $k_{\rm Nyquist}$). If there were substantial non-fluid non-linear effects, we would have expected excess power at $k_{\rm Nyquist}$ due to the earlier onset of non-linear growth, which is not present. This is not a purely linear regime, either: the non-linear contribution to the power is about 10% at $z = 5$ (dashed line). The scatter of $P(\mathbf{k})$ about the reference (reaching

[8]To make our comparisons meaningful, we compare the density fields as computed by the divergence of the displacements fields. In other words, we discard any curl components in the displacement fields. This is because the PLT growing mode introduces a small curl component, which is smaller for the 256^3 modes than the 64^3 modes, but it is ultimately a fixed property of the grid and does not change the density power spectrum.

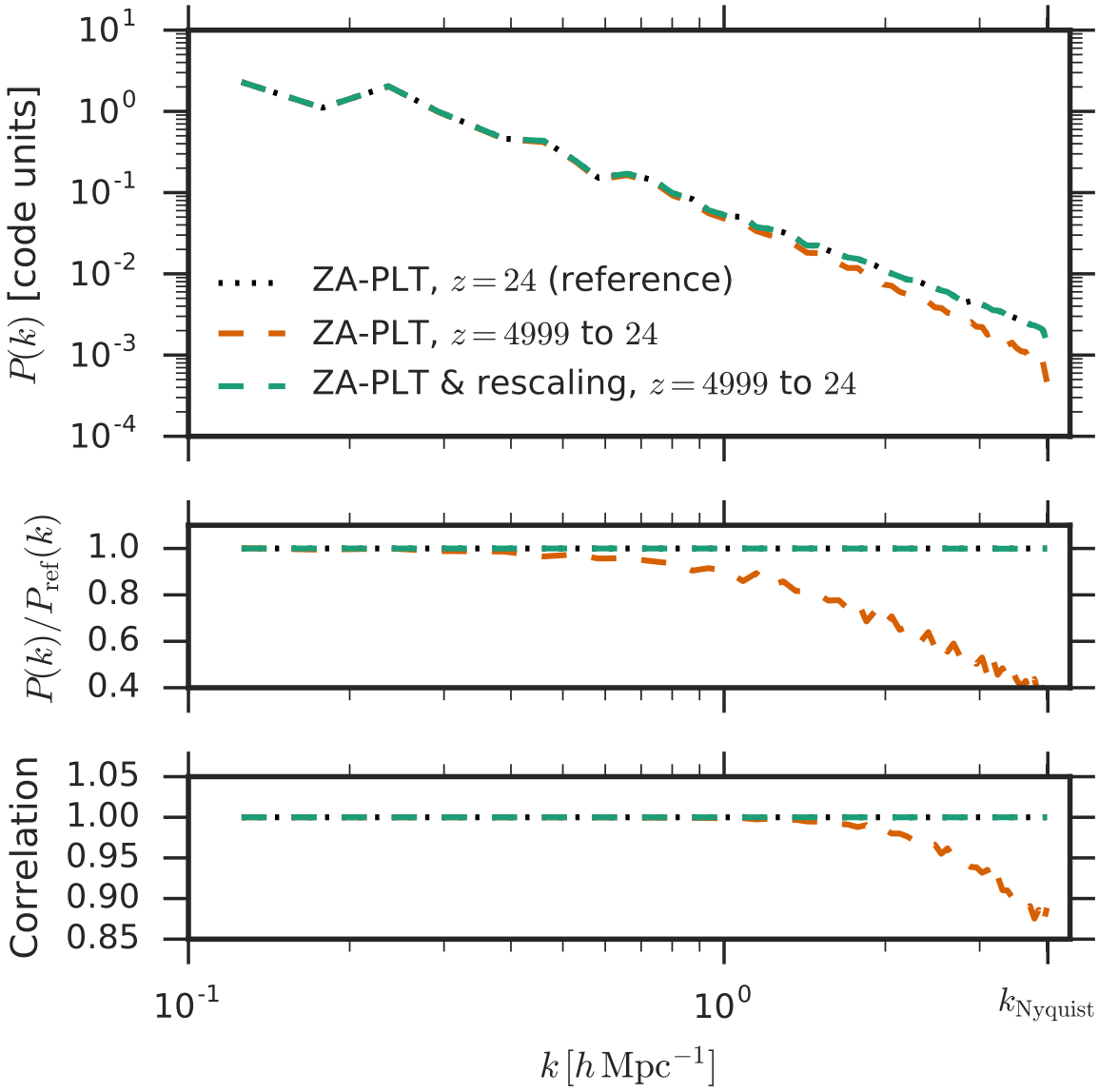

Figure 4.3: A test of rescaling in the strongly linear regime (evolving a simulation from $z = 4999$ to $z = 24$ and using $\sigma_8 = 0.8/1000$ at $z = 0$). *Top*: The density power spectrum for the reference (input) power spectrum at $z = 24$ (black dotted line), and the simulation power spectra at $z = 24$ with and without rescaling (dashed lines). The green dashed line is not the reference theory line – it is a simulation output – but it matches the reference to a factor of 10^{-5} in the displacements. This demonstrates that rescaling is capable of completely restoring the predicted fluid power spectrum in the strongly linear regime. *Middle*: The ratio of the simulation power spectra with the reference. *Bottom*: The cross-correlation of the phases of the density fields with the reference (see Fig. 4.5 caption for details).

15% at $k_{\rm Nyquist}$) may be evidence for these effects, but this is relatively unconcerning given that the uncorrected power spectrum has a 40% power deficit at the same scale. Thus, we consider $z_{\rm target} = 5$ a safe choice for use in cosmological simulations. Rescaling does not substantially change the cross correlation, which is already very good on all scales. This is consistent with our expectation from PLT that only the mode amplitudes are wrong relative to fluid linear theory; the phases remain unaffected.

We use 2LPT in this test because it is important for removing non-linear transients (as we show in the next section), but we choose a relatively high starting redshift of 49 to decrease its relative amplitude, since we want our results to be a measurement of rescaling, not 2LPT.

It should be emphasized that rescaling is only possible because of the eigenmode and growing mode corrections that we have already made. Otherwise, we would not know the growth rate for any mode and could not compensate for it, as each would be a mixture of three growth exponents (some negative!). In other words, the position corrections start the displacements in the longitudinal grid eigenmodes, the velocity corrections select the growing solution, and rescaling "divides out" the under-/overgrowth.

4.4 Second-order Lagrangian Perturbation Theory in Configuration Space

We present next a new technique for computing non-linear displacement and velocity corrections with second-order Lagrangian perturbation theory (2LPT). We emphasize that this scheme is derived from continuous theory – not PLT – so does not explicitly account

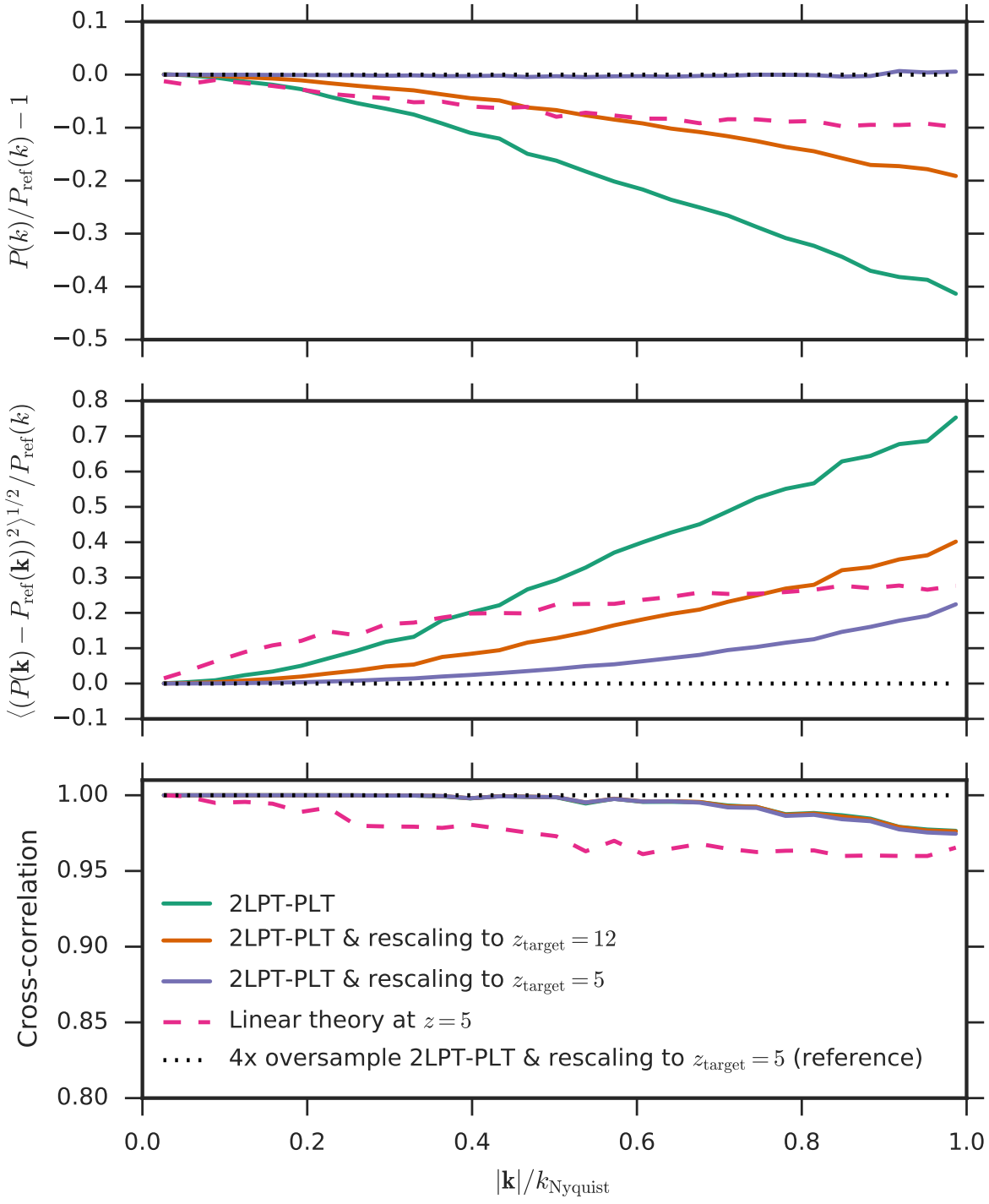

Figure 4.4: A test of rescaling of the initial conditions as we intend to use it in practice (see §4.3.3). The cosmology is ΛCDM with $k_{\rm Nyquist} = 4h$ Mpc^{-1} and a spline softening radius of 1/20 of the particle spacing. The reference is a 256^3 particle oversampled simulation (black dotted line), and the three test cases are 64^3 simulations with no rescaling, rescaling to $z_{\rm target} = 12$, and rescaling to $z_{\rm target} = 5$, respectively. All are initialized at $z = 49$ with 2LPT in the PLT growing mode. The dashed line is the linear theory prediction for $z = 5$, so the difference with the reference line quantifies the amplitude of non-linearities. *Top*: the ratio of each power spectrum with the reference. Rescaling completely restores

for grid effects. However, the configuration-space approach we employ naturally preserves the displacements in the PLT longitudinal eigenmodes. We numerically demonstrate the accuracy of our approach by comparing it with the actual evolution from very high redshift – an approach only possible with PLT corrections. We first present a derivation of our technique from continuous theory, then detail its implementation in ABACUS and tests of its accuracy.

4.4.1 Theory

As before, we take $\mathbf{x} = \mathbf{R} + \mathbf{u}$ to be the comoving position, where $\mathbf{R}$ is the initial grid and $\mathbf{u}$ is the comoving Lagrangian displacement. Taking $\mathbf{v}$ and $\mathbf{g}$ to be the comoving velocity and gravitational force, respectively, the equations of motion in an expanding universe are

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \tag{4.20}$$

$$\frac{d\mathbf{v}}{dt} + 2H\mathbf{v} = \mathbf{g} \tag{4.21}$$

$$\nabla_x \cdot \mathbf{g} = -4\pi G \rho_{\text{comoving}} a^{-3} \delta \tag{4.22}$$

In linear theory, $\mathbf{g} = (3/2)\Omega_m H^2 \mathbf{u}$. Substituting this and Eq. 4.21 into Eq. 4.22 yields the equation of motion

$$\frac{d^2\mathbf{u}}{dt^2} + 2H\frac{d\mathbf{u}}{dt} = \frac{3\Omega_m H^2}{2}\mathbf{u}. \tag{4.23}$$

For $\Omega_m = 1$, we have $a \propto t^{2/3}$ and $H = 2/3t$, which gives the usual growing-mode solution $\mathbf{u} \propto t^{2/3} \propto a$.

Beyond linear theory, the relation $\mathbf{u} \propto \mathbf{g}$ will break down. Thus, we can consider perturbative corrections to this field by writing

$$\mathbf{u}(t) = \varepsilon d_1(t)\mathbf{u}_1 + \varepsilon^2 d_2(t)\mathbf{u}_2 + \varepsilon^3 d_3(t)\mathbf{u}_3 + O(\varepsilon^4) \tag{4.24}$$

where ε is a bookkeeping notation that represents the order of the term in this perturbative expansion. The functions $d_n(t)$ and fields $\mathbf{u}_n$ are arbitrary at this point, but we will find them shortly by considering the dynamics.

If we take just the first term of the expansion ($\mathbf{u}(t) = \varepsilon d_1(t)\mathbf{u}_1$) but also consider the interaction of the field with itself, then we must have a force series of the form

$$\mathbf{g} = \frac{3\Omega_m H^2}{2}[\varepsilon d_1(t)\mathbf{u}_1 + \varepsilon^2 d_1^2(t)\mathbf{S}(\mathbf{u}_1) + O(\varepsilon^3)], \tag{4.25}$$

where $\mathbf{S}$ is some field that is second-order in $\mathbf{u}_1$.

Taking the next term in the expansion, we have $\mathbf{u}(t) = \varepsilon d_1(t)\mathbf{u}_1 + \varepsilon^2 d_2(t)\mathbf{u}_2$. We must now consider the self-interaction of the first-order part, the cross-interaction of the first- and second-order parts, and the self-interaction of the second-order part. This yields

$$\mathbf{g} = \frac{3\Omega_m H^2}{2}[\varepsilon d_1(t)\mathbf{u}_1 + \varepsilon^2 d_2(t)\mathbf{u}_2 + \varepsilon^2 d_1^2(t)\mathbf{S}(\mathbf{u}_1) + O(\varepsilon^3)], \tag{4.26}$$

where we have dropped any terms smaller than ε^2.

Substituting this into the equation of motion, we have

$$\left(\frac{d^2}{dt^2} + 2H\frac{d}{dt}\right)(\varepsilon d_1(t)\mathbf{u}_1 + \varepsilon^2 d_2(t)\mathbf{u}_2) = \frac{3\Omega_m H^2}{2}(\varepsilon d_1(t)\mathbf{u}_1 + \varepsilon^2 d_2(t)\mathbf{u}_2 + \varepsilon^2 d_1^2(t)\mathbf{S}(\mathbf{u}_1)). \tag{4.27}$$

Separating by order, we recover the linear growth equation

$$\left(\frac{d^2}{dt^2} + 2H\frac{d}{dt} - \frac{3\Omega_m H^2}{2}\right) d_1(t)\mathbf{u}_1 = 0, \tag{4.28}$$

which has the growing-mode solution $d_1(t) \propto t^{2/3}$. The next order is

$$\left(\frac{d^2}{dt^2} + 2H\frac{d}{dt} - \frac{3\Omega_m H^2}{2}\right) d_2(t)\mathbf{u}_2 = \frac{3\Omega_m H^2}{2} d_1^2(t)\mathbf{S}(\mathbf{u}_1). \tag{4.29}$$

Thus, we find that $\mathbf{S}(\mathbf{u}_1) = \mathbf{u}_2$. That is, as the linear displacement field grows, the first non-linear correction to the displacements is given by the part of the force due to interaction of the linear part with itself, up to an overall scaling. The time dependence is simply given by solving the above ODE for $d_2(t)$, which yields $d_2(t) = (3/7)d_1^2(t)$.

How do we find $\mathbf{S}(\mathbf{u}_1)$? If we write the force from $d_1(t)\mathbf{u}_1$ as $\mathbf{g}[d_1(t)\mathbf{u}_1]$, then Eq. 4.25 tells us

$$\begin{aligned} d_2(t)\mathbf{u}_2 &= \frac{3}{7}d_1^2(t)\mathbf{S}(\mathbf{u}_1) \\ &= \frac{3}{7}\frac{2}{3\Omega_m H^2}\frac{1}{2}(\mathbf{g}[d_1(t)\mathbf{u}_1] + \mathbf{g}[-d_1(t)\mathbf{u}_1]). \end{aligned} \tag{4.30}$$

Specifically, this is possible because $\mathbf{S}(\mathbf{u}_1)$ has even parity with respect to $\mathbf{u}_1$. Furthermore, note that the $O(\varepsilon^3)$ terms have canceled due to their odd parity. In summary, two force calculations with opposing first-order displacements isolates the second-order displacements to third-order accuracy.

We can calculate the velocities at each order simply from $\mathbf{v} = \dot{\mathbf{u}}$. For $\mathbf{u} = \sum d_j(t)\mathbf{u}_j$,

$$\frac{d(d_j)}{dt} = d_j\frac{1}{d_j}\frac{d(d_j)}{da}\frac{da}{dt} = d_j H f_j, \tag{4.31}$$

where f_j is the familiar $d\ln d_j/d\ln a$, which is just a property of the cosmology; for example, with $\Omega_m = 1$, we have $d_1 \propto a$ and $d_2 \propto a^2$, so $f_1 = 1$ and $f_2 = 2$. This yields velocities $\mathbf{v} = \sum H(t) f_j(t) d_j(t)\mathbf{u}_j$.

Having described our theory, we can now identify how it will interact with PLT and thus the particle grid. Since our first-order displacements will be in the longitudinal eigenmode, they can only produce second-order forces also in the longitudinal eigenmode, in analogy with fluid $\hat{\mathbf{k}}$ modes being unable to produce forces with a curl. Thus, the particle displacements after 2LPT will still be in the longitudinal eigenmode. However,

our fluid theory assumes that the second-order force is proportional to a^2 and that its direction is constant, both of which are not true in PLT. As a displacement grows in PLT, the wavevector-dependent linear growth factors will cause it to change direction in configuration space, causing the force direction to change as well. Thus, we attach the wrong displacement amplitudes and velocities. Of course, we would have to Fourier transform the displacements to correct these effects, which would negate much of the advantage of our configuration space approach. On large scales, we expect our corrections to be accurate, as the grid converges to fluid behavior. This is the behavior we find in §4.4.3.

4.4.2 Implementation

We implement our 2LPT as follows in Abacus. Normally, every timestep contains a velocity update ("Kick") and position update ("Drift") for every particle; in the following, we express our 2LPT approach as a set of Kick and Drift operators.

1. Generate a field of first-order displacements $d_1(t)\mathbf{u}$ using any standard technique [like the Zel'dovich approximation (ZA)], preferrably with PLT corrections. Apply the displacements to the particles.

2. Compute the force $\mathbf{g}[d_1(t)\mathbf{u}]$. Store in the velocity. This is like a Kick.

3. Reverse the displacement of every particle; that is, give every particle the position $\mathbf{x} = \mathbf{R} - d_1(t)\mathbf{u}$. This requires retrieving the initial grid location $\mathbf{R}$, which we store in each particle's ID number. This is like a Drift.

4. Compute the force $\mathbf{g}[-d_1(t)\mathbf{u}]$. Add to the velocity. This is like a Kick.

5. Take the position (currently holding the displacement $-d_1(t)\mathbf{u}$) and the velocity (currently holding $7H^2\Omega_m d_2(t)\mathbf{u}_2$) and manipulate to form the second-order position $\mathbf{R}+d_1(t)\mathbf{u}_1+d_2(t)\mathbf{u}_2$ and the second-order velocity $Hd_1(t)\mathbf{u}_1+2Hd_2(t)\mathbf{u}_2$. Store in the position and velocity. This is like a Drift.

The result is second-order Lagrangian perturbation theory for the cost of two force evaluations and memory requirements equal to the normal simulation code.

4.4.3 Accuracy

The purpose of 2LPT is to correct for evolution that is missed by starting at a low redshift instead of a very high redshift. Thus, to test the accuracy of our 2LPT theory and implementation, we run a simulation in a high-precision mode of ABACUS (see Table 4.1) with 512^3 particles in a $50h^{-1}$Mpc box from $z_{\rm init}=4999$ to $z_{\rm final}=24$. We set up the initial conditions using the ZA in the PLT growing mode, using rescaling as described above[9] with $z_{\rm target}=24$. We truncate the power spectrum at $k_{\rm Nyquist}/8$ to reduce the amplitude of the rescaling and grid effects in general to avoid non-linearities beyond those that we are seeking to measure here. Thus, our three test cases are produced on 64^3 particle grids, so they sample the same modes as the 512^3 reference. Fig. 4.5 shows the results of comparing the following fields:

(i) the Zel'dovich Approximation (labeled "No 2LPT");

(ii) 2LPTIC (Crocce et al. 2006), a standard Fourier-space 2LPT code ("2LPTIC");

[9] As a reminder, this means that, on a mode-by-mode basis, we increase/decrease the initial amplitudes to counteract the predicted linear under-/overgrowth at the target redshift.

(iii) our configuration-space 2LPT ("Abacus 2LPT"); and

(iv) the reference simulation evolved from $z = 4999$, as described above ("Full evolution").

Both Abacus 2LPT and 2LPTic do a very good job of reproducing the full evolution – much better than ZA in all metrics. Indeed, on large scales (larger than $\sim k_{\rm Nyquist}/3$), Abacus 2LPT and 2LPTic are nearly indistinguishable, and only show systematic differences of $< 0.5\%$ from the reference solution.

On smaller scales, the most discriminatory metrics are the root-mean-square scatter of $P(\mathbf{k})$ (Fig. 4.5, top right) and the transverse power $P_{\perp\hat{\mathbf{e}}}(k)$ (bottom left). The former is useful since k-averaged power $P(k)$ can hide anisotropic (e.g. lattice) effects, while the latter is useful since $P(k)$ is blind to the presence of curl modes. At $k_{\rm Nyquist}$, Abacus 2LPT has 4% scatter about the reference, while 2LPTic has 1.5%, since 2LPTic does not suffer from the same anisotropic particle discreteness effects as Abacus (see §4.4.1 for a discussion). At $k_{\rm Nyquist}/2$, these effects reduce to 1% and 0.5%, respectively.

In transverse power $P_{\perp\hat{\mathbf{e}}}(k)$, 2LPTic reaches 8% of its power in transverse modes at $k_{\rm Nyquist}$, versus 0% for Abacus 2LPT. Specifically, this transverse power is measured relative to the PLT eigenmodes, which are the proper eigenmodes that will not excite lattice transients. Assuming these transverse modes do not source forces, their power will decay as $1/a$. However, this is not a good assumption – the non-zero eigenvalues on the lower branches in Fig. 4.1 are direct evidence of that – which means that most of these modes will grow or oscillate indefinitely. This means that 92% of the continuum power at $k_{\rm Nyquist}$ is in the correct eigenmode, where it grows at a moderately suppressed rate, and 8% is in a transverse eigenmode, where it grows at a drastically suppressed rate. Furthermore, in all eigenmodes, the velocities (which are derived from continuum theory)

will mix growing and decaying solutions. Abacus's use of PLT eigenmodes eliminates all mixing of eigenmodes and decaying solutions.

2LPTic operates in Fourier space, while Abacus operates in configuration space, so comparing the two is useful and important test. Our detailed approach to generating comparable fields between Abacus and 2LPTic is the following:

1. Generate a ZA field and its corresponding 2LPT field in 2LPTic;
2. Project a copy of the ZA field onto the PLT grid modes;
3. Generate a 2LPT prediction with Abacus from the projected ZA field;
4. Compare to the 2LPTic 2LPT field.

We run 2LPTic with `Nmesh` $= 512$ and `Nsample` $= 64$ to generate both the 512^3 and 64^3 lattices.

Finally, we compare the 2LPT results from Abacus's high-precision configuration to those from Abacus's normal-precision configuration, since that is how it will be used in practice. The results match to remarkable precision: the particle-averaged fractional error is 7×10^{-6}, or close to floating-point precision, which is the floor, since Abacus's normal-accuracy calculations are in single precision.

4.4.4 Implementation caveats

For several reasons, this "displacement flipping" technique would not be well-suited to a normal N-body code with standard Zel'dovich approximation ICs. First, the accuracy of the 2LPT correction is a direct function of the force accuracy of the code being used. The

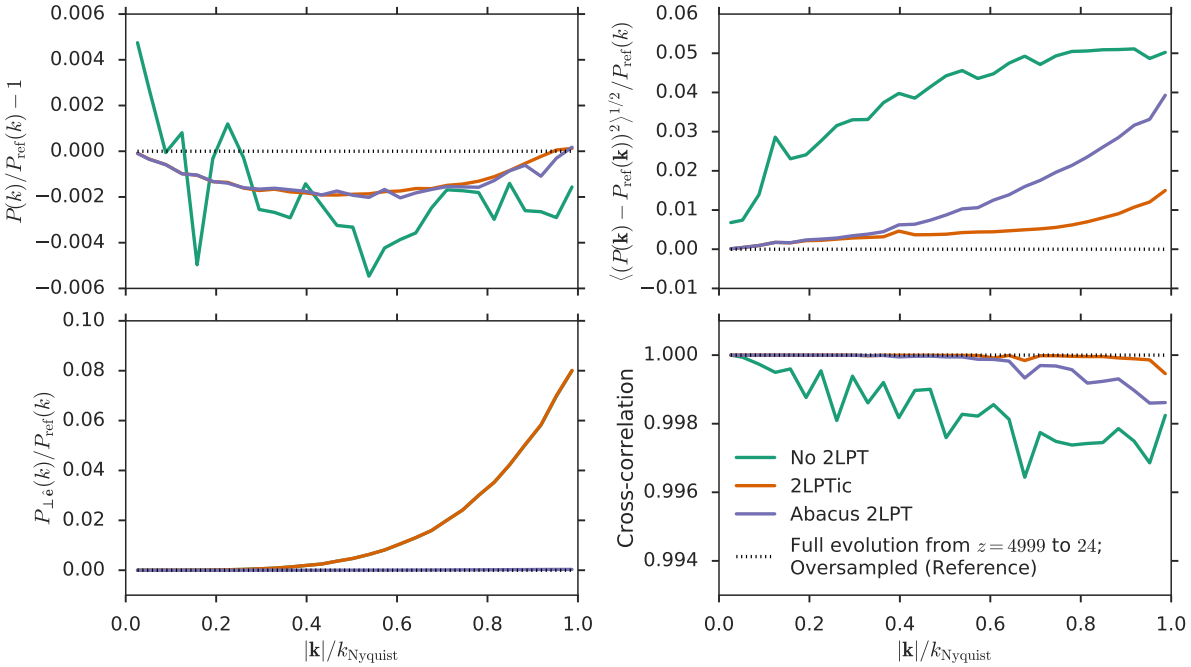

Figure 4.5: A comparison of our configuration-space 2LPT ("Abacus 2LPT") with a standard Fourier-space 2LPT code ("2LPTic") at $z = 24$ on a 64^3 particle grid. The reference solution is taken to be the full evolution from $z = 4999$ of an oversampled, rescaled simulation in the PLT growing mode with 512^3 particles as described in §4.4.3. Abacus 2LPT has larger RMS deviations in $P(\mathbf{k})$ versus 2LPTic (top right; 4% vs 1.5% at $k_{\rm Nyquist}$), but no power $P_\perp(k)$ in transient curl modes (0% versus 8% at $k_{\rm Nyquist}$; bottom left). The mean power $P(k)$ and phase cross-correlation are excellent at all k for both Abacus 2LPT and 2LPTic. *Top left*: The ratio of the power spectrum $P(k)$ of each of the density fields with the reference. We compute $P(k)$ from the density modes $\tilde{\delta}(\mathbf{k})$ as $P(k) = \left\langle \tilde{\delta}(\mathbf{k})\tilde{\delta}^*(\mathbf{k}) \right\rangle$, where * denotes complex conjugation and $\langle\cdot\rangle$ denotes averaging in annular bins of k. We compute $\tilde{\delta}(\mathbf{k})$ from the displacements $\tilde{\mathbf{u}}(\mathbf{k})$ in Fourier space as $\tilde{\delta}(\mathbf{k}) = \mathbf{k} \cdot \tilde{\mathbf{u}}$. *Top right*: The root-mean-square deviation of $P(\mathbf{k}) = \tilde{\delta}(\mathbf{k})\tilde{\delta}^*(\mathbf{k})$ in bins of k. *Bottom left*: Transverse (curl-mode) power, measured relative to the longitudinal PLT eigenvector $\hat{\mathbf{e}}_{\parallel}(\mathbf{k})$. The transverse power is computed as $P_{\perp\hat{\mathbf{e}}}(k) = \left\langle \tilde{\delta}_\perp(\mathbf{k})\tilde{\delta}_\perp^*(\mathbf{k}) \right\rangle$, where $\tilde{\delta}_\perp(\mathbf{k}) = |k\hat{\mathbf{e}} \times \tilde{\mathbf{u}}|$. *Bottom right*: Cross-correlation of the phases of the density fields, defined as $\mathrm{Re}\left(\tilde{\delta}^*_{\rm ref}(\mathbf{k})\tilde{\delta}(\mathbf{k})\right) / \left|\tilde{\delta}_{\rm ref}(\mathbf{k})\right| \left|\tilde{\delta}(\mathbf{k})\right|$.

highly symmetric configuration of the particles makes this a particularly difficult task, because the near- and far-field components of the force both have large amplitudes (but opposite signs). If the code does not respect the symmetry of the system, it is unlikely to produce accurate 2LPT corrections. ABACUS has exceptional force accuracy, even in this difficult configuration: in ABACUS's high-precision mode (see Table 4.1), a homogeneous lattice has a maximum absolute force error of 1×10^{-10} (mean 5×10^{-12}), compared to the mean amplitude of 2×10^{-3} for typical second-order forces at $z = 49$. In ABACUS's normal-precision configuration, the maximum noise is 2×10^{-5} (mean 1×10^{-6}). Thus, we might place an extremely conservative estimate of 1% 2LPT errors due to noise in the lattice; in practice, however, as we showed above, the normal precision 2LPT result matches the high-precision result to an average fractional error of 7×10^{-6}. Thus, we conclude that with ABACUS, even our normal precision results are more than adequate to produce 2LPT corrections.[10]

The second implementation challenge for most ICs is that the displacements should be in the longitudinal eigenmodes of the grid (see §4.3.1). Otherwise, the 2LPT corrections themselves will contain transverse modes. Since 2LPT corrects for missing evolution from high z, one should not expect a system in a transient configuration that cannot be reached from high z to be improved (at least on small scales) by this approach to 2LPT. The code developed by the authors to generate initial conditions from the Zel'dovich Approximation in the PLT growing mode is publicly available[11].

[10]Note that we quote force errors here on a homogeneous lattice, as opposed to the random particle configuration quoted in Table 4.1.

[11]`https://github.com/lgarrison/zeldovich-PLT`

Finally, we note that our recommended implementation overwrites the particle velocities. In theory, this is not a problem, because we can recompute the ZA velocity directly from the ZA displacement, as we prescribe in the last step of our implementation. However, in PLT, the velocity for the pure growing solution is not related to the displacement so simply (see §4.3.2). Thus, in practice, we must either save or re-read the original first-order velocities to restore them after our 2LPT computations.

4.5 Cosmological Results

We now test the impact of our modifications to the initial conditions (PLT, rescaling, and 2LPT) on common observables extracted from simulations. Specifically, we examine the density power spectrum, halo mass function, and halo two-point correlation function.

4.5.1 Simulation details

The base cosmology for our simulations is the Planck 2015 cosmology (Planck Collaboration 2016). All of our simulations are initialized at $z = 49$ and run to $z = 1$, with outputs at $z = 5$, 3, and 1. Our nominal simulation size is 720^3 particles in a $562.5\,h^{-1}$Mpc box, except for our "oversampled" simulation with 1440^3 particles in the same volume. The nominal particle mass is $4 \times 10^{10}\,h^{-1}\mathrm{M}_\odot$, and we use a Plummer softening length of 1/8 of the interparticle spacing, or 78 h^{-1} kpc. Our internal code parameters are those in the "normal precision" column of Table 4.1, with $\texttt{CPD} = 225$ (375) for 720^3 (1440^3) particles.

Each simulation was repeated 4 times with different realizations ("phases") of the input power spectrum. The results that follow are the average of the four, with error bars

representing the full range of variation across the phases (i.e. not the standard deviation).

The 1440^3 simulation serves as our point of reference in the results below. We truncate the input power spectrum at $k_{\rm Nyquist}/2$, so we are oversampling the existing modes in the 720^3 boxes by a factor of two. Thus, we consider it a more accurate representation of the "fluid truth" value, although it does not represent an absolutely converged reference point. We hold the softening fixed; i.e., the softening is 1/4 of the interparticle spacing in the 1440^3 simulation.

4.5.2 Power spectrum

We measure the density power spectrum at $z = 1$ both in projection and as a full 3D set of modes. In both cases, the density field is calculated with triangle-shaped cloud (TSC) mass assignment and deconvolved with the aliased-TSC window function from Jeong (2010, Chapter 7).

3D power spectrum

We compute the 3D density power spectrum with a fast Fourier transform (FFT) on a 720^3 mesh (Fig. 4.6). The combination of 2LPT and rescaling (2LPT-PLTR) (with either $z_{\rm target} = 5$ or 12) reproduces the power spectrum of the oversampled simulation to within 1% for nearly the whole range of modes down to $k_{\rm Nyquist}$. With just 2LPT, the accuracy falls below 1% at $k_{\rm Nyquist}/4$. In other words, to achieve 1% accuracy at $k_{\rm Nyquist}$, a simulation with only 2LPT would have to have 64 times the mass resolution as a simulation that also uses rescaling.

All of the results in Fig. 4.6 agree with our expectations. The Zel'dovich Approximation (ZA) misses substantial power (between 1% and 6%) at all but the largest scales, and adding our PLT eigenmode corrections (ZA-PLT) slightly worsens the $z = 1$ power. This is because the ZA-PLT initial velocities are smaller to match correctly the generically suppressed growth rate. Adding second-order Lagrangian Perturbation Theory (2LPT-PLT) is extremely helpful in recovering power on all scales, but 1 to 3% errors persist above $k_{\rm Nyquist}/4$, corresponding roughly to 64 particle haloes. Combining rescaling and ZA (ZA-PLTR) helps recover power on small scales, but not as well as 2LPT, and does very little on large scales. The combination of rescaling and 2LPT (2LPT-PLTR) produces the best match to the oversampled simulation, with sub-1% errors nearly down to $k_{\rm Nyquist}$, largely independent of our choice of $z_{\rm target}$.

Since a consideration of this work is the anisotropy of the simulation imposed by the axes of the particle lattice, we also produce a 3D power spectrum by rotating the simulation domain such that the particle lattice is skew to the FFT mesh and then measuring the 3D power spectrum. We also shrink the domain of the FFT by a factor of $\sqrt{3}$ to avoid gaps at the edges due to the rotation. The resulting power spectrum shows no substantial differences from Fig. 4.6; thus, we do not show it here.

Projected power spectrum

Weak lensing measures the projection of the matter power spectrum on the sky, so forecasts of weak lensing from simulations must use the 2D (projected) power spectrum, which we compute with a $(8 \times 720)^2$ FFT (Fig. 4.7). Before projection, we rotate the simulation domain so the particle lattice is skew to the FFT mesh (as described above). However,

this makes almost no difference in the resulting power spectrum. Note that we have plotted power above $k_{\rm Nyquist}$, where the particle lattice contributes significant power. This should serve as a cautionary example against considering evolved power in simulations above $k_{\rm Nyquist}$; however, we do expect that the non-linearities of structure formation will somewhat lessen the amplitude of these effects at lower z.

4.5.3 Halo mass function

We measure halo properties with three halo finders: ROCKSTAR (Behroozi et al. 2013), ROCKSTAR spherical overdensity (SO), and friends-of-friends (FoF). On large scales, the three are in very good agreement concerning the behavior of the nominal-resolution simulations compared to the oversampled reference simulation, but below ~ 500 particles we find qualitative differences. Understanding the behavior of different halo finders in this regime is relevant to interpreting our results – especially the impact of rescaling, which is fundamentally a small-scale correction.

The improper growth of modes near $k_{\rm Nyquist}$ has been identified as the dominant source of systematic errors in high-precision halo mass functions (Warren 2013). Previous attempts to correct the anisotropic aspect of this improper growth have resulted in dramatic suppression of small haloes (Reed et al. 2013) and, to our knowledge, no attempts have been made to compensate for the improper growth rates before this work.

We search for haloes at $z = 1$, 3, and 5, and restrict our analysis to haloes larger than 30 particles and halo mass bins with approximately 100 haloes or more.

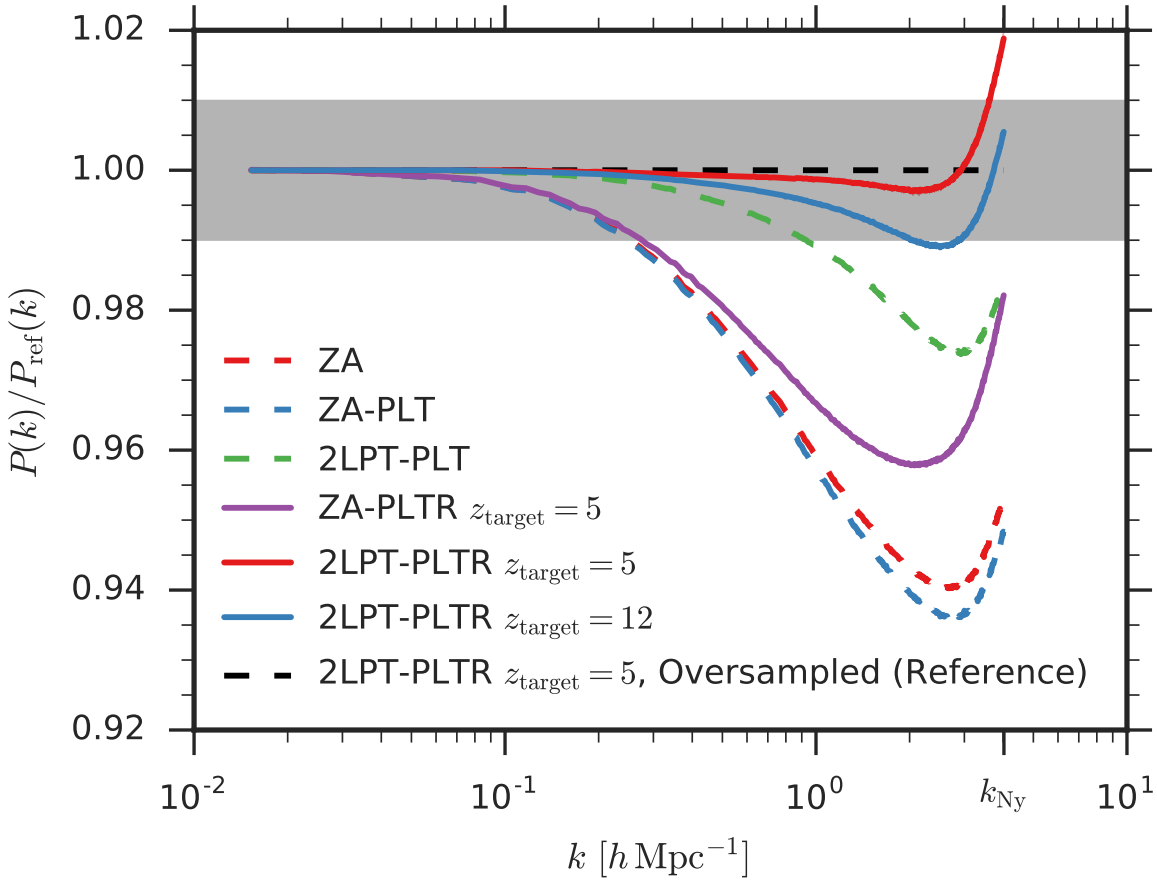

Figure 4.6: Density power spectra at $z = 1$ for different initial conditions. Each line is the average of 4 different simulations corresponding to different realizations of the input power spectrum. The red shaded region encloses total variability between the realizations for our preferred ICs (red solid line), but the variability is smaller than the line width. The grey shaded band indicates our target of 1% accuracy in the power spectrum. See §4.5.2.

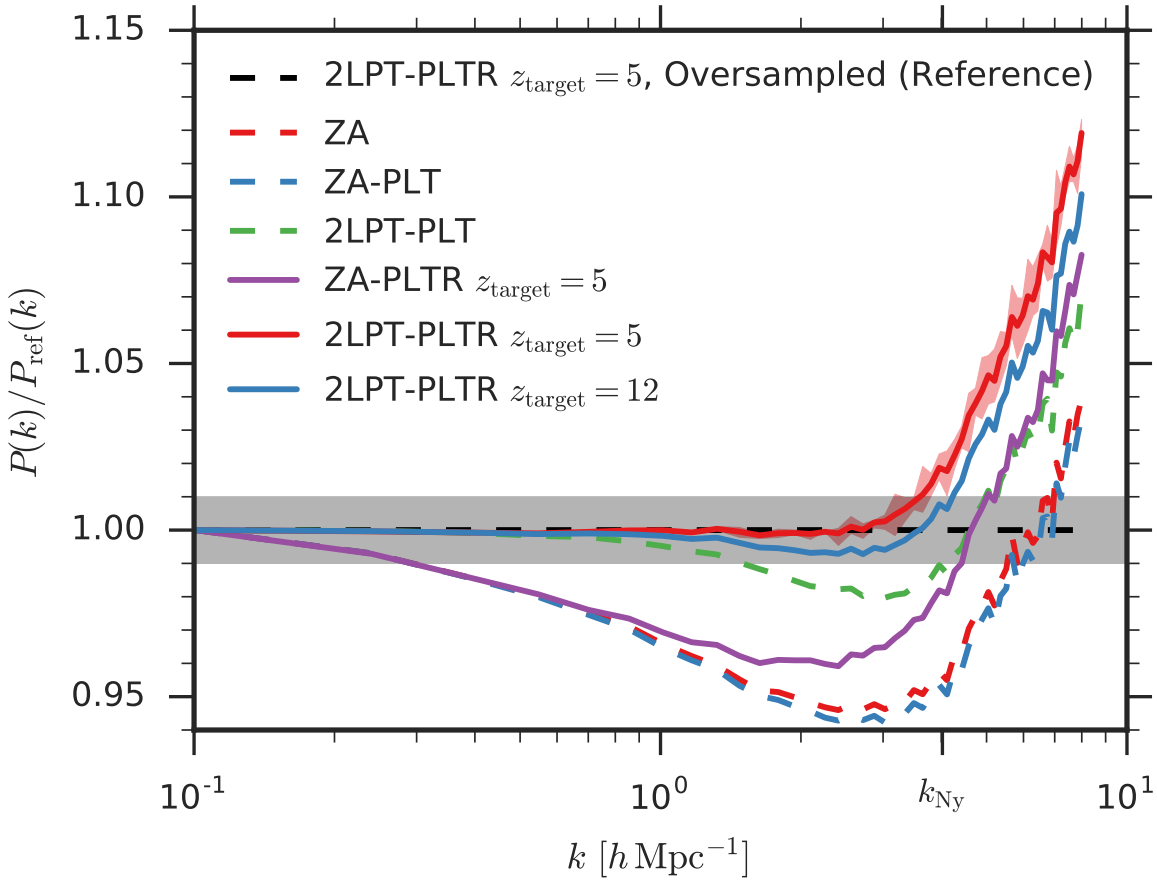

Figure 4.7: Projected power spectra at $z = 1$ for different initial conditions. The simulation domain is rotated to an angle skew to the FFT grid before projection. Each line is the average of 4 different simulations, each corresponding to a different realization of the input power spectrum. The red shaded region encloses total variability between the realizations for our preferred ICs (red solid line). The grey shaded band indicates our target of 1% accuracy in the power spectrum. See §4.5.2.

ROCKSTAR

ROCKSTAR is a hierarchical halo finder that recursively applies the friends-of-friends algorithm (see below) in six-dimensional phase space to identify dark matter structure and substructure. ROCKSTAR can also track haloes across time, but we restrict our analysis to strictly static simulation snapshots. We also only examine parent haloes (whose masses include all substructure). A parent halo is a halo whose center does not lie within the the radius of a larger halo. We use the default "virial density" threshold to define haloes.

The ROCKSTAR halo mass functions are shown in Fig. 4.8 and summarized in Table 4.2. The same trends that were visible in the $z = 1$ power spectrum (§4.5.2) are present in the halo mass function. These effects are perhaps most clearly elucidated at $z = 3$, where we find a 20 to 25% deficit of haloes across the whole mass range of 30 – 500 particles when using the Zel'dovich Approximation (ZA), compared to the oversampled reference simulation (corresponding to the mass range of 240 – 4000 particles). Adding rescaling to the initial conditions restores half of the missing small haloes, but does very little for the large haloes, as we would expect. 2LPT is the most important factor for recovering large haloes, and also has an appreciable impact on small haloes. The combination of 2LPT and rescaling successfully recovers the halo mass function of the oversampled simulation to within 5% across the whole mass range at this redshift.

At $z = 1$, our preferred combination of 2LPT and rescaling successfully restores the 5 to 10% deficit of haloes seen with ZA across the whole mass range of 30 – 5000 particles. However, below about 500 particles, we also overproduce haloes by 1 to 6%. To test the origin of this surplus, we downsample the 1440^3 reference simulation by a factor of 8 and then run ROCKSTAR on the resulting 720^3 particles. Specifically, we downsample

by a factor of 2 on every dimension of the original particle lattice, such that we select the particles whose initial lattice sites matches those in the 720^3 simulations. The result of this procedure is labeled "Downsampled" in Fig. 4.8, where we see that downsampling tends to overproduce small haloes by 3 to 10%. Thus, it appears that the over-production of small haloes in our preferred ICs (rescaling and 2LPT) may be an artefact of halo finding in a coarsely sampled simulation, rather than a physical feature of the simulation itself.

Because our downsampling procedure is likely to produce non-physical variations in the binding energy of haloes, we disable halo unbinding in all of our ROCKSTAR analyses. This corresponds to setting `UNBOUND_THRESHOLD` to 0 and disabling `BOUND_PROPS`.

ROCKSTAR SO

ROCKSTAR also has the option to output spherical overdensity masses. The correspondence between ROCKSTAR haloes and SO haloes is one-to-one – SO simply uses the halo centers that ROCKSTAR has already identified. The SO mass is computed by expanding a spherical search volume from the halo center until the average density within the sphere falls below the threshold density. SO masses are interesting for our analysis, because, naively, we would expect this technique to be relatively less sensitive to the difference in the spatial sampling between the nominal-resolution and oversampled simulations. In particular, FoF-based techniques tend to link together haloes along filamentary structures, which SO will not do (Knebe et al. 2013).

The ROCKSTAR SO mass functions are shown in Fig. 4.9 and summarized in Table 4.3. The same trends are visible in SO masses that were visible in ROCKSTAR masses.

However, while we do not overproduce small haloes at $z = 1$, we do slightly underproduce large haloes at $z = 1$. This is evidence of a systematic mass shift across the whole mass range. Specifically, on a halo-by-halo basis, switching from ROCKSTAR to ROCKSTAR SO inflates the masses of the haloes in the oversampled simulation more than the haloes in the nominal-resolution simulations.

Friends-of-friends

The friends-of-friends (FoF) algorithm links particles together if they are within a "linking length" b. A set of linked particles is identified as a halo. The linking length is commonly expressed as a fraction the interparticle spacing $\Delta x = N^{1/3}$. We use $b = 0.2$ in this analysis.

The FoF mass functions are shown in Fig. 4.10 and summarized in Table 4.4. FoF is very sensitive to the spatial stochasticity of the particle sampling, due to the Poisson nature of particle linking and the above-mentioned filamentary linking problem. Thus, we consider the downsampled simulation (where we take 1/8 of the oversampled particles) as our reference case. Without this, we would conclude that our simulations overproduce haloes below 1000 particles by 5 to 15% at $z = 1$. We consider the above ROCKSTAR results as further evidence that the downsampled simulation is indeed the appropriate reference case.

Our preferred ICs using rescaling and 2LPT match the downsampled mass function to within 1% across nearly the whole mass range of 30 – 2000 particles at $z = 1$, with similar success at the other redshifts. At high masses, the small number of haloes causes fluctuations at the 5% level, but there is no evidence for additional systemic shifts.

The particularly simple nature of FoF makes it unsurprising that we recover very

similar results to the power spectrum analysis, namely, that our preferred ICs are an excellent match to the oversampled simulation, and that 2LPT alone is insufficient below 500 particles, with 5% too few haloes of 100 particles.

4.5.4 Halo clustering

We compute the two-point correlation function (2PCF) of haloes as

$$\xi(s) = \frac{DD}{RR} - 1, \tag{4.32}$$

which is equivalent to the Landy-Szalay estimator (Landy & Szalay 1993) because the domain is a cubic, periodic box. We restrict our analysis to haloes separated by 3 – 30 h^{-1}Mpc at $z = 1$, where we have a sufficient number of halo pairs and are not approaching the box periodicity scale. We compute the DD term using TreeCorr (Jarvis 2015) and the RR term analytically. We repeat this analysis for the haloes found by each halo finder.

We compute the correlation function in four mass bins. The first two are simple halo mass cuts: 100 – 300 particles, and 300 – 1000 particles. The next two are halo-mass rank cuts: all haloes ranked between 10^5 and 3×10^4, and all haloes ranked above 3×10^4. These ranks are chosen such that the average mass limits across phases and ICs approximately correspond to the absolute mass cuts of the first two bins. This halo-mass ranking procedure is akin to "abundance matching", wherein dark matter haloes are matched to their observational counterparts by matching the mass rank of each, rather than attempting to make absolute mass calibrations (e.g. Guo et al. 2010). This is particularly important for volume-limited galaxy surveys, which have no direct knowledge of the masses of haloes but do know the number densities of the most massive haloes. The following results suggest that clustering derived from abundance matching has a relatively smaller systematic

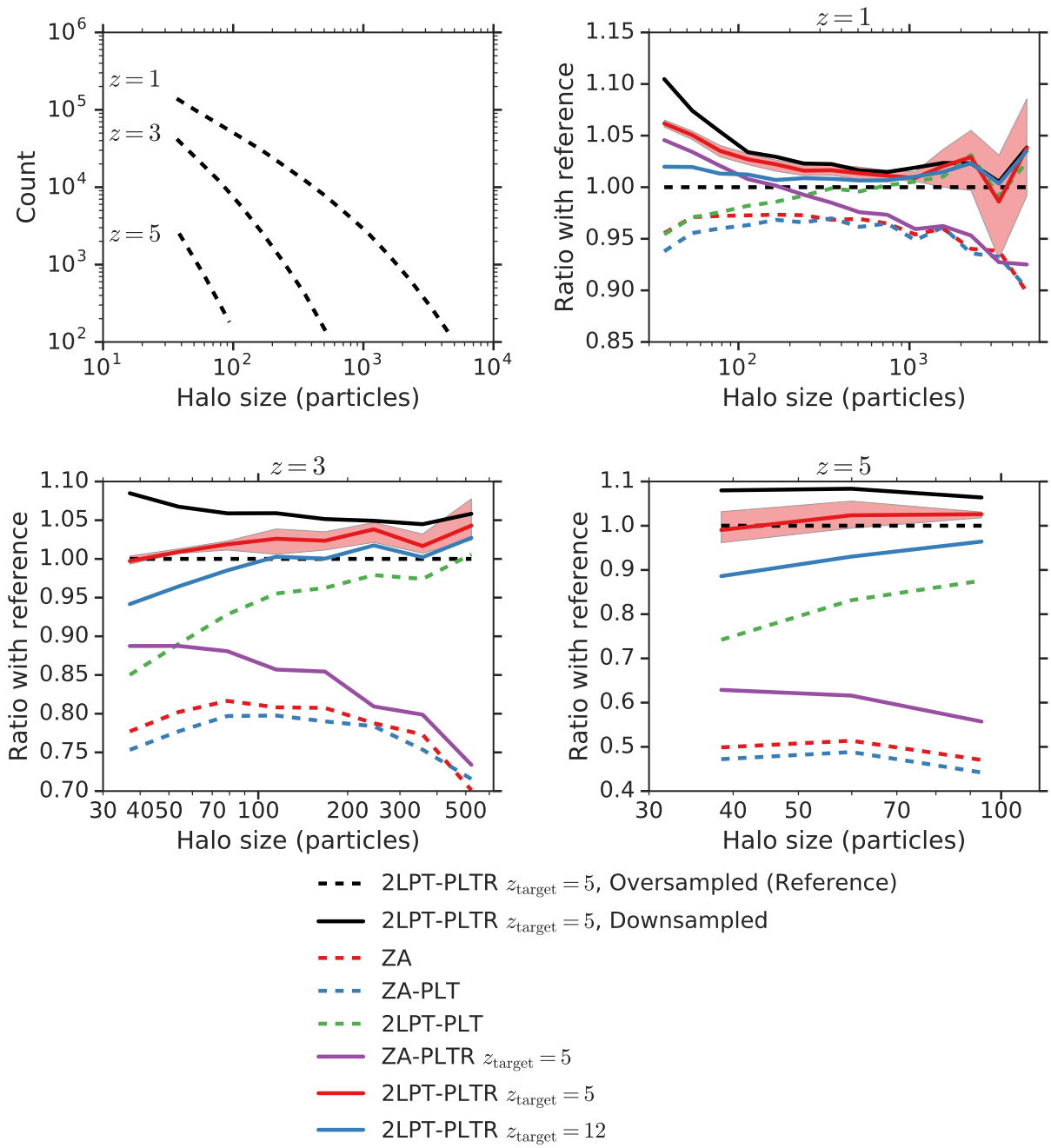

Figure 4.8: *Top left*: ROCKSTAR halo mass functions at the three output redshifts in the reference (oversampled) simulation. Each line corresponds to one of the three other panels. Halo particle counts have been divided by 8, to show them on the same mass scale as our other simulations. *Top right, bottom left, bottom right*: The halo mass functions at the three output redshifts divided by the reference mass function at that redshift. These results are summarized in Table 4.2.

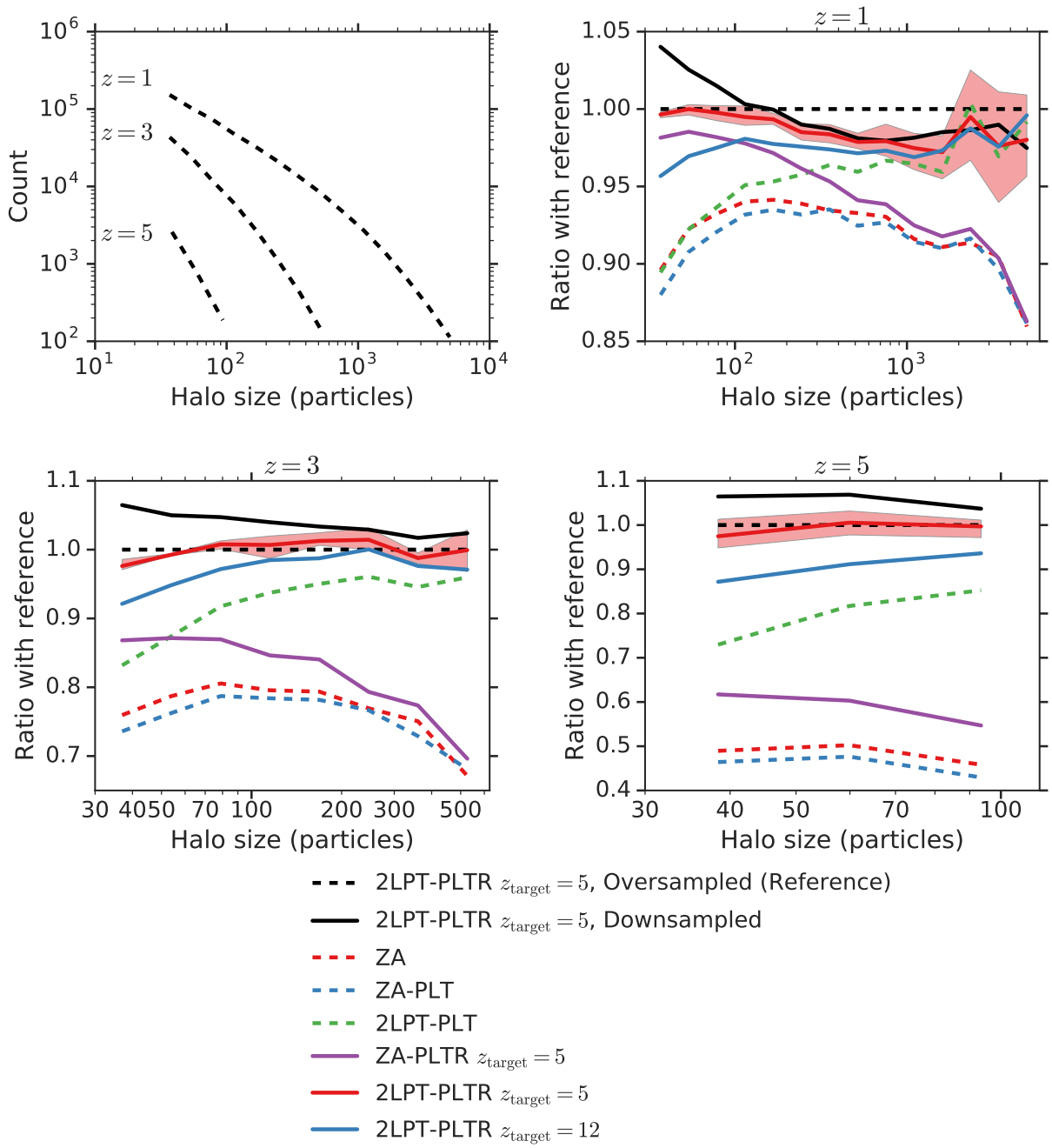

Figure 4.9: Same as Fig. 4.8, but for ROCKSTAR spherical overdensity halo masses. These results are summarized in Table 4.3.

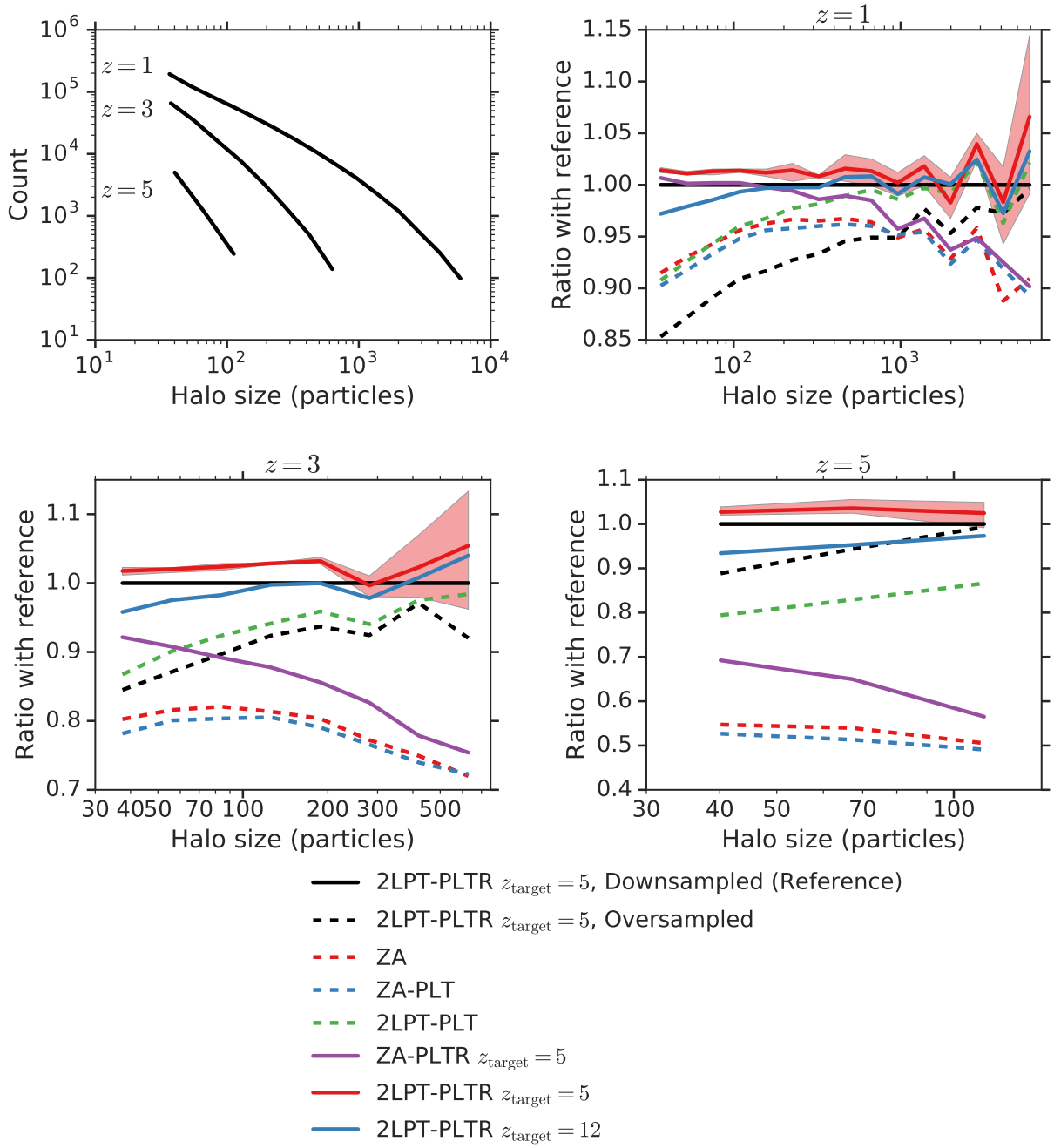

Figure 4.10: Same as Fig. 4.8, but for friends-of-friends halo masses. Note that the reference line is the "downsampled" result, because the "oversampled" result is a very poor match to the nominal-resolution results. The downsampled result is produced by taking a subsample of one out of eight particles and running that subsample through FoF. This matches the spatial stochasticity of the nominal-resolution simulations and produces results that agree with the ROCKSTAR and power spectra results. These results are summarized in Table 4.4.

Table 4.2:: Mean errors in the ROCKSTAR halo mass functions (Fig. 4.8)

Simulation	$z = 1$	$z = 3$	$z = 5$
2LPT-PLTR $z_{\rm target} = 5$, Oversampled (Ref.)	$0.0\% \pm 0.0\%$	$0.0 \pm 0.0\%$	$0.0 \pm 0.0\%$
2LPT-PLTR $z_{\rm target} = 5$, Downsampled	3.4 ± 4.3	5.9 ± 6.0	7.6 ± 7.6
ZA	-4.2 ± 4.7	-21.6 ± 21.9	-50.6 ± 50.6
ZA-PLT	-4.8 ± 5.1	-22.9 ± 23.1	-53.3 ± 53.3
2LPT-PLT	-0.6 ± 2.1	-5.7 ± 7.4	-18.3 ± 19.2
ZA-PLTR $z_{\rm target} = 5$	-1.7 ± 3.9	-16.1 ± 16.9	-39.9 ± 40.1
2LPT-PLTR $z_{\rm target} = 5$	2.4 ± 3.0	2.2 ± 2.6	1.3 ± 2.1
2LPT-PLTR $z_{\rm target} = 12$	1.3 ± 1.6	-0.7 ± 2.7	-7.3 ± 8.0

NOTES – Summary of the offset and scatter of the halo mass functions for different ICs relative to the reference ICs. The mean deviation for a given simulation is the average fractional difference of that simulation's halo mass function from the reference. The quoted uncertainty is the root-mean-square of this fractional difference.

Table 4.3:: Mean errors in the ROCKSTAR SO halo mass functions (Fig. 4.9)

Simulation	$z = 1$	$z = 3$	$z = 5$
2LPT-PLTR $z_{\text{target}} = 5$, Oversampled (Ref.)	$0.0\% \pm 0.0\%$	$0.0\% \pm 0.0\%$	$0.0\% \pm 0.0\%$
2LPT-PLTR $z_{\text{target}} = 5$, Downsampled	-0.4 ± 1.9	3.8 ± 4.1	5.7 ± 5.8
ZA	-8.0 ± 8.3	-23.3 ± 23.7	-51.7 ± 51.7
ZA-PLT	-8.6 ± 8.9	-24.7 ± 24.9	-54.3 ± 54.4
2LPT-PLT	-4.3 ± 5.0	-7.8 ± 8.9	-20.0 ± 20.7
ZA-PLTR $z_{\text{target}} = 5$	-5.5 ± 6.5	-18.0 ± 18.9	-41.1 ± 41.2
2LPT-PLTR $z_{\text{target}} = 5$	-1.4 ± 1.7	-0.0 ± 1.2	-0.8 ± 1.5
2LPT-PLTR $z_{\text{target}} = 12$	-2.5 ± 2.6	-3.0 ± 3.8	-9.4 ± 9.7

NOTES – See Table 4.2 Notes.

Table 4.4:: Mean errors in the friends-of-friends halo mass functions (Fig. 4.10)

Simulation	$z = 1$	$z = 3$	$z = 5$
2LPT-PLTR $z_{\text{target}} = 5$, Downsampled (Ref.)	$0.0\% \pm 0.0\%$	$0.0\% \pm 0.0\%$	$0.0\% \pm 0.0\%$
2LPT-PLTR $z_{\text{target}} = 5$, Oversampled	-6.5 ± 7.6	-8.9 ± 9.6	-5.8 ± 7.2
ZA	-5.6 ± 6.1	-21.3 ± 21.5	-46.9 ± 47.0
ZA-PLT	-6.1 ± 6.5	-22.4 ± 22.6	-49.0 ± 49.0
2LPT-PLT	-2.5 ± 3.9	-6.3 ± 7.3	-17.0 ± 17.3
ZA-PLTR $z_{\text{target}} = 5$	-2.7 ± 4.1	-14.8 ± 15.9	-36.4 ± 36.8
2LPT-PLTR $z_{\text{target}} = 5$	1.4 ± 2.4	2.5 ± 2.9	2.9 ± 3.0
2LPT-PLTR $z_{\text{target}} = 12$	-0.2 ± 1.7	-0.8 ± 2.5	-4.7 ± 4.9

NOTES – See Table 4.2 Notes.

error due to details of the initial conditions than clustering from a mass-selected set of haloes.

ROCKSTAR

The 2PCF of ROCKSTAR haloes is shown in Fig. 4.11. The absolute-mass bins show large sensitivity ($\sim 5\%$) to the ICs, because the masses of haloes at a constant bias are systematically shifting up or down. Thus, the mass bins are gaining or losing some bias relative to the reference simulation, shifting the 2PCF.

Switching to mass-ranked bins reduces the scatter among the ICs by a factor of 2 or more. Regardless of absolute-mass or mass-ranked binning, either 2LPT or 2LPT with rescaling is the best match to the reference. With mass ranking, we can recover the 2PCF to within a fraction of a percent, especially after averaging over phases. Adding volume to our boxes would produce the same effect, which is one reason why multi-Gpc boxes will be required to calibrate upcoming galaxy surveys.

The most important factor for recovering the 2PCF is 2LPT, followed by rescaling. Without 2LPT, systematic shifts of 1 to 3% are seen in all absolute-mass bins, or 1% in the mass-ranked analyses.

ROCKSTAR **SO**

The 2PCF of ROCKSTAR SO haloes is shown in Fig. 4.12. We see the same reduction in scatter moving from absolute-mass to mass-ranked bins, but we see a large (3%) offset from the reference solution that was not present before. Since the halo centers are identical to the ROCKSTAR halo centers, the SO haloes must be reordering the masses of haloes,

such that the 3×10^4 most massive haloes (for example) are a relatively less-biased sample than the reference. We note that the downsampled simulation is an excellent match to the nominal-resolution simulations, as it is in the mass functions, suggesting that mass-reordering due to the sensitivity of ROCKSTAR to the mass resolution of the simulations is the main culprit.

Friends-of-friends

The 2PCF of FoF haloes is shown in Fig. 4.13. The FoF results are quite similar to the ROCKSTAR results (with 2LPT and rescaling offering the best match to the reference, followed by 2LPT alone), with the exception that our preferred solution works equally well in the absolute-mass and mass-ranked bins. This is due to the excellent match in the mass function between the reference and the 2LPT results, so very little halo reordering must occur when switching to the mass-ranked functions. Our preferred solution recovers the 2PCF to within a fraction of a percent in most cases. The other ICs are shifted by 2 to 4% in the absolute-mass bins, or 1 to 2% in the mass-ranked.

4.5.5 Glass initial conditions

All of the simulations in this work have used particle-lattice pre-initial conditions, because less structured configurations (such as a "glass") are harder to treat in the PLT framework. Analytically, the eigenmodes are no longer plane waves, and computationally, the N 3×3 eigenvalue problems become a $3N \times 3N$ problem. Our approach here is to generate glass initial conditions and empirically demonstrate that they do not alleviate the systematic suppression of small-scale power that is predicted by PLT. This reproduces the result of

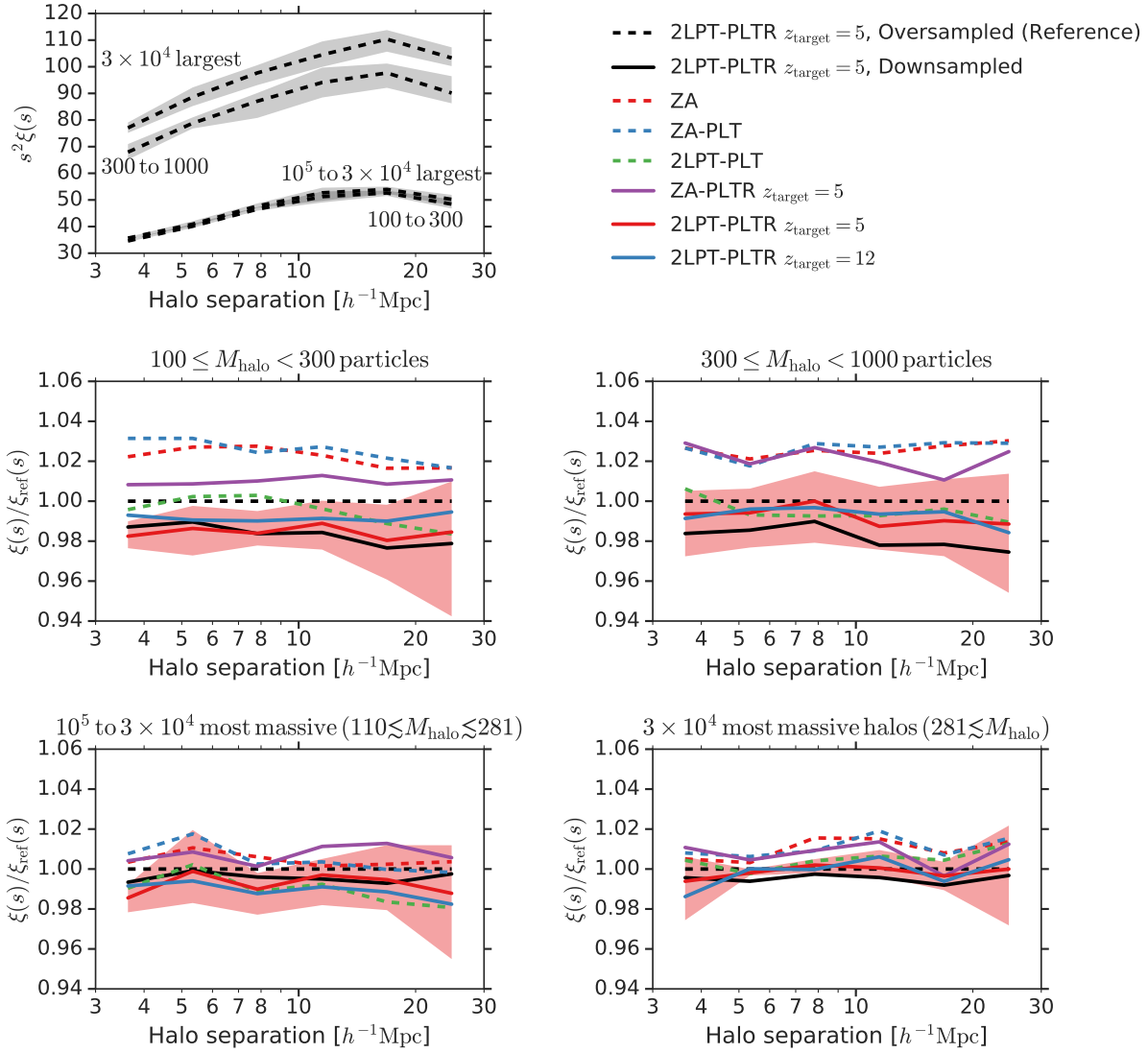

Figure 4.11: The two-point correlation function of ROCKSTAR haloes at $z = 1$. Each of the bottom four panels shows a different mass bin. The middle row shows mass-selected haloes, while the bottom row shows mass-rank-selected. The approximate mass ranges are the average across all of the simulations.

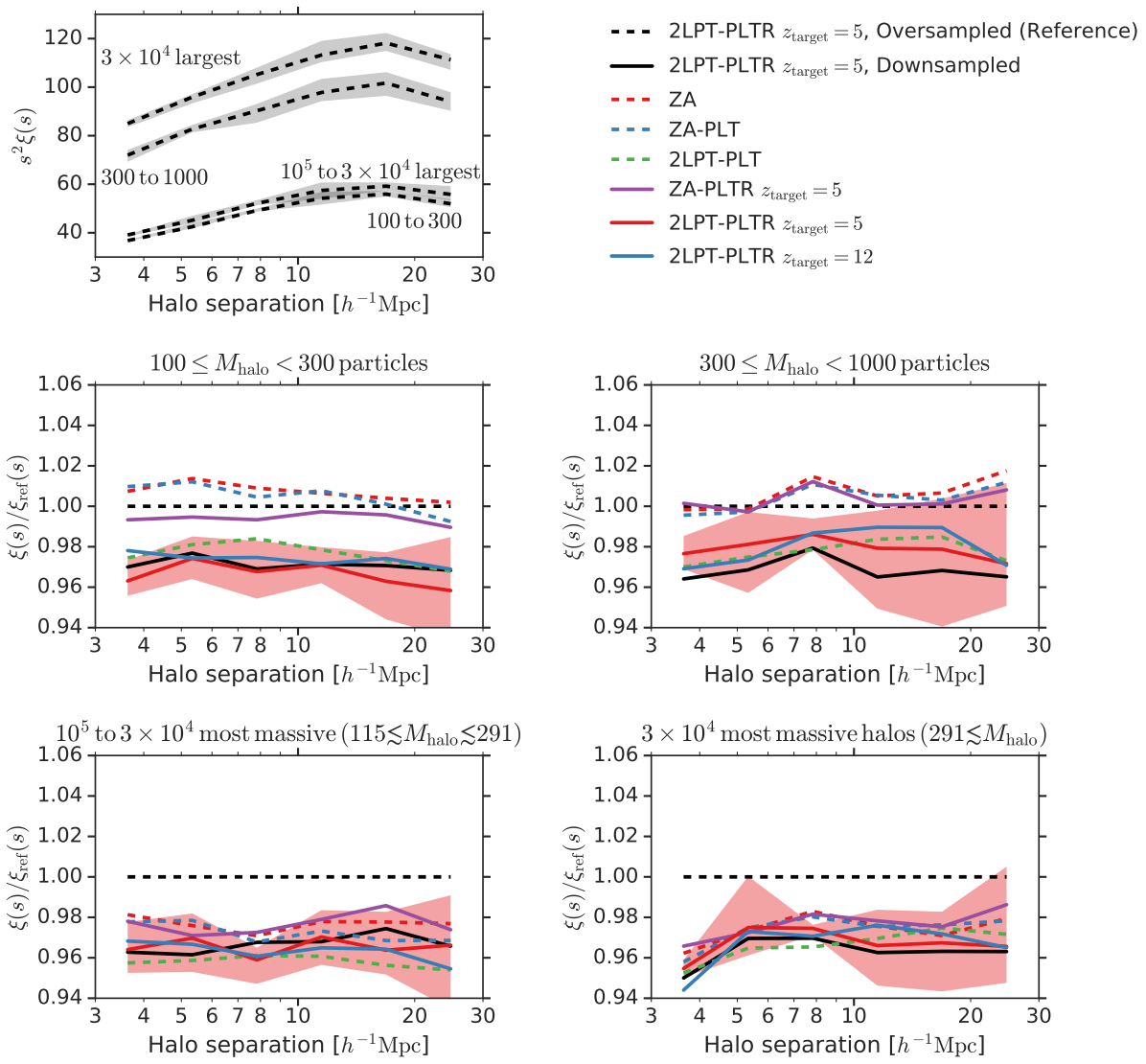

Figure 4.12: Same as Fig. 4.11, except with ROCKSTAR spherical overdensity halo masses.

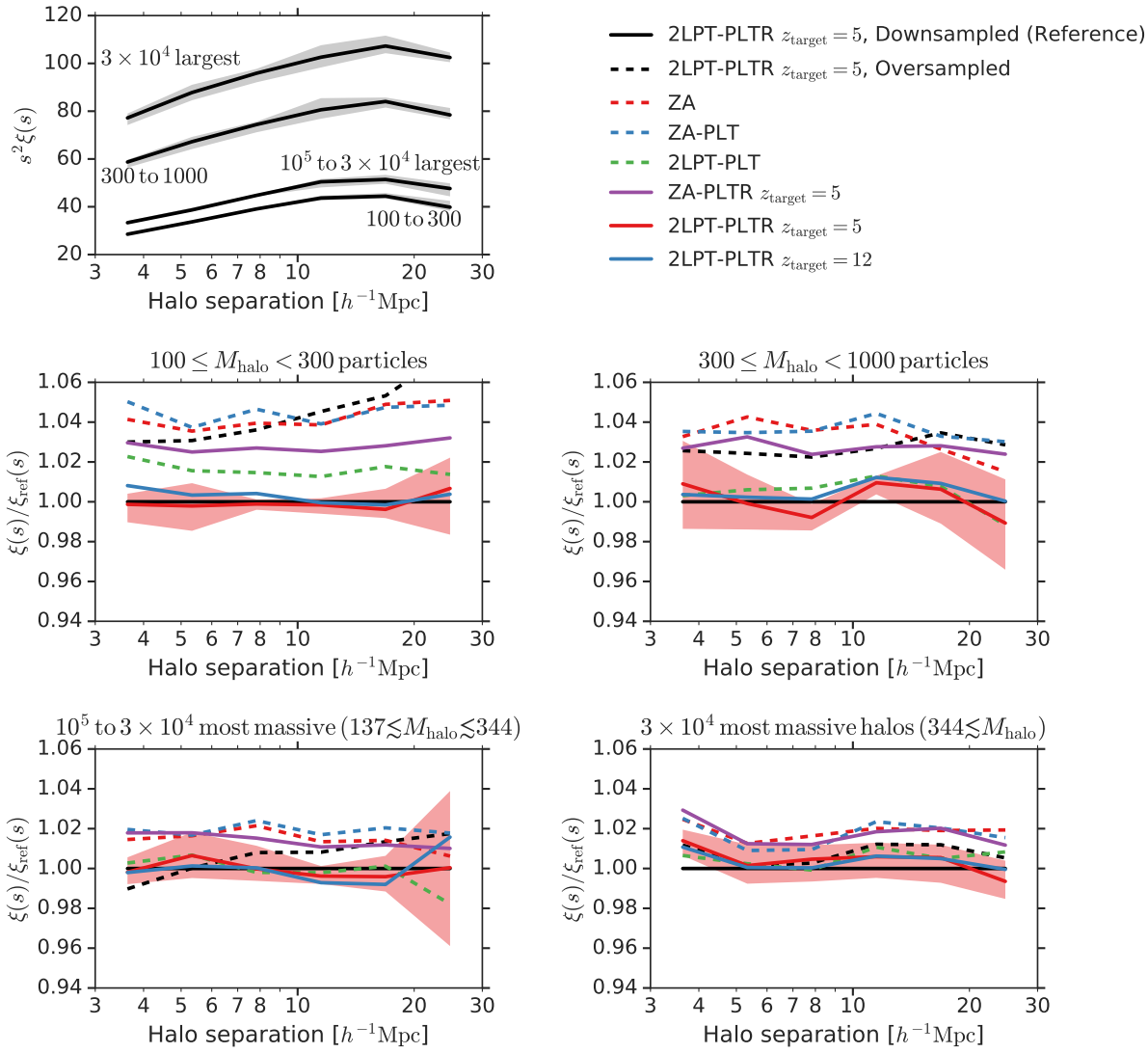

Figure 4.13: Same as Fig. 4.11, except with friends-of-friends halo masses.

Joyce et al. (2009).

A glass is a force-free configuration of particles that is reached by evolving a random distribution of particles in an expanding background with the sign of gravitational acceleration reversed. All the particles repel each other and oscillate about their equilibrium positions until they have been sufficiently damped by the background expansion. Like a lattice, a glass is uniform on large scales, but unlike a lattice, it is also isotropic on scales appreciably larger than the particle spacing.

We expect that glass pre-initial conditions will eliminate large-scale anisotropy produced by the particle lattice, but not the systematic small-scale suppression of power. This effect depends only on the fact that the continuum density field has been discretized, not the nature of that discretization. This is what Joyce & Marcos (2007a) call *dynamical sparse sampling effects*.

To test this, we use ABACUS to generate 72^3 and 80^3 particle glasses, which are tiled using 2LPTIC (Crocce et al. 2006) to produce 360^3 and 720^3 2LPT initial conditions, respectively. We use 2LPTIC's cloud-in-cell deconvolution to reduce the suppression power from interpolating to the glass. The number of ABACUS cells-per-glass-tile is kept constant between the glass generation and actual simulations, as are the precision and softening. This ensures no discontinuities in the residual forces as we transition from glass-making to simulation. The mean residual forces after glass generation are a factor of 500 below the mean forces on the particles at $z_{\rm init} = 49$.

Fig. 4.14 shows that switching from lattice to glass pre-ICs does not restore power. The 360^3 glass configuration at $z = 1$ shows just as much loss of power relative to the 720^3 glass as the 360^3 lattice does from the 720^3 lattice.

The 720^3 glass shows a small loss of power (0.5% at $k_{\rm Nyquist}$) relative to the 720^3 lattice, which we attribute to loss of power during interpolation of the displacements from the FFT mesh to the particles (despite our use of CIC deconvolution[12]). If the effect were due to residual forces in the glass tiles, we would have expected accelerated growth of structure and a surplus of power, not a deficit. Indeed, increasing the size of the FFT mesh in 2LPTic has no effect on our results other than to decrease this loss of power, hence our use of a relatively fine 1440^3 mesh.

4.6 Conclusions

We have built upon the particle linear theory of Marcos et al. (2006) and shown how to eliminate transients at linear order in initial conditions for cosmological simulations. We consider the case of a simple cubic lattice of particles as the pre-initial configuration instead of a glass, because these dynamical discreteness effects are present in both cases but only analytically tractable in the particle lattice case. We then consider how such a system evolves in time and reproduce the PLT result that modes near $k_{\rm Nyquist}$ will be systematically suppressed as a simulation evolves. PLT gives the exact amplitude of this $\mathbf{k}$-dependent suppression, so we rescale the initial mode amplitudes such that they will arrive at the correct amplitudes at a later time, with the motivation of seeding non-linear evolution with the correct linear power spectrum. In a ΛCDM simulation with $z_{\rm init} = 49$ and $k_{\rm Nyquist} = 4\,h\ {\rm Mpc}^{-1}$, we show that the this suppression results in a 15% power deficit near $k_{\rm Nyquist}/2$ at $z = 5$, and that rescaling completely restores this power (Fig. 4.4).

[12]See Jenkins (2010) for a superior method of interpolating to a glass.

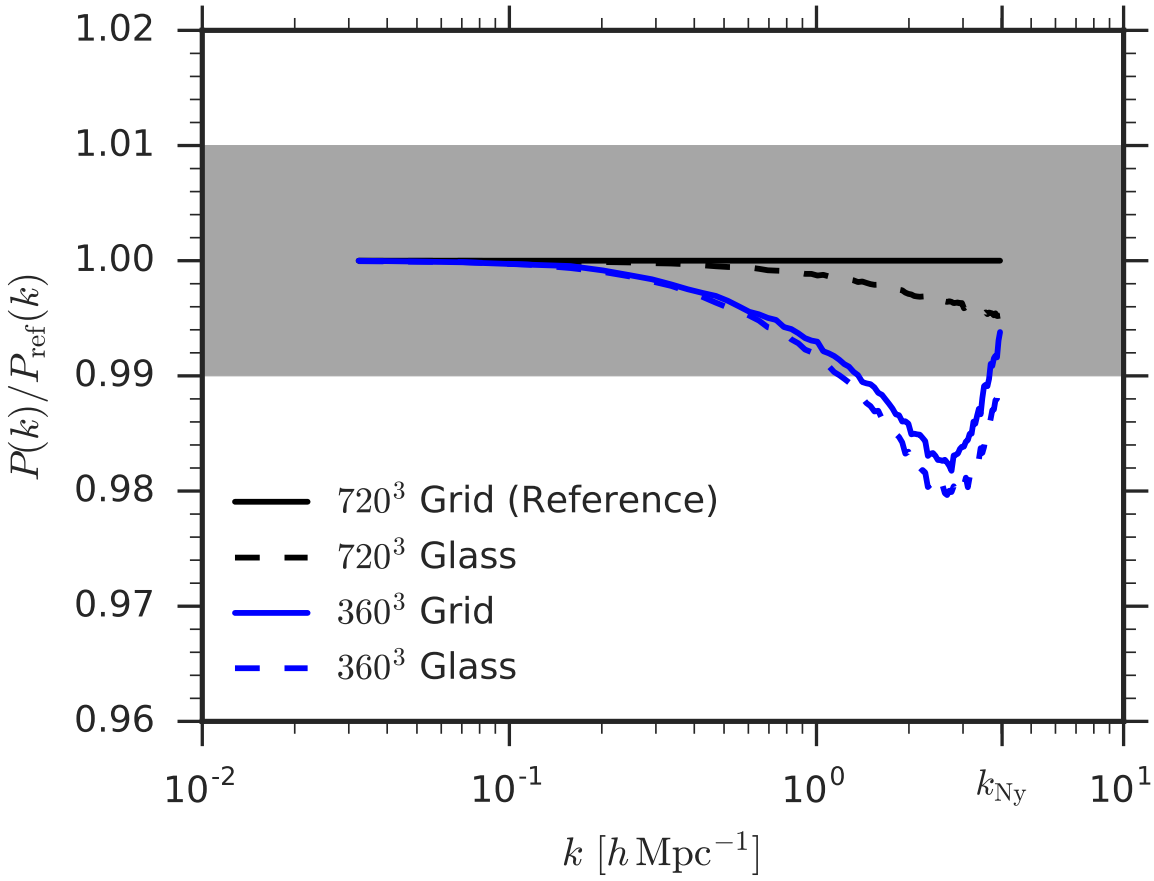

Figure 4.14: Power spectra at $z = 1$ for different pre-initial conditions. The grey shaded band indicates our target of 1% accuracy in the power spectrum. The 360^3 glass shows as much loss of power relative to the 720^3 glass as the 360^3 lattice does to the 720^3 lattice. In other words, glass initial conditions do not alleviate the small-scale suppression of power predicted by PLT. This suppression is due to the fact that the continuum density field is discretized, not the arrangement (glass or lattice) of the discretization. See §4.5.5.

We have also presented a new way to calculate second-order corrections in Lagrangian perturbation theory from direct force calculations, without the need for large Fourier transforms. We compare our 2LPT to the actual evolution of the particle system from $z = 4999$ and find excellent agreement on large scales, with differences of 1% at $k_{\rm Nyquist}/2$ due to particle lattice anisotropy. We also find excellent agreement with 2LPTic, a standard FFT-based code, below $k_{\rm Nyquist}/3$. Above that scale, both approaches have drawbacks: 2LPTic starts to introduce transverse-mode power (as any non-PLT code would), reaching 8% of power in transverse (i.e. transient) modes at $k_{\rm Nyquist}$; similarly, our 2LPT suffers a scatter of 4% due to anisotropic lattice effects at the same scale.

Finally, we have tested the impact of PLT, rescaling, and our 2LPT implementation on the matter power spectrum, halo masses, and halo two-point clustering at $z = 1$ in a series of 720^3 particle simulations initialized at $z = 49$ with particle mass $4 \times 10^{10} h^{-1}{\rm M}_\odot$. We compare the results to an oversampled reference simulation at 8 times the mass resolution. While our reference configuration does not represent an absolutely converged state, increasing particle density for a fixed set of modes will necessarily converge towards the continuum limit.

The power spectrum results confirm that the combination of 2LPT and rescaling is necessary to achieve 1% accuracy down to $k_{\rm Nyquist}$. 2LPT gives 1 to 3% errors below $k_{\rm Nyquist}/4$, corresponding roughly to 64 particle haloes. In other words, a simulation with 2LPT alone would need 64 times the mass resolution to achieve 1% accuracy at $k_{\rm Nyquist}$.

We identify haloes using three halo finders and found that, with a few exceptions, our results are largely independent of the finder. Specifically, at $z = 1$ we recover the known result (e.g. Crocce et al. 2006; L'Huillier et al. 2014) that 2LPT is important for the most

massive haloes (5 to 10% of haloes above 1000 particles are missing without 2LPT) and show that rescaling is necessary to correct a 5 to 10% deficit of haloes below 500 particles. These deficits increase at higher redshift, since the non-linearities of structure formation have not yet had time to transfer power from low k to high k. At all redshifts and mass ranges, 2LPT with rescaling was the best match to the reference simulation.

We analyse the halo 2PCF both in bins of absolute mass and mass rank. The absolute mass bins show the strongest dependence on the choice of ICs (at the level of 5%), because haloes are changing mass at constant bias, causing them to shift in or out of the mass bin. Binning by mass rank greatly reduces this effect and lowers the dependence on the ICs to the level of 1 to 3%. In nearly all cases, the combination of 2LPT and rescaling produces the best match to the reference 2PCF, reaching agreement of a fraction of a percent in many cases.

Our PLT growing mode corrections are manifestly the correct way to initialize cosmological N-body simulations to linear order. These corrections, in combination with rescaling and our 2LPT implementation, are crucial for recovering accurate small-scale power spectra, halo masses, and clustering. We anticipate that our 2LPT implementation will be particularly useful for extremely large N-body simulations, where large FFTs are expensive. Rescaling may be most useful for arrays of medium-resolution simulations in which substantially increasing the particle density is too expensive computationally, e.g. when constructing covariance matrices or building cosmic emulators (e.g. Lawrence et al. 2010).

Code to generate ZA initial conditions with PLT eigenmode corrections and rescaling is available at `https://github.com/lgarrison/zeldovich-PLT`.

Acknowledgements

We thank Svetlin Tassev for helpful discussions and the referee for comments that helped improve the quality of this paper. We acknowledge use of the University of Washington N-body Shop friends-of friends code. Some computations in this paper were run on the Odyssey cluster supported by the FAS Division of Science, Research Computing Group at Harvard University. This work has been supported by grant AST-1313285 from the National Science Foundation and by grant DE-SC0013718 from the U.S. Department of Energy. Some of the computations used in this study were performed on the El Gato supercomputer at the University of Arizona, supported by grant 1228509 from the National Science Foundation.

Chapter 5

Abacus Cosmos: A Suite of Cosmological N-body Simulations

This thesis chapter was originally published as

Garrison, Lehman H., Daniel J. Eisenstein, Douglas Ferrer, Jeremy L. Tinker, Philip A. Pinto, David H. Weinberg, ApJS, 236, 43

Abstract

We present a public data release of halo catalogs from a suite of 125 cosmological N-body simulations from the Abacus project. The simulations span 40 wCDM cosmologies centered on the Planck 2015 cosmology at two mass resolutions, $4 \times 10^{10} h^{-1}$ $\mathrm{M}_\odot$ and $1 \times 10^{10} h^{-1}$ $\mathrm{M}_\odot$, in $1.1 h^{-1}$ Gpc and $720 h^{-1}$ Mpc boxes, respectively. The boxes are phase-matched to suppress sample variance and isolate cosmology dependence. Additional volume is available via 16 boxes of fixed cosmology and varied phase; a few boxes of

single-parameter excursions from Planck 2015 are also provided. Catalogs spanning $z = 1.5$ to 0.1 are available for friends-of-friends and ROCKSTAR halo finders and include particle subsamples. All data products are available at `https://lgarrison.github.io/AbacusCosmos`.

5.1 Introduction

High-precision forward modeling of large-scale structure is a necessary complement to upcoming galaxy surveys, including DESI (Levi et al. 2013), *WFIRST* (Spergel et al. 2015), *Euclid* (Laureijs et al. 2011), and LSST (LSST Science Collaboration et al. 2009), that will catalog tens of millions of galaxies in unprecedentedly large volumes. Mock catalogs must be available that allow design of survey strategies and testing of cosmological parameter estimation and covariance. Although many fast approximate methods exist for generating such catalogs (e.g. COLA (Tassev et al. 2013) and QPM (White et al. 2014); see Monaco (2016) for a recent review), the highest quality mocks remain those derived from N-body simulations. These gravity-only simulations directly evolve collisionless Monte Carlo tracers of the matter density field from the early, linear regime to late, non-linear times when galaxies form. These simulations are fraught with challenges, from bias introduced by discretization at early times (e.g. Joyce & Marcos 2007a; Garrison et al. 2016), to artificial relaxation at late times (e.g. Diemand et al. 2004; Power et al. 2016), to lack of hydrodynamical and baryonic physics (Mead et al. 2015; Schneider & Teyssier 2015; Schaller 2015, and references therein), to disagreements among N-body solvers (Schneider et al. 2016a) and among halo finders (Behroozi et al. 2015; Knebe et al. 2013), to the sheer computational challenge of evolving $\mathscr{O}(10^{12})$ self-interacting particles, the scale that will

be needed within a few years (Schneider et al. 2016a; Potter et al. 2016). Nevertheless, *N*-body remains an invaluable tool in testing models of large-scale structure.

Given the computational expense of *N*-body, most public data products lie at one of two extremes: full halo catalogs available at a fixed cosmology, or reduced data products that allow variations in the cosmology. Some examples of the former are simulations like Bolshoi and MultiDark (Riebe et al. 2013), Las Damas (McBride et al. 2009), and Millennium (Lemson & Virgo Consortium 2006), which provide halo catalogs but focus on a single fiducial cosmology. Examples of the latter are tools like CosmicEmu (Lawrence et al. 2017) or fitting formulae like those found in Tinker et al. (2008) or Comparat et al. (2017) for common data products like the power spectrum or halo mass function. These tools take cosmological parameters as inputs but do not provide access to the underlying halo catalogs or particles, so the science applications are limited. We aim to bridge these two extremes by providing a suite of many different cosmologies with access to the underlying halos and subsamples of the particles. In this sense, our approach is most similar to `skiesanduniverses.org` (Klypin et al. 2017) because the focus is access to raw halos and particles.

In the next section, we introduce the code used to run the simulations, and in §5.3 we discuss various parameter choices for the simulations and initial conditions. In §5.4, we introduce the design of the Latin hypercube grid of 40 cosmologies that form the core of our simulation effort, and in §5.5 we provide an overview of all the available sets of simulations. In §5.6, we detail the data products that we generate for each simulation, and in §5.7 we validate their accuracy. We summarize in §5.8.

5.2 Abacus: fast and precise N-body cosmology

5.2.1 Overview

The simulations in this work all employ the N-body code Abacus, described in Chapter 2 of this thesis. In this section, we will focus on the application of Abacus to the El Gato system. Each simulation here was run on a single node; indeed the parallel version of Abacus was not implemented at the time of this work.

5.2.2 Hardware and performance

Most of the simulations in this work were run on the El Gato cluster at the University of Arizona. The GPU nodes are dual-socket machines with two 8-core Intel Ivy Bridge E5-2650v2 processors (2.6 GHz) and 256 GB DDR3 RAM (1800 MHz). Abacus is not designed to hold all particles in memory but rather stream them from disk, so the I/O demands are fairly high. In many cases, we use hardware RAID, but for these simulations our strategy was to choose a problem size that would fit on a ramdisk (a filesystem hosted in RAM). A local scratch disk or network filesystem would typically not have the bandwidth to handle Abacus's I/O demands. Thus, the choice of 1440^3 particles was set by the size of the system ramdisk.

Our typical speeds at the outset of this project were about $4\,\mathrm{s}^{-1}\mathrm{Mpart}$ (million particle updates per second) per timestep with about 20% of this time in the convolution step between each primary timestep; later code improvements increased this to about $12\,\mathrm{s}^{-1}\mathrm{Mpart}$. This higher speed is still a factor of two lower than typical speeds on our production machines (which were not used in this work), in part due to the extra load on

the memory bandwidth from the ramdisk. The simulations took roughly 1000 timesteps to reach the final redshift from $z_{\rm init} = 49$. The large number of timesteps is due to the lack of adaptive timestepping, so the global timestep is set by the shortest dynamical time in the whole box. This will be addressed with on-the-fly group finding and multi-stepping within those groups in future versions of ABACUS.

5.2.3 Force softening

Several force softening options are available in ABACUS. The simplest is Plummer softening, where the $\mathbf{F}(\mathbf{r}) = \mathbf{r}/r^3$ force law is modified as

$$\mathbf{F}(\mathbf{r}) = \frac{\mathbf{r}}{(r^2 + \varepsilon_p^2)^{3/2}}, \tag{5.1}$$

where ε_p is the softening length. This softening is very fast to compute but is not compact, meaning it never explicitly switches to the exact r^{-2} form at any radius (in contrast with spline softening). This modifies the growth of structure on large scales (Joyce & Marcos 2007a). This is the softening we employ for the `emulator_1100box_planck` and `emulator_720box_planck` sets of simulations (see §5.5).

An alternative is spline softening, in which the force law is softened for small radii but explicitly changes to the unsoftened form at large radii. Traditional spline implementations split the force law into three or more piecewise segments (e.g. the cubic spline of Hernquist & Katz (1989)); we split only once for computational efficiency[1] and call this `single_spline`. We derive this form by considering a Taylor expansion in r of Plummer

[1]Multiple splits can cause code path branching, which incurs a significant performance penalty on CPUs and an even larger one on GPUs.

softening (Eq. 5.1) and requiring a smooth transition at the softening scale up to the second derivative[2]. This gives

$$\mathbf{F}(\mathbf{r}) = \begin{cases} \left(10 - 15(r/\varepsilon_s) + 6(r/\varepsilon_s)^2\right)\mathbf{r}/\varepsilon_s^3, & r < \varepsilon_s; \\ \mathbf{r}/r^3, & r >= \varepsilon_s. \end{cases} \tag{5.2}$$

This is the softening we employ for the `AbacusCosmos_1100box` and `AbacusCosmos_720box` simulation sets.

The softening scales ε_s and ε_p imply different minimum dynamical times (an important property, as this sets the requisite temporal resolution to resolve orbits). We always set the softening length as if it were a Plummer softening and then internally convert to a softening length that gives the same minimum dynamical time for the chosen softening method. For `single_spline`, the conversion is $\varepsilon_s = 2.16\varepsilon_p$.

5.3 Simulation Details

We present technical details of the simulation configuration here. For an overview of the available sets of simulations, see §5.5.

[2] A Taylor expansion in r^2 is also possible, but we discard that solution due to a large plateau of constant angular frequency near $r \sim 0$ that could excite dynamical instabilities.

5.3.1 Initial conditions

The initial conditions were generated by the public `zeldovich-PLT`[3] code of Garrison et al. (2016). We do not provide initial conditions files with the catalogs, but we do provide the input parameter file (`info/abacus.par`) for the IC code and the input power spectrum from CAMB (Lewis et al. 2000). The initial conditions can thus be generated by re-running the IC code with those inputs.

The simulations use second-order Lagrangian perturbation theory (2LPT) initial conditions, but `zeldovich-PLT` only outputs first order displacements. The 2LPT corrections are generated by Abacus on-the-fly using the configuration-space method of Garrison et al. (2016).

Two non-standard first-order corrections are implemented by `zeldovich-PLT`. The first is that the displacements use the particle lattice eigenmodes rather than the curl-free continuum eigenmodes. This eliminates transients that arise due to the discretization of the continuum dynamical system (the Vlasov-Boltzmann distribution function) into particles on small scales near $k_{\rm Nyquist}$. The second correction is "rescaling", in which initial mode amplitudes are adjusted to counteract the violation of linear theory that inevitably happens on small scales in particle systems. This violation usually takes the form of growth suppression; thus the initial adjustments are mostly amplitude increases. We choose $z_{\rm target} = 5$ as the redshift at which the rescaled solution will match linear theory; this choice is tested in Garrison et al. (2016).

Later, we will refer to simulations that are "phase-matched" in the initial conditions.

[3] `https://github.com/lgarrison/zeldovich-PLT`

This refers to initial conditions with the same random number generator seed, called `ZD_Seed` in the `abacus.par` file. Matching this value (and `ZD_NumBlock`) between two simulations guarantees that the amplitudes and phases of the initial modes are identical between the simulations (up to differences in the input power spectrum and cosmology).

5.3.2 Input power spectrum

We use CAMB (Lewis et al. 2000) to generate a linear $z = 0$ power spectrum for each cosmology in our grid. We then scale the power spectrum back to $z_{\rm init} = 49$ by scaling σ_8 by the ratio of the growth factors $D(z = 49)/D(z = 0)$. This σ_8 is passed to `zeldovich-PLT`, which handles the re-normalization of the power spectrum. The computation of the growth factors is done by ABACUS's cosmology module, so it is consistent by construction with the simulation's cosmological evolution. We only use massless neutrinos and include no cosmological neutrino density. The exact CAMB inputs and outputs are available with each simulation.

In the computation of the power spectrum, baryons and CDM are treated as separate species; however, ABACUS is a gravity-only N-body solver (i.e. no hydrodynamics or baryonic physics), so we simply combine the baryonic and CDM density when computing the overall mass density of the universe. It is this combined mass density that sets our particle mass.

5.3.3 Code parameters

The important ABACUS parameters (including those that might affect code accuracy) are given in Table 5.1 and described here. The simulations were run with a mix of force softening laws. The "`ZD`" parameter prefix indicates that this is an input to our `zeldovich-PLT` IC code.

`LagrangianPTOrder` The order of Lagrangian perturbation theory corrections to compute at runtime (see §5.3.1).

`Order` The multipole order used for computing the far-field force.

`SofteningLength` The comoving Plummer-equivalent force softening length (see §5.2.3).

`TimeSliceRedshifts` The output redshifts. The last slice ($z = 0.1$) is only available for the higher resolution "`720box`" simulations.

`TimeStepAccel` Parameter to limit the time step based on particle accelerations. The time step is set such that $a_{\rm max}\Delta t/v_{\rm rms}$ is never larger than this parameter. This often sets the choice of time step at late times. Various choices of this parameter are tested in Ferrer, et al. (in prep.); 0.15 is considered a conservative choice.

`TimeStepDlna` Maximum $\Delta(\ln a)$ allowed for a time step. This often sets the choice of time step at early times. For example, 0.03 means at least 33 steps per e-folding of the scale factor.

`ZD_PLT_target_z` The target redshift for PLT rescaling of the initial conditions (see §5.3.1 for a description of PLT corrections).

Parameter	Value
`LagrangianPTOrder`	2
`Order`	8
`SofteningLength`	$63h^{-1}$ or $41h^{-1}$ kpc
`TimeSliceRedshifts`	1.5, 1.0, 0.7, 0.5, 0.3, [0.1]
`TimeStepAccel`	0.15
`TimeStepDlna`	0.03
`ZD_PLT_target_z`	5
`ZD_qPLT`	1
`ZD_qPLT_rescale`	1
Max (median) force error	1×10^{-4} (2×10^{-6})

Table 5.1:: ABACUS code parameters, described in §5.3.3. The force error is the maximum fractional error on the unsoftened forces on a set of 66K randomly distributed particles compared to the true $1/r^2$ forces computed with an Ewald summation in 256-bit precision. The last `TimeSliceRedshift` is only present for the higher-resolution simulations.

`ZD_qPLT` Initialize the displacements and velocities in the eigenmodes of the particle system.

`ZD_qPLT_rescale` Do PLT rescaling.

5.4 Cosmology Grid Design

The 40 cosmologies listed in Table 5.2 are distributed in a 6-dimensional wCDM parameter space according to a Latin hypercube algorithm (see Fig. 5.1 for a visual representation). Specifically, the algorithm samples a $(H_0, \Omega_M h^2, \Omega_b h^2, \sigma_8, n_s, w_0)$ space centered on the Planck 2013 cosmology (Planck Collaboration et al. 2014). We use $N_{\rm eff} = 3.046$ in all cases. These 40 cosmologies are realized in the two `AbacusCosmos` simulation sets.

The goal of the cosmological parameter selection is to evenly span the parameter space with only a limited number of cosmologies. The procedure used here follows the Latin hypercube method described in Heitmann et al. (2009). In the Latin hypercube method, each of the N dimensions is divided into M bins, with M being the number of desired cosmologies. Each bin in each dimension is only sampled once, thus guaranteeing that the entire range of parameter space is covered in the set of cosmologies. The hypercube is optimized such that, for each cosmology, the distance to the nearest neighboring cosmology is maximized.

The optimized hypercube is then rotated into the parameter space defined by the union of WMAP 9-year (Hinshaw et al. 2013) and Planck 2013 (Planck Collaboration et al. 2014) CMB results, combined with recent BAO and SN results. The results used to define this space are obtained from Anderson et al. (2014). The axes of the Latin

hypercube then correspond to the eigenvectors of the CMB-defined parameter space. In the original hypercube design, each axis ranges from 0 to 1. In the CMB parameter space, these ranges correspond to -4 to 4 times the eigenvalue along each eigenvector, so we are sampling from the 4-sigma CMB constraints.

Although the Planck 2013 results were used in the hypercube design instead of the 2015 results, the resulting cosmologies span a larger space than allowed by either data set. Thus, derivatives measured from these simulations should be equally useful when assuming a fiducial cosmology of either Planck 2013 or 2015. Indeed, our fiducial cosmology for the `emulator_planck` simulations is Planck 2015, and it falls nearly at the center of the cosmology hypercube (the blue square in Fig. 5.1).

5.5 Catalogs

We present four collections of simulations organized into "sets". Sets have names like `AbacusCosmos_1100box`, while individual simulations have a two-digit number appended to them like `AbacusCosmos_1100box_00`. The two-digit number refers to the cosmology. Different phases of the same cosmology are indicated by a dash and number after the cosmology, as in `emulator_1100box_planck_00-1`.

The four sets of simulations are as follows:

`AbacusCosmos_1100box` 41 phase-matched[4] boxes: 40 spanning the 6-dimensional wCDM cosmology parameter space (§5.4) and one with the fiducial Planck cosmology. Each

[4]"Phase-matched" means the initial conditions random number generator was given the same seed. This ensures differences between boxes are due to cosmology and not cosmic variance; see §5.3.1.

	H_0	$\Omega_{\rm DE}$	Ω_M	n_s	σ_8	w_0
00	69	0.698	0.302	0.93	0.854	-1.14
01	63	0.669	0.331	0.982	0.719	-0.765
02	72.3	0.739	0.261	0.97	0.851	-1.08
03	66.2	0.671	0.329	0.975	0.858	-0.982
04	74.2	0.733	0.267	0.954	0.889	-1.22
05	71.5	0.706	0.294	0.954	0.913	-1.29
06	68.6	0.717	0.283	0.96	0.768	-0.969
07	66.7	0.709	0.291	0.987	0.734	-0.788
08	65.8	0.695	0.305	0.957	0.692	-0.796
09	72.7	0.719	0.281	0.931	0.89	-1.24
10	67	0.677	0.323	0.961	0.83	-0.995
11	72.2	0.732	0.268	0.963	0.851	-1.16
12	65.2	0.687	0.313	0.99	0.778	-0.827
13	64.1	0.67	0.33	0.976	0.776	-0.792
14	74.8	0.731	0.269	0.971	0.999	-1.37
15	67.4	0.708	0.292	0.976	0.732	-0.89
16	70.9	0.727	0.273	0.969	0.835	-0.999
17	62.1	0.658	0.342	0.95	0.714	-0.745
18	67.9	0.693	0.307	0.97	0.831	-0.934
19	71	0.717	0.283	0.955	0.849	-1.04
20	64.5	0.675	0.325	0.967	0.74	-0.742
21	68.3	0.689	0.311	0.975	0.847	-1.02
22	73.3	0.723	0.277	0.937	0.923	-1.3
23	62.7	0.645	0.355	0.965	0.77	-0.796
24	73.9	0.731	0.269	0.966	0.938	-1.24
25	61.6	0.633	0.367	0.965	0.728	-0.754
26	69.3	0.705	0.295	0.963	0.831	-1.06
27	62.5	0.663	0.337	0.959	0.687	-0.655

	H_0	$\Omega_{\rm DE}$	Ω_M	σ_8
00 & 00-0 to 00-15	67.3	0.686	0.314	0.83
01	67.3	0.686	0.314	**0.78**
02	67.3	0.686	0.314	**0.88**
03	**64.3**	**0.656**	**0.344**	0.83
04	**70.3**	**0.712**	**0.288**	0.83

Table 5.3:: The cosmologies for the `emulator_1100box_planck` and `emulator_720box_planck` sets of simulations, all with $n_s = 0.965$, $w_0 = -1$, and $N_{\rm eff} = 3.04$. The first cosmology represents fiducial parameters for which 17 boxes with different phases were run. The latter four represent "derivative" boxes in which one parameter (in bold) is changed at a time. Note that the cosmologies are actually chosen in the space of physical densities $\Omega_x h^2$, which is why a change in H_0 results in change in Ω_x.

box has size $1100h^{-1}$ Mpc and particle mass $4\times10^{10}h^{-1}$ $M_\odot$. Spline softening of $63h^{-1}$ kpc.

`AbacusCosmos_720box` Same as the above, but at higher mass resolution ($1\times10^{10}h^{-1}$ $M_\odot$) and smaller box size ($720h^{-1}$ Mpc). Note that these are not zoom-in simulations of the larger boxes, but independent realizations of the power spectrum. Spline softening of $41h^{-1}$ kpc.

`emulator_1100box_planck` 21 boxes: 16 with identical Planck cosmologies but different IC phases, and 5 phase-matched boxes of single-parameter variations from the Planck cosmology ($\pm5\%$ in σ_8 and H_0, plus a central Planck box). All boxes have size $1100h^{-1}$ Mpc and particle mass $4\times10^{10}h^{-1}$ $M_\odot$. Plummer softening of $63h^{-1}$ kpc.

`emulator_720box_planck` Same as the above, but at higher mass resolution ($1\times10^{10}h^{-1}$ $M_\odot$) and smaller box size ($720h^{-1}$ Mpc). As with the `AbacusCosmos` boxes, these are not zoom-in simulations. Plummer softening of $41h^{-1}$ kpc.

The 40 `AbacusCosmos` simulations are designed to allow estimation of derivatives of cosmological measurables with respect to cosmology. They can either be used as an ensemble to construct an emulator/interpolator, or each individual box can be differenced with the central `AbacusCosmos_planck` box to provide an estimate of the derivative for that particular change in cosmology.

The boxes do not provide much cosmic volume of any individual cosmology ($1.7h^{-3}\,\mathrm{Gpc}^3$ if both resolutions are combined), but the 17 phase-varied `emulator_planck` boxes provide additional volume and a path toward suppressing cosmic variance and estimating covariance. The 4 `emulator_planck` single-parameter excursion boxes provide a more

direct route to measuring derivatives but only for two parameters.

The motivation for two mass resolutions was first to provide a convergence test for large-scale structure properties, at least in the intermediate regime well-sampled by both resolutions (see §5.7.2). Second, the larger boxes provide the volume that is needed for BAO-type studies (Klypin & Prada 2017 argue that modes longer than $1h^{-1}$ Gpc have very little impact on the matter power spectrum), while the smaller boxes provide the halo resolution that is needed by weak-lensing studies (see e.g. Wibking et al. 2019).

Our softening lengths — $41h^{-1}$ and $63h^{-1}$ kpc in the two box resolutions — were chosen to support halos, not subhalos. Future enhancements to ABACUS should make it possible for us to reach dramatically smaller softening lengths. In the near term, we are continuing to run simulations at these mass scales and resolutions to build volume. These simulations will be made available on the release website as they finish.

In total, these 125 simulations represent roughly 200K CPU-hours, or 25K GPU-hours, of computational effort. This is a relatively modest amount for a collection of 370 billion particles and is a testament to ABACUS' computational efficiency.

5.6 Data products: halos and power spectra

We provide three data products at every redshift slice: friends-of-friends (FoF) halos, ROCKSTAR halos, and a high-resolution matter power spectrum. The redshift slices are $z = \{1.5, 1.0, 0.7, 0.5, 0.3, [0.1]\}$, where the $z = 0.1$ slice is only provided for the higher resolution "`720box`" simulations. Particle subsamples are included with the catalogs. The data formats of each product are described on the data release website.

5.6.1 Friends-of-friends

The friends-of-friends (FoF) algorithm links particles separated by less than a linking length b, equal to 0.186 in our catalogs (expressed as a fraction of the mean particle spacing). A halo is defined as a set of linked particles (Davis et al. 1985). Our implementation is based on the University of Washington N-body Shop's halo finder, and we include halos down to 25 particles. The linking length 0.186 was chosen to correspond to an overdensity contour of 100 times the background density using the percolation theory results of More et al. (2011).

We compute a number of halo properties relative to the FoF centers of mass and velocity, including velocity dispersion, circular velocity profiles, and radial quantiles. However, the center of mass is susceptible to "barbell" pathologies, in which two largely distinct halos are connected by a chance alignment of a thin particle "bridge". To guard against this, we also compute a second level of FoF with a smaller linking length (equal to 0.117 in our catalogs). This linking length was chosen to capture half of the mass of a singular isothermal sphere. We store the masses of the 4 most massive subhalos and additionally compute global halo properties centered on the most massive subhalo. These allow for a first-order defense against common FoF pathologies.

We provide a 10% subsample of particles both inside and outside halos ("halo particles" and "field particles"). Particles are assigned to be subsample particles as a pseudo-random function of the particle ID, so the same 10% of particles are output at every timestep. This allows for construction of crude merger trees and other simple box-to-box comparisons. A uniform sampling of the density field (that is, a 10% sample of all particles) can be formed via the union of the halo and field particle subsamples. Particle

information includes positions, velocities, and IDs.

5.6.2 Rockstar

ROCKSTAR (Behroozi et al. 2013) is a hierarchical halo finder that uses friends-of-friends in six-dimensional phase space at successively smaller linking lengths to find halos and substructure. The inner-most substructure defines a halo seed to which particles are assigned based on their phase-space proximity. ROCKSTAR also can use temporal information (multiple time slices) to improve structure tracking, but we do not use this mode as the time between our outputs is large. We run ROCKSTAR with mostly default settings, including the default mass definition of M_{vir}; however, we report both regular ROCKSTAR masses and strict spherical overdensity masses. Strict spherical overdensity masses contain all particles, including those considered "unbound" and those not associated with the halo. Subhalos are reported and tagged with their parent halo as decided by ROCKSTAR. We also output a 10% subsample of particles in halos. The exact ROCKSTAR input file is available with each catalog (`rockstar.cfg`).

5.6.3 Power spectra

We compute the matter power spectra by gridding the particles onto a mesh (2048^3 or finer) with triangle-shaped cloud (TSC) mass assignment. We then Fourier transform the density field, convert the result to a power spectrum, de-convolve the TSC-aliased window function from Jeong (2010), and bin in spherical annuli. The resulting 1D power spectrum is available for every simulation time slice.

5.6.4 Plummer vs. spline data products

The two `AbacusCosmos` sets were run with spline softening, while the `emulator_planck` boxes were run with Plummer softening (§5.2.3). The spline was developed partway through the simulation campaign and was thus only applied to the remaining sets of simulations. However, we also re-ran one of the Plummer boxes with spline softening to calibrate differences in the data products between spline and Plummer. Those results are presented here. The simulation is also available on the data release website as `emulator_1100box_planck_spline_00` if further calibrations are required.

In Fig. 5.2, we show halo mass functions from the friends-of-friends halo finder for Plummer and spline. The differences are small ($< 3\%$ for halos above 100 particles), but the number of Plummer halos steadily increases with increasing mass (except for the highest mass bin which has very few halos). The Plummer softening may be inflating halos, causing large halos to come in contact with neighboring structures, increasing their mass. At the low-mass end, inflating small, tenuously bound halos (below 100 particles) may be causing them to unbind, resulting in fewer halos. These trends are almost completely insensitive to redshift. ROCKSTAR and ROCKSTAR SO halos behave very similarly at the low mass end, but Plummer under-predicts the number of > 1000 particle halos by 1 to 3%. This is consistent with the picture that high-mass halos are inflated with Plummer softening, since ROCKSTAR is less susceptible than FoF to over-merging of inflated structures.

In Fig. 5.3, we show matter power spectra for Plummer and spline softening. As expected, Plummer results in a loss of power at high k. At the Nyquist wavenumber of the particle lattice, Plummer misses 6% of power compared to spline. This loss of power

due to non-compact gravitational softening is a known phenomenon; see e.g. Joyce & Marcos (2007a); Garrison et al. (2016). As with halo mass, there is almost no evolution of this relation with redshift.

5.6.5 Example Python Interfaces

Example Python code is provided on the release website to load and manipulate the halo catalogs, including particle subsamples. The data formats are also documented on the website, allowing any user to write their own code to parse the catalogs. However, the provided Python code can at least serve as an implementation example of the data specifications. A wrapper that loads the halo catalogs into HALOTOOLS[5] format (Hearin et al. 2017) is also provided.

Another halo occupation distribution (HOD) code that is well-integrated with the Abacus Cosmos simulation suite is GRAND-HOD[6]. In particular, GRAND-HOD can use the halo particle subsamples when populating a halo with satellite galaxies.

5.7 Validation

5.7.1 CosmicEmu and HaloFit

We validate our power spectrum results against CosmicEmu (Lawrence et al. 2017) which produces a power spectrum as a function of cosmology. However, the AbacusCosmos

[5] http://halotools.readthedocs.io

[6] https://github.com/SandyYuan/GRAND-HOD

cosmologies span a slightly larger parameter space than CosmicEmu, so we are only able to do this test for 28 of our 40 sims.

Fig. 5.4 shows that `AbacusCosmos` and CosmicEmu are in very good agreement (at the level of a few percent) for $k \sim 0.03$ to 4 at z=0.5. At the low-k end, we see large deviations due to cosmic variance, or finite box size[7]. At the high-k end, we see a downturn in the Abacus power spectra past the Nyquist wavenumber of the particle lattice. This downturn is expected due to the failure of particle systems reproduce linear theory near, and especially past, $k_{\rm Nyquist}$ (Joyce & Marcos 2007a; Garrison et al. 2016).

Fig. 5.5 shows the agreement of the `emulator_planck` simulations with CosmicEmu and HaloFit (Takahashi et al. 2012), as invoked through CAMB's `do_nonlinear` feature at $z = 0.3$. In particular, it shows that the low k scatter in the power spectrum seen in Fig. 5.4 can be suppressed by averaging the results of many boxes (recall that these boxes have fixed cosmology but different IC phases). The agreement is within 4% (the accuracy quoted by CosmicEmu) for $k \sim 0.01$ to 3, with a downturn before $k_{\rm Nyquist}$ due to the Plummer softening (see also Fig. 5.3). The `emulator_720box_planck_00-0` simulation shows a small systematic deviation around $k = 1$ from the mean Abacus result, but we find no evidence for misbehavior of this simulation in our diagnostics.

5.7.2 Convergence

We compare simulation data products in the intermediate regime well-sampled by both `emulator_720box_planck` and `emulator_1100box_planck`. We examine the power spec-

[7]For statistics that need more cosmological volume, this low-k scatter can be averaged down with the `emulator_planck` sims, which have 17 boxes of the same cosmology but different phases. See Fig. 5.5.

trum and halo mass function at three different redshifts. In all cases, we average over all 17 boxes at each resolution to suppress sample variance.

The matter power spectrum agreement in Fig. 5.6 is excellent from $k = 0.03$ to $k = 6$ (the Nyquist wavenumber of the higher-resolution box), indicating that our results are stable with respect to box size and mass resolution. To compare the matter power spectra at incommensurate wavenumbers, we applied a cubic spline in log-log space to the reference spectrum. However, at the smallest k the binning breaks the smoothness of the power spectrum, so the cubic spline appears to slightly over-predict the disagreement between the resolutions. Regardless, agreement in this regime is cosmic-variance limited even after averaging over 17 boxes, so this discrepancy is not concerning.

The FoF halo mass function is converged to within 6% in the regime sampled by at least 100 particles at both mass resolutions (Fig. 5.7). The lower mass resolution systematically overproduces small halos; this is a known effect in friends-of-friends that arises from the mismatch in spatial stochasticity in the particle sampling at the two resolutions (More et al. 2011). See figure 10 of (Garrison et al. 2016) for a very similar test, in which downsampling the higher-resolution simulation before running FoF produces excellent agreement with the lower-resolution result. With ROCKSTAR, the downturn effect almost entirely disappears (Fig. 5.8).

When considering ROCKSTAR SO masses, however, a substantial deviation is seen for all but the largest halos (Fig. 5.9). At 100 particles (400 particles at the higher resolution), 30% more halos are found at the higher resolution. We speculate that the higher resolution box finds more physically small halos (due to mass and softening resolution) near large halos; these small halos then have their SO masses inflated by the presence of nearby

structure since ROCKSTAR allows overlapping SO spheres and "double-counting" of mass.

5.8 Summary

We have presented a suite of cosmological N-body simulations produced by the new ABACUS code. The modest computational requirements of ABACUS (a single GPU node for a few days) enabled us to run one hundred twenty-five ~ 1 Gpc boxes, each with 3 billion particles. These boxes span 40 cosmologies near Planck 2015, allowing for emulation/interpolation in this important parameter region. The accompanying halo catalogs include particle subsamples, allowing for detailed investigations of galaxy bias models and other effects sensitive to dark matter halo structure like weak lensing. The data products are publicly available and include example code to manipulate the catalogs.

As of May 2019, these simulations have been used in refereed works by ourselves and others in Hada & Eisenstein (2018); Yuan et al. (2018); Wibking et al. (2019); Garrison & Eisenstein (2019); Yuan & Eisenstein (2019).

Acknowledgements

We would like to thank the Research Computing Group in the FAS Division of Science at Harvard University for their assistance in hosting the catalogs. LHG would like to thank Ben Wibking, Andres Salcedo, and Sihan Yuan for their feedback as "alpha" testers of these catalogs, Nina Maksimova for helpful discussions on the Abacus algorithm, and the referee for comments that helped improve the quality of the work. Our FoF implementation is

based on the University of Washington N-body Shop friends-of friends code. This work has been supported by grant AST-1313285 from the National Science Foundation and by grant DE-SC0013718 from the U.S. Department of Energy. Some of the computations used in this study were performed on the El Gato supercomputer at the University of Arizona, supported by grant 1228509 from the National Science Foundation. PAP was supported by NSF AST-1312699.

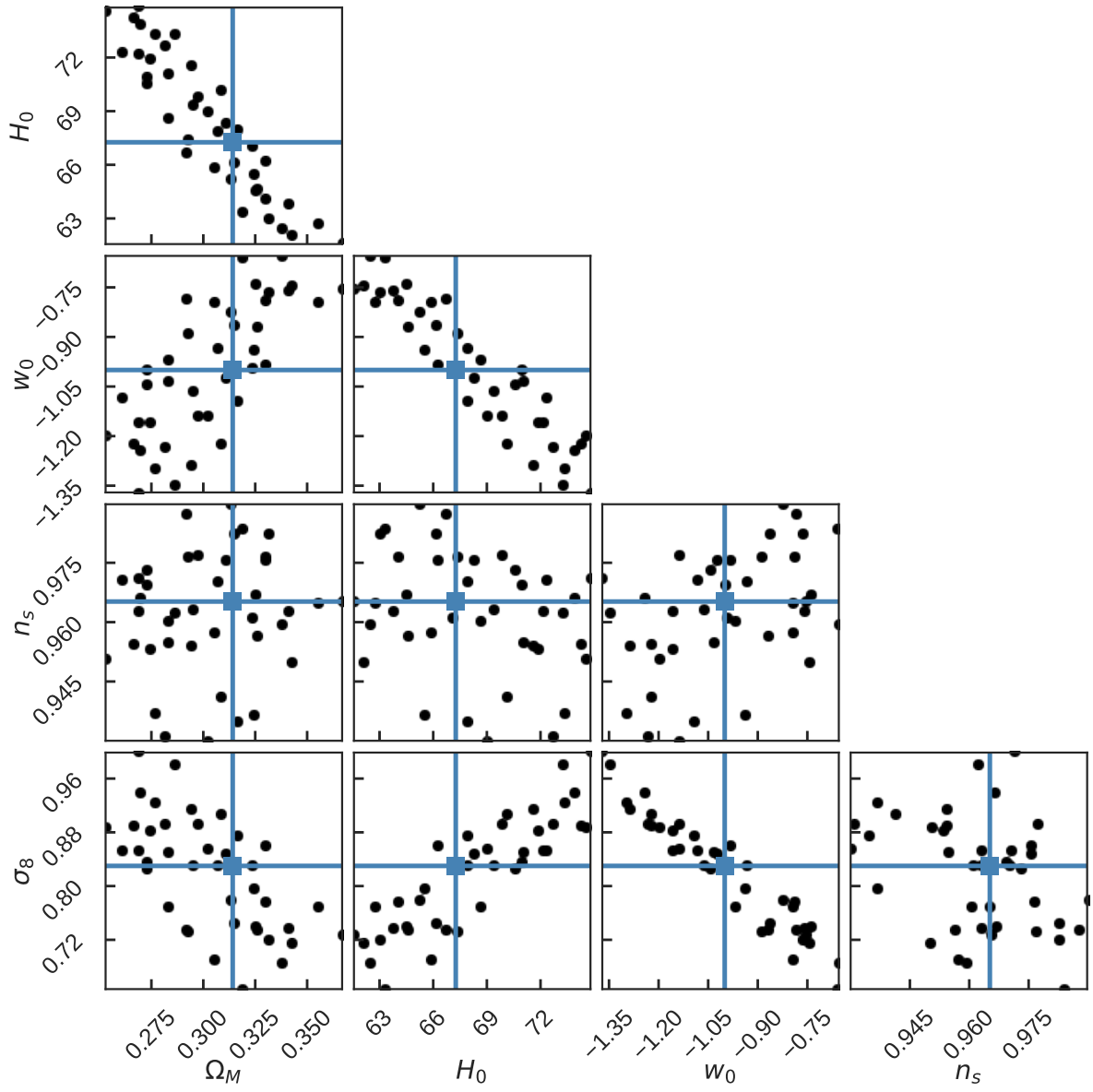

Figure 5.1: A corner plot representation of the cosmology space spanned by the `AbacusCosmos` simulations, where we have combined $\Omega_{\rm CDM}$ and Ω_b into Ω_M. The blue square marks the fiducial central cosmology.

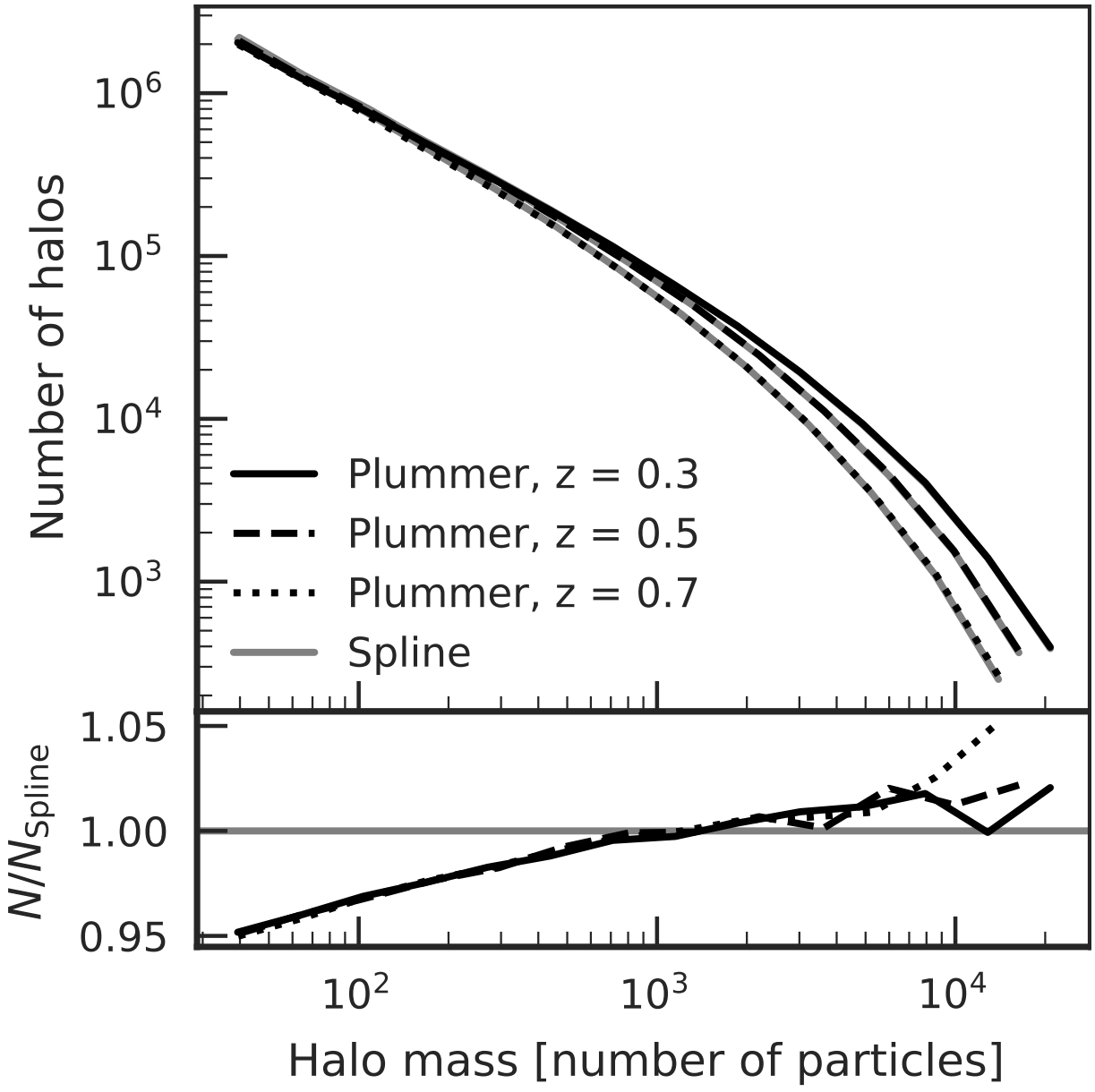

Figure 5.2: Comparison of the FoF halo mass function of two identical simulations that differ only in the force softening technique (Plummer or spline). For halos larger than 100 particles, the differences are consistently small ($< 3\%$), although the relative number of Plummer halos steadily increases with increasing mass. There is almost no evolution of this relation with redshift.

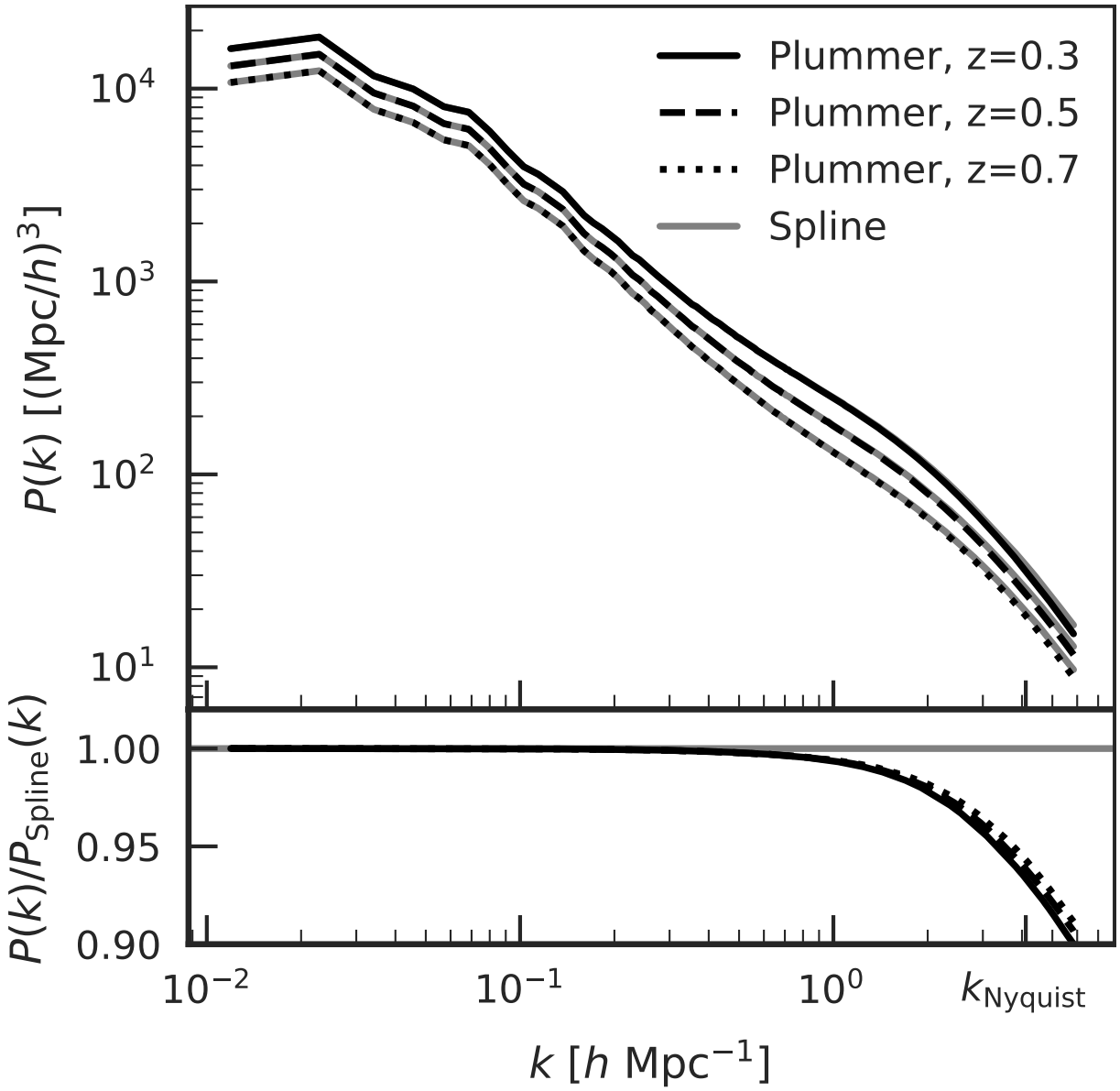

Figure 5.3: Comparison of the matter power spectra of two identical simulations that differ only in the force softening technique (Plummer vs. spline). As expected, the Plummer simulation is missing power at high k (6% at $k_{\rm Nyquist}$) due to the long tail of the Plummer force softening law.

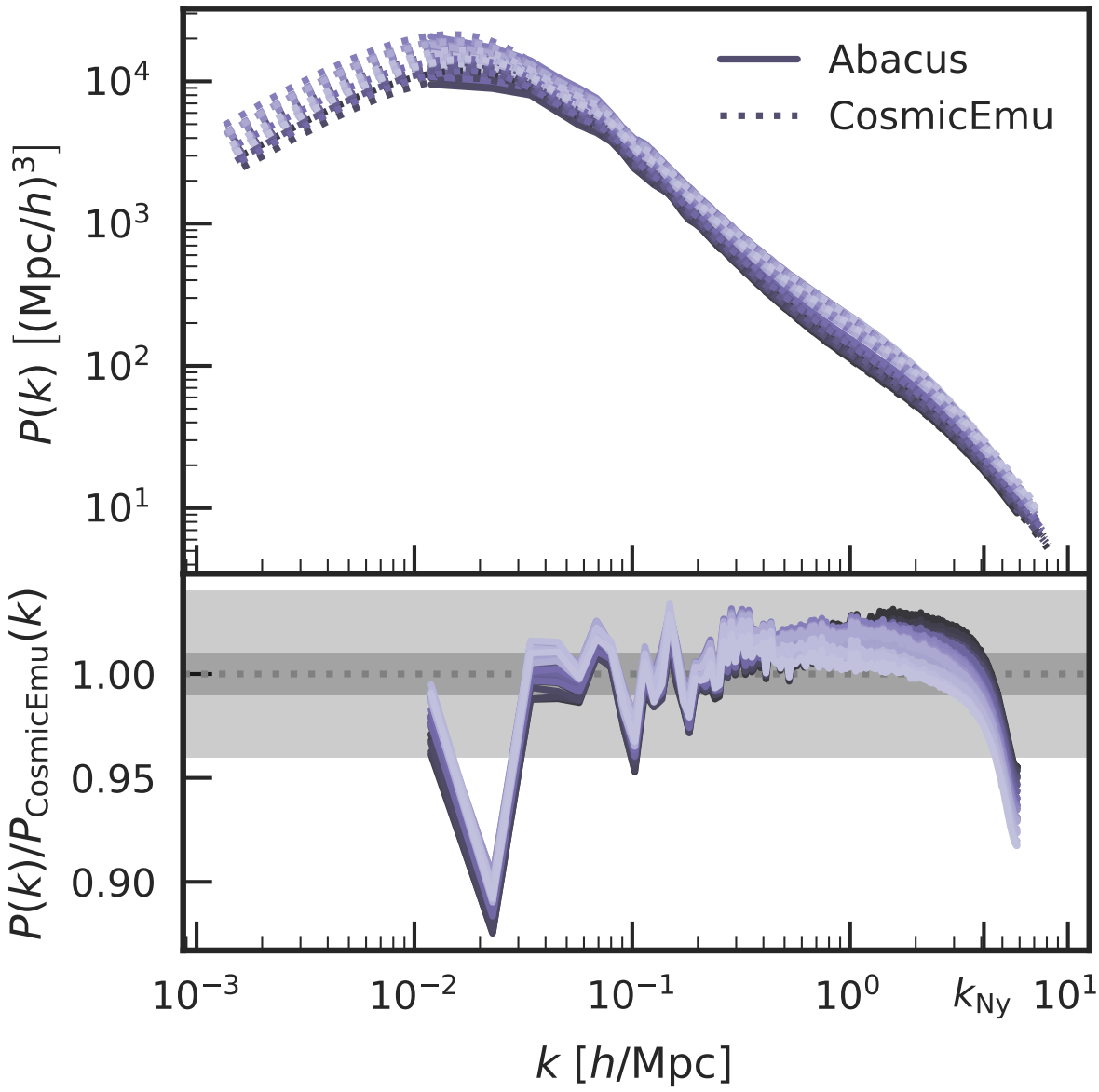

Figure 5.4: Comparison of the $z = 0.5$ power spectra from the `AbacusCosmos_1100box` simulations to the CosmicEmu power spectrum emulator (Lawrence et al. 2017). The comparison is shown for the 28 cosmologies that fall within the CosmicEmu domain. The shaded bar shows the 1% error region. Each line represents a simulation; the colors have no meaning beyond distinguishing the lines. The overall agreement is very good; the low-k differences are due to cosmic variance, while the high-k differences are due to the finite resolution of the simulations.

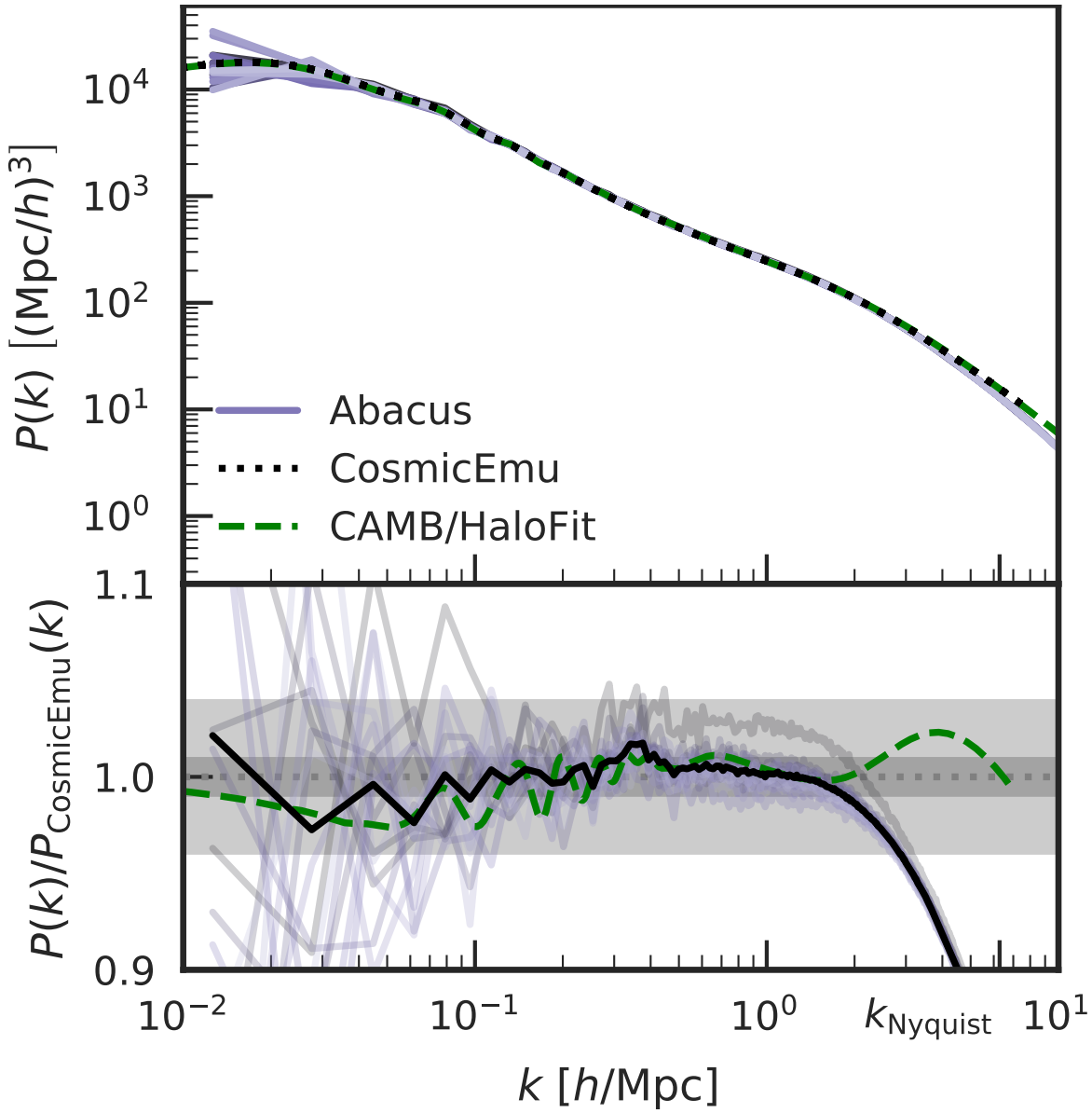

Figure 5.5: Comparison of the $z = 0.3$ power spectra from the `emulator_720box_planck` simulations to the CosmicEmu (Lawrence et al. 2017) and HaloFit (Takahashi et al. 2012) power spectrum emulators. The inner shaded bar shows 1% agreement, while the outer bar shows the 4% accuracy quoted by CosmicEmu. Each Abacus line represents a simulation, all of which have the same cosmology but different initial condition phases. The solid black line is the average of the Abacus lines. The overall agreement is very good among Abacus, CosmicEmu, and HaloFit. The low-k scatter due to cosmic variance is suppressed when averaging over multiple Abacus simulations, while the high-k differences are due to the finite resolution of the simulations and Plummer softening.

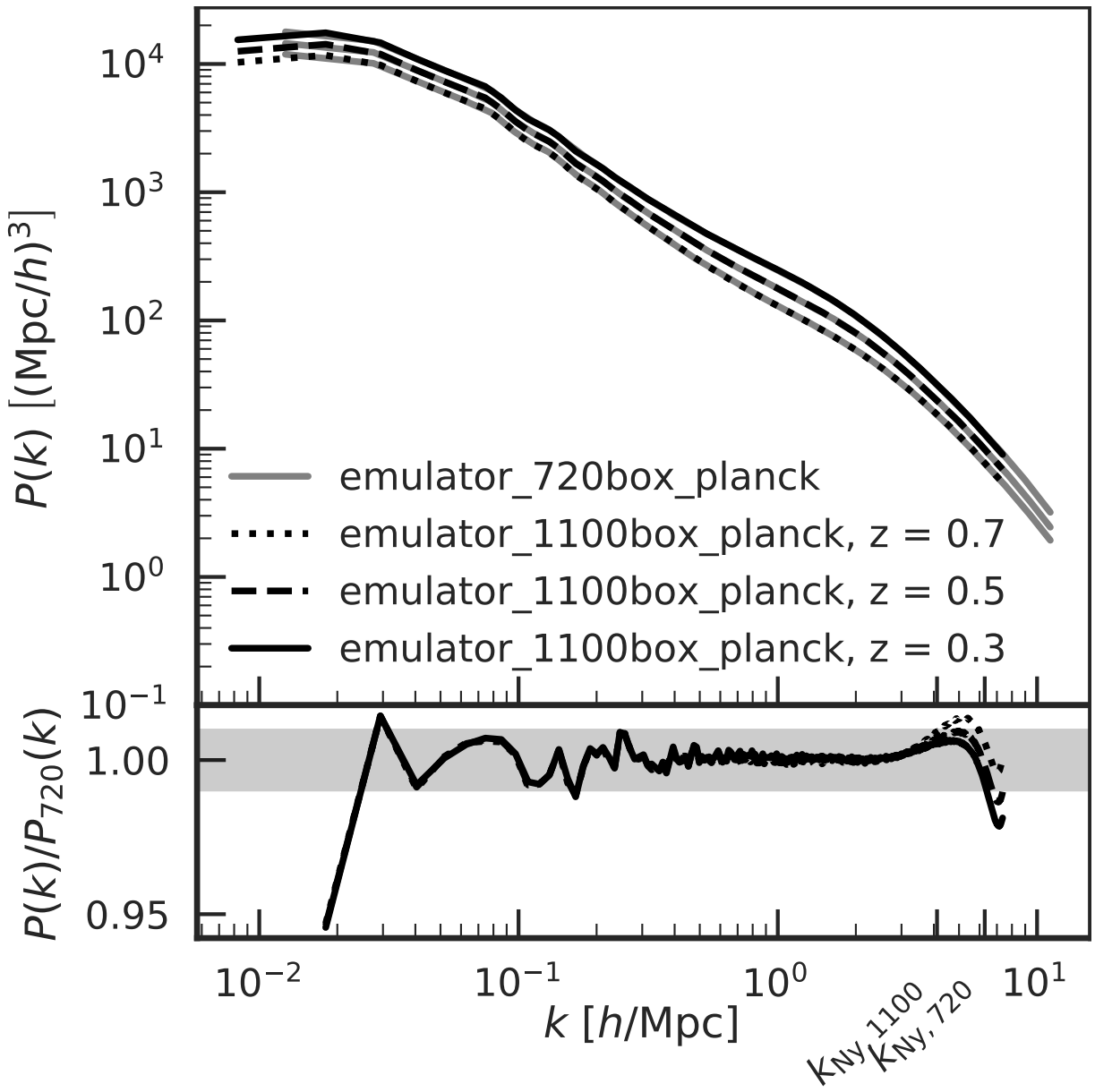

Figure 5.6: A comparison of the matter power spectrum at the two mass resolutions/box sizes. In the intermediate k regime well-sampled by both resolutions, we expect the results be converged, since we are averaging over 17 boxes to suppress sample variance. Indeed, we find excellent agreement of $\sim 1\%$ over a wide range of k. The agreement changes by less than a percentage point from $z = 0.7$ to 0.3.

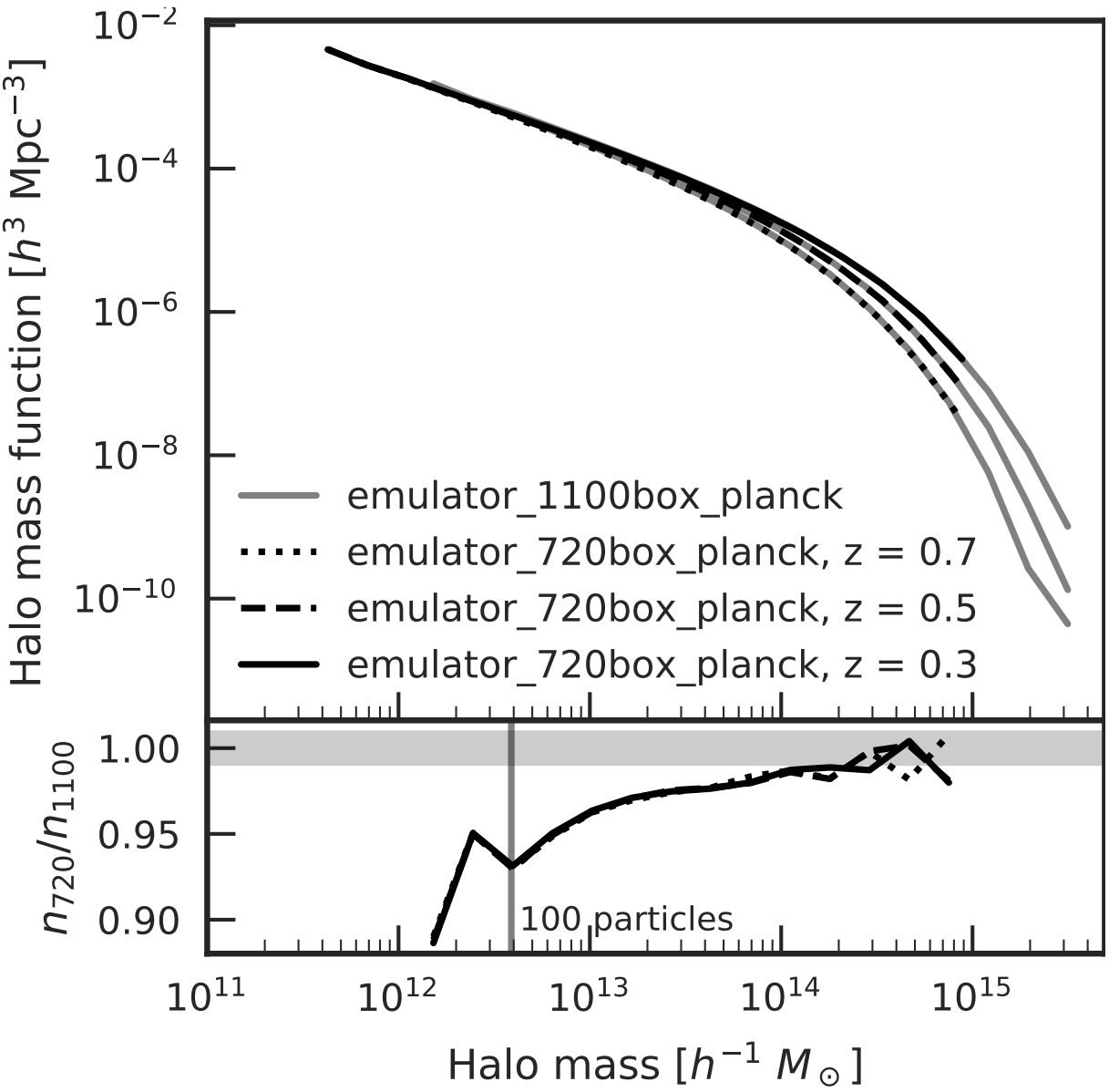

Figure 5.7: A comparison of the FoF halo mass function at the two mass resolutions/box sizes. The lower mass resolution box over-predicts the number of 100-particle halos by 5% compared to 400-particle halos at the higher resolution. This is an effect of friends-of-friends and is not seen with ROCKSTAR; see §5.7.2.

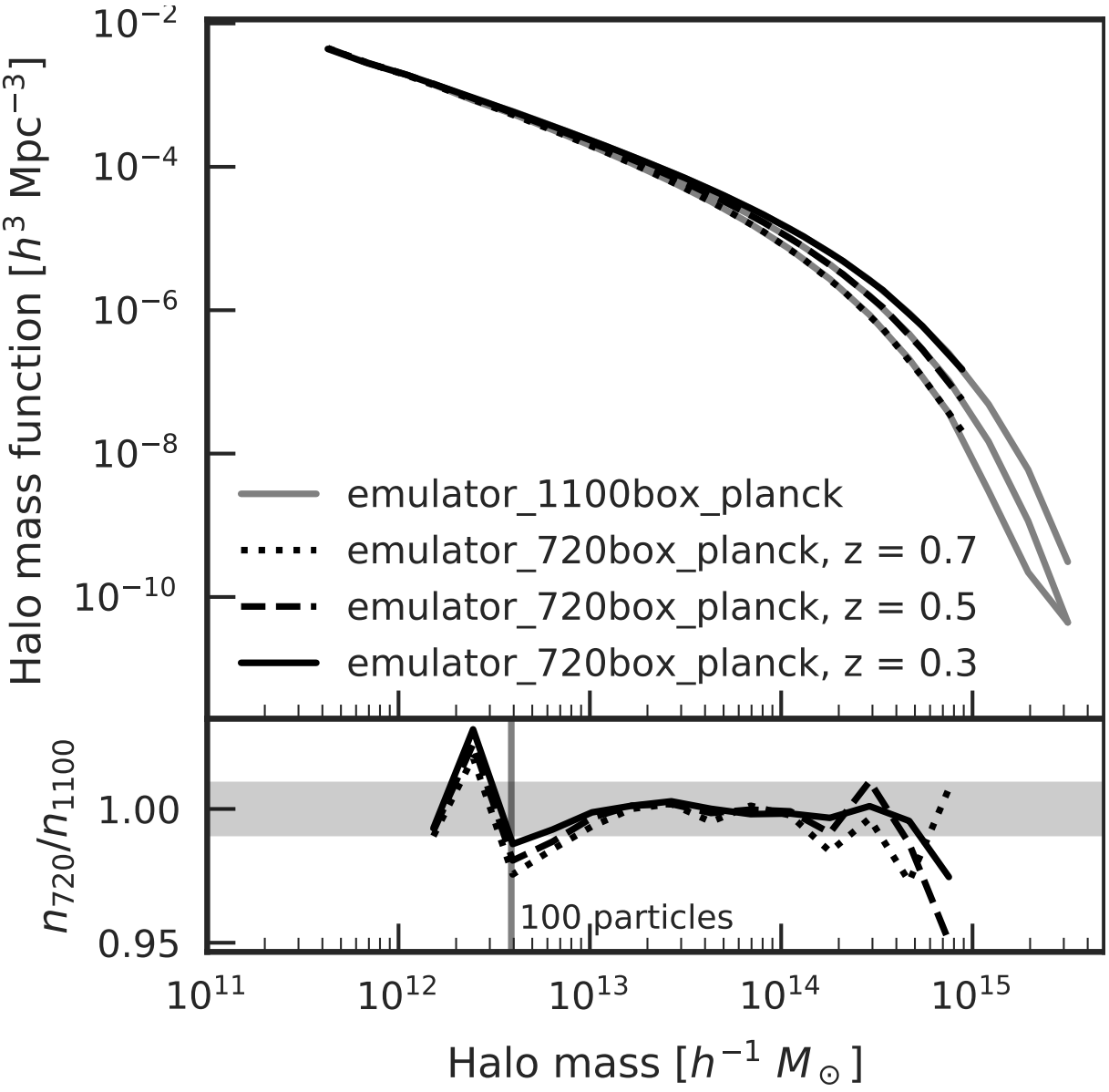

Figure 5.8: Same as Fig. 5.7, but for ROCKSTAR halos.

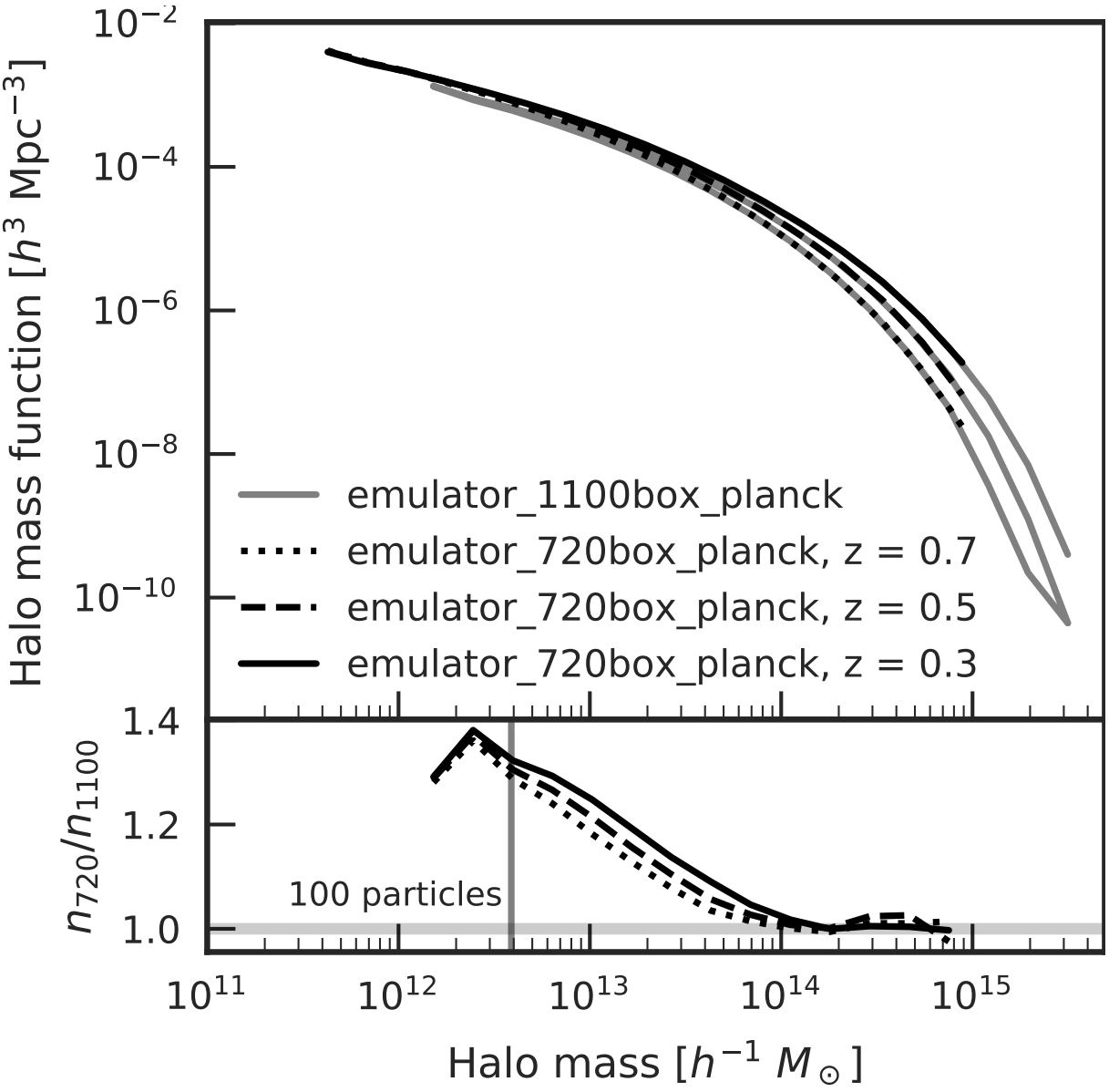

Figure 5.9: Same as Fig. 5.7, but for ROCKSTAR halos with spherical overdensity masses.

Chapter 6

Generating Approximate Halo Catalogs for Blind Challenges in Precision Cosmology

This thesis chapter was originally published as

Garrison, Lehman H., Daniel J. Eisenstein *2019, MNRAS, 485, 2407*

Abstract

We present a method for generating suites of dark-matter halo catalogs with only a few N-body simulations, focusing on making small changes to the underlying cosmology of a simulation with high precision. In the context of blind challenges, this allows us to reuse a simulation by giving it a new cosmology after the original cosmology is revealed. Starting with full N-body realizations of an original cosmology and a target cosmology, we

fit a transfer function that displaces halos in the original so that the galaxy/HOD power spectrum matches that of the target cosmology. This measured transfer function can then be applied to a new realization of the original cosmology to create a new realization of the target cosmology. For a 1% change in σ_8, we achieve 0.1% accuracy to $k = 1h\,\mathrm{Mpc}^{-1}$ in the real-space power spectrum; this degrades to 0.3% when the transfer function is applied to a new realization. We achieve similar accuracy in the redshift-space monopole and quadrupole. In all cases, the result is better than the sample variance of our $1.1h^{-1}$ Gpc simulation boxes.

6.1 Introduction

Cosmological N-body simulations are a computationally expensive but important tool for forward-modeling a cosmological model to an observed distribution of galaxies. Analytic methods cannot yet reproduce small-scale features in the galaxy field to the precision that observations provide, but N-body simulations are too expensive to densely sample the allowed cosmological parameter space. Thus, much attention has turned to semi-numerical methods based on sparsely sampling the parameter space with N-body simulations. Such techniques include "emulation", or interpolation, and "warping", or modifying the cosmology of a simulation output (e.g. halo catalog).

Emulation typically focuses on certain key one-dimensional statistics like the power spectrum (e.g. Heitmann et al. 2016), 2PCF (e.g. Zhai et al. 2018), and halo mass function (e.g. McClintock et al. 2018). This allows for fast evaluation in the space of cosmological parameters and possibly galaxy bias parameters, but analysis is limited to those statistics and to the prescribed galaxy bias models. Warping, on the other hand, is typically slower

but produces a full simulation output (e.g. Angulo & White 2010) or halo catalog (e.g. Mead & Peacock 2014) to which any catalog-based analysis may be applied. This is our approach in this work.

Our motivation is to develop a warping technique suitable for use in blind mock challenges. This is an analysis verification methodology in which an analyzer is given a mock galaxy catalog and asked to infer the underlying cosmology without knowing the true values. This mimics real data analysis and helps avoid human bias in tuning of fitting parameters that could cause underestimation of the systematic error budget of the survey. A fast warping technique is desirable for blind challenges since it allows a simulation to be used more than once. Normally, a simulation can never be reused once its cosmology is revealed, but an accurate warping technique effectively re-blinds the simulation by changing the underlying cosmology. This means that simulations can be reused between blind challenge epochs, so computational effort can be spent on making a few large-volume, high-quality catalogs instead of suites of single-use simulations.

This blind-challenge context has implications for what kinds of rescaling techniques are allowable. The analyzer should not be given any hints as to the direction of the change in cosmology, so changing the box size (e.g. to hold the non-linear mass scale fixed) is not allowed. Also, the precision of the mocks must be high enough that errors of the size of the analysis error budget can be identified, which itself is a fraction of the total survey error budget.

In this work, we present a new warping technique focusing on high precision ($\sim$0.1%) for small changes in cosmology ($\sim$1%), rather than rough precision for a large change in cosmology. A 1% change is sufficient to re-blind a catalog at the expected level of

precision of analysis of a upcoming galaxy surveys.

To achieve this level of precision, our warping technique requires a full N-body realization of the target cosmology. This one realization can be leveraged into many warped catalogs using more realizations of the original cosmology, as we will show. Having a realization of the target cosmology is a requirement not all warping methods share, but it allows us to generate mocks of high precision. Future extensions to this work will relax this requirement.

6.2 Warping

6.2.1 Outline

Our warping procedure starts with two halo catalogs from phase-matched simulations with slightly different cosmologies: the "original" and the "target". The catalogs include particle subsamples. The goal is to modify the clustering of the original catalog to match that of the target. The halo clustering does not simply following the mass clustering—we find the halo bias changes with cosmology at a non-negligible level—so we adjust the halo clustering directly by displacing halos and halo particles (see Section 6.A for why we use displacements instead of re-labeling of halo mass). The warping procedure consists of four main steps:

1. displace halos by the difference of the 2LPT initial conditions in the two simulations, scaled to the catalog redshift;

2. rescale halo radii (and other properties) as a function of abundance;

3. iteratively fit a transfer function that displaces the original halos to match the target halo power spectrum;

4. fit a second transfer function that adjusts the velocities of the original halos to match the target redshift-space halo power spectrum multipoles.

Each of these steps will be explained in detail in the following sub-sections.

Although the resulting transfer functions are fit to a particular pair of phase-matched simulations, they are applicable to a new realization of the original cosmology (as we show in Section 6.3.4). This is how we generate many warped catalogs from a single realization of the target cosmology. With our Abacus Cosmos suite of simulations (Garrison et al. 2018), for example, we have 20 realizations of a fiducial Planck cosmology and 40 with different wCDM cosmologies but matched phases, so this technique can generate 800 mock catalogs from 40 transfer function measurements.

6.2.2 Initial Condition Residuals

As with most warping procedures (e.g. Angulo & White 2010), our first step is to replace the original large-scale simulation modes with those of the target cosmology. We generate initial conditions (ICs) at z_{init} in the Zel'dovich Approximation (ZA, Zel'dovich 1970) for both cosmologies using the `zeldovich-PLT` code of Garrison et al. (2016) which includes particle discreteness corrections. We label these comoving displacement fields $\mathbf{q}_1(\mathbf{x}_i)$ and $\mathbf{q}'_1(\mathbf{x}_i)$, where the prime indicates a quantity in the target cosmology, $\mathbf{x}_i$ is the Lagrangian coordinate of particle i, and $\mathbf{q}_1$ is the first-order (ZA) displacement. The associated velocities are labeled by $\mathbf{v}_1$ (we will leave the argument $\mathbf{x}_i$ implicit in the following). We

also generate the second-order Lagrangian Perturbation Theory displacements (2LPT, labeled $\mathbf{q}_2$) with the ABACUS configuration-space method (Garrison et al. 2016), which uses two force evaluations and reversal of particle displacements. We store the second-order part separately from the first-order part so we can apply the correct redshift and cosmology scaling for each.

These scalings are as follows:

$$\mathbf{q}_1(z) = \left[\frac{D(z)}{D(z_{\rm init})}\right] \mathbf{q}_1, \tag{6.1}$$

$$\mathbf{q}_2(z) = \left[\frac{D(z)}{D(z_{\rm init})}\right]^2 \left[\frac{\Omega_m(z)}{\Omega_m(z_{\rm init})}\right]^{-1/143} \mathbf{q}_2, \tag{6.2}$$

where z is the catalog redshift and $D(z)$ is the linear growth factor, which ABACUS computes via direct integration of the linear growth equation. ABACUS does not compute second-order growth factors, so in Eq. 6.2 we instead use the scalings of Bernardeau et al. (2002) for the Ω_m dependence in a flat cosmology. When computing the redshift dependence of primed quantities, we use the target cosmology.

The sum of the 1st and 2nd order residual differences between the cosmologies forms the total residual:

$$\Delta\mathbf{q} = (\mathbf{q}_1'(z) - \mathbf{q}_1(z)) + (\mathbf{q}_2'(z) - \mathbf{q}_2(z)). \tag{6.3}$$

The velocity scaling is similar but carries a dependence on the growth rate f:

$$\mathbf{v}_1(z) = \left[\frac{D(z)}{D(z_{\rm init})}\right] \left[\frac{f(z)}{f(z_{\rm init})}\right] \mathbf{v}_1, \tag{6.4}$$

$$\mathbf{v}_2(z) = \left[\frac{D(z)}{D(z_{\rm init})}\right]^2 \left[\frac{\Omega_m(z)}{\Omega_m(z_{\rm init})}\right]^{-1/143} \left[\frac{\Omega_m(z)}{\Omega_m(z_{\rm init})}\right]^{6/11} \mathbf{v}_2, \tag{6.5}$$

$$\Delta\mathbf{v} = (\mathbf{v}_1'(z) - \mathbf{v}_1(z)) + (\mathbf{v}_2'(z) - \mathbf{v}_2(z)). \tag{6.6}$$

Again, ABACUS computes the linear growth rate f but not the corresponding second order

quantity, so we use the known Ω_m scaling in Eq. 6.5. We store velocities as comoving redshift-space displacements (that is, the same units as the positions), so we avoid any explicit cosmology dependence via $H(z)$.

We apply these position and velocity residuals to halo h by taking the mean residual over the halo subsample particles:

$$\Delta\bar{\mathbf{q}}_h = \frac{1}{N_h^{\rm ss}} \sum_{i \in h} \Delta\mathbf{q}(\mathbf{x}_i), \tag{6.7}$$

and likewise for the velocities. $N_h^{\rm ss}$ is the number of subsample particles in halo h; we typically use 10% of all halo particles. The subsample particles contain an ID number that encodes the particle's Lagrangian coordinate $\mathbf{x}_i$, and thus the residual displacement $\Delta\mathbf{q}(\mathbf{x}_i)$ can be found. This averaging process samples the initial Lagrangian patch from which the halo forms and effectively smooths the displacement field. We have also tried introducing an explicit smoothing scale, but it has very little effect on the final result; any difference gets absorbed by the transfer function (Section 6.2.4).

We apply the comoving halo displacement $\Delta\bar{\mathbf{q}}_h$ to the halo center and to all its subsample particles. The particles are simply "advected" along with the halo center. We will consider internal halo structure changes in the next section.

In addition to the 10% halo particle subsample, we also have a 10% sample of all particles at the catalog redshift z which is outputted during halo finding. We will use this as a uniform sampling of the late-time matter density field in a later step. To keep this field consistent with the displaced halo field, we save the initial condition residuals for these particles.

6.2.3 Halo Property Rescaling

We will now consider a basic prescription for rescaling internal halo structure by matching halo population statistics between cosmologies. First, we must select the populations to match. One choice is to rank halos by mass in one cosmology and make a mass cut, then rank halos in the other cosmology by mass and make a number-density cut so the two catalogs have the same galaxy density. In this work, we use a mass cut of 100 particles in the original cosmology, chosen so that every halo has at least a few subsample particles (from which the IC displacements were computed in the previous section). The abundance match cutoff will likely fall in the middle of a mass bin in the target cosmology—that is, many halos will have exactly 100 particles—so there is no clear abundance ordering in the last bin. We select a random sample of halos in this bin mainly to avoid any pathologies from correlations between catalog order and spatial order.

To rescale halo radii, we bin the halos by abundance, compute the median halo radius in each bin, and take ratio between cosmologies. For small changes in cosmology, we observe that this ratio changes monotonically with log-abundance and can be fit with a line or low-order function. We then adjust particles radially in each halo according to the ratio from this fit. The exact radius definition we use depends on the halo finder. For the simple friends-of-friends halos (Davis et al. 1985) considered here, we use r_{50}, or the 50th percentile of the radial particle distribution.

When considering redshift-space distortions, we apply this same rescaling procedure to the halo velocity dispersion σ_v. Future extensions will consider more complicated changes to halo structure, such as changes to concentration.

6.2.4 Transfer Function

The previous two steps of applying IC residuals and rescaling halo properties are designed to increase the agreement between simulations but will not achieve our target of 0.1% precision. On large scales, small cosmological parameter changes cause the halo bias to shift even for abundance-selected halo samples (Figure 6.1). On smaller scales, the step of applying the IC residuals tilts the power spectrum due to the effective smoothing imposed by averaging over subsample particles. A power spectrum fitting routine would likely interpret this tilt as a change in cosmology.

We would like to force the clustering to match; we do so in Fourier space by moving halos according to a transfer function $T(k)$. The transfer function operates isotropically on a "late-time displacement field": we compute the gradient of the gravitational potential of the late-time matter field and treat the resulting vector field as displacements. This is the same idea as the Zel'dovich Approximation for initial conditions, except we are operating on the late-time matter density. The idea is that these late-time displacements will trace bulk flows of the matter and that by pushing the halos along these flows we can increase or decrease the clustering of the catalog. To allow for changes in the shape of the power spectrum, not just the mean amplitude, we modulate the late-time displacement field by an isotropic function $T(k)$ instead of a scalar.

Specifically, the late-time displacement field $\mathbf{s}(\mathbf{x})$ is computed as

$$\mathbf{s}(\mathbf{x}) = \mathscr{F}^{-1}\left[\tilde{\mathbf{s}}(\mathbf{k})\right] \tag{6.8}$$

$$\tilde{\mathbf{s}}(\mathbf{k}) = \begin{cases} ik^{-2}T(k)\tilde{\delta}_m(\mathbf{k})\mathbf{k}, & k \leq k_{\max}; \\ \mathbf{0}, & k > k_{\max}, \end{cases} \tag{6.9}$$

where $\tilde{\delta}_m(\mathbf{k})$ is the Fourier transform of the matter density field from subsample particles at the catalog redshift. We apply the initial condition residuals $\Delta\mathbf{q}$ to these particles before computing the density field. The inverse Fourier transform is indicated by $\mathscr{F}^{-1}$.

Since $\tilde{\mathbf{s}}(\mathbf{k})$ is computed with an FFT, $\mathbf{s}(\mathbf{x})$ exists on a lattice. We evaluate a halo's displacement using tri-linear interpolation from the lattice to the halo center. As with the IC residuals, the halo particles are advected along with the halo center.

We now discuss how we use optimization to find $T(k)$. We parametrize $T(k)$ as $N_{\rm bin}$ discrete segments linearly spaced from the fundamental mode $k_{\rm fund}$ to $k_{\rm max}$, the latter of which is set by our desired analysis range. We label this discretized transfer function $T(k_i)$. Our target clustering metric is a pseudo-HOD power spectrum computed by giving every halo center a weight of 1 and every halo subsample particle a weight of $w_s = 0.007$; this choice yields a satellite fraction of about 25%. Using every subsample particle as a "fractional satellite" has the effect of reducing the shot noise relative to a single HOD realization. We compute the resulting overdensity field with TSC mass assignment and deconvolve the window function upon computing the power (Jing 2005). We seek to minimize the difference in this quantity between original and target cosmologies. Specifically, we minimize the difference in the real-space monopole $P_0(k)$:

$$\chi^2 = \sum_{k_i}^{k_{\rm max}} \frac{[P_0(k_i) - P_0'(k_i)]^2}{2\alpha\sigma_{P0}'^2(k_i)}, \tag{6.10}$$

$$\sigma_{P0}'^2(k_i) = 2P_0'^2(k_i), \tag{6.11}$$

where we take $P_0(k_i)$ to be the monopole power in bin i and $\sigma_{P0}'^2(k_i)$ to be the monopole variance in that bin. As before, a prime indicates a quantity from the target simulation. We drop $N_{\rm modes}^{-1}$ from the formal sample variance definition since these are phase-matched simulations, and also we only care about the monopole variance relative to the quadrupole

variance (during fitting of the redshift-space velocity transfer function; see below). α is a numerical "fudge factor" to rescale χ^2 to a convenient range for convergence testing; we typically use $\alpha = 10^{-6}$.

The process of drifting the halos and applying TSC mass assignment is non-linear, so we use a non-linear numerical optimizer to minimize χ^2 with respect to $T(k_i)$. For $k_{\rm max} = 1h\,{\rm Mpc}^{-1}$ and a 512^3 FFT mesh, we typically use $N_{\rm bin} = 10$, so this is a 10-dimensional optimization problem. We have tried Powell's method (Powell 1964) and Nelder-Mead (Nelder & Mead 1965) from the SciPy package (Jones et al. 2001–); both work well. For a trial transfer function $\hat{T}(k_i)$, a single χ^2 evaluation consists of the following steps.

1. Apply $\hat{T}(k_i)$ to the late-time displacements $\tilde{\mathbf{s}}(\mathbf{k})$ (Eq. 6.9); call this field $\hat{\tilde{\mathbf{s}}}(\mathbf{k})$.
2. Take the inverse FFT:
$$\hat{\mathbf{s}}(\mathbf{x}) = \mathscr{F}^{-1}[\hat{\tilde{\mathbf{s}}}(\mathbf{k})]. \tag{6.12}$$
3. Interpolate the displacements $\hat{\mathbf{s}}(\mathbf{x})$ to halo centers using tri-linear interpolation.
4. Apply the halo center displacements to halos and halo subsample particles.
5. Compute the resulting power spectrum monopole $P_0(k_i)$.
6. Compute χ^2 (Eq. 6.10).

Note that $\tilde{\mathbf{s}}(\mathbf{k})$ can be precomputed from the matter density field and is not updated during iteration.

Executing the above 6 steps takes about 3 seconds with 1.5×10^6 halos and 5.5×10^7 halo particles on a 24-core machine. Powell's method uses about 6 steps and 1400 χ^2

evaluations for a total optimization time of 1.2 hours. The resulting P_0 matches P_0' to 0.1% down to $k_{\rm max}$ (except for the sample-variance-dominated large scales); see Section 6.3.

The choice of $k_{\rm max}$ is set by our science goals: we want to modify the power spectrum over the range that we will reasonably analyze. Another consideration is the power spectrum mesh size: we want $k_{\rm max}$ to be somewhat smaller than $k_{\rm Nyquist}$ of the mesh to avoid the worst of the aliasing effects, but a larger mesh slows requires more memory and makes the optimization slower.

6.2.5 Redshift Space: Residuals, Velocity Dispersion, and Transfer Function

The discussion thus far has focused on the real-space power spectrum, but we would like to match the redshift-space monopole and quadrupole, too. Thus, we must consider how to warp the velocities. We will apply initial condition residuals, rescale halo velocity dispersion, and apply a transfer function, as we did with the positions. The IC residuals and transfer function modify the two-halo "Kaiser" redshift-space distortions (RSD); the velocity dispersion modifies the one-halo "Finger-of-God" RSD.

Applying the velocities from the IC residuals to the halo centers and particles is straightforward; the prescription is already given in Eqs. 6.4–6.6. We apply the halo center kick identically to all halo particles.

We then rescale halo particle velocities according by matching velocity dispersion σ_v as a function of abundance against the target cosmology. As with the radius rescaling, we find that the ratio of the medians is a slowly changing function of abundance and can

be fit with a line. The particle velocities are simply scaled by this fit to the ratio.

Before starting the velocity warping, we apply one other correction. The late-time displacements $\mathbf{s}(\mathbf{x})$ in Eq. 6.8 imply a unique velocity $\mathbf{w}(\mathbf{x})$ due to their dynamical origin (just as displacements in Zel'dovich Approximation initial conditions have a unique velocity):

$$\mathbf{w}(\mathbf{x}) = \frac{f(z)H(z)}{H_0}\mathbf{s}(\mathbf{x}). \tag{6.13}$$

The $f(z)H(z)$ factor is a statement of redshift z dynamics and gives the comoving velocity at that redshift. The H_0^{-1} factor converts this to the comoving redshift-space displacement for a $z = 0$ observer. We use redshift-space displacement units for all velocities.

The velocity transfer function framework is largely the same as the displacement transfer framework, except that we use the line-of-sight late-time velocities instead of the 3D late-time displacements. The density field from which the late-time velocities are computed and the halo positions to which they are interpolated remain in real-space, but the power spectrum is computed on the redshift-space quantities. We pre-apply the peculiar velocities to the halo particles as redshift-space distortions, so the velocity transfer function just has to drift them by the same amount as the halo center.

The velocity transfer function is still an isotropic monopole, as any velocity modifications must be isotropic. We choose a line of sight for the RSD, however, so we only apply the transfer function to the z velocities for efficiency while fitting. We operate in the flat-sky approximation in a periodic simulation box, but the final warped catalog has the velocity modification applied isotropically, so the redshift space distortions should be accurate in any direction as long as the z axis is not special.

We adopt a χ^2 that includes both the redshift-space monopole and quadrupole:

$$\chi^2 = \sum_{k_i}^{k_{\rm max}} \frac{[P_0(k_i) - P'_0(k_i)]^2}{2\alpha\sigma'^2_{P0}(k_i)} + \frac{[P_2(k_i) - P'_2(k_i)]^2}{2\alpha\sigma'^2_{P2}(k_i)}, \tag{6.14}$$

$$\sigma'^2_{P2}(k_i) = 10P'_0(k_i)^2, \tag{6.15}$$

where we have used $\ell = 2$ in the multipole variance $2(2\ell+1)P'_0(k_i)^2$. As with the monopole variance (Eq. 6.11), we drop the formal $N_{\rm modes}^{-1}$ dependence from the quadrupole variance, since these are phase-matched simulations and the variance only matters relative to the monopole.

The fitting takes about 9 steps with 2300 function evaluations, although it is largely converged in half that number. This takes about 2.3 hours on a 24-core machine.

6.3 Results

6.3.1 Outline

In the following, we test our warping procedure on the `AbacusCosmos_1100box_00-0` and `AbacusCosmos_1100box_01-0` simulations which are available on the AbacusCosmos website[1]. These are two phase-matched simulations with 1440^3 particles in $1100h^{-1}\,$Mpc boxes and are identical except for a 1% change in σ_8—the former (the original) has $\sigma_8 = 0.83$, while the latter (the target) has $\sigma_8 = 0.8383$. We focus on this simple cosmology change for now while we validate the basic premise of our warping. We use friends-of-friends halo catalogs at $z = 0.5$ and 10% halo and field particle subsamples. We use

[1] `https://lgarrison.github.io/AbacusCosmos/`

halos of 100 particles or more in the original cosmology, which corresponds to halo mass $3.7 \times 10^{12}\ h^{-1}\ \mathrm{M}_\odot$. There are 1.5×10^6 of these halos in a $1.3 h^{-3}\ \mathrm{Gpc}^3$ volume, for a halo density of $1.1 \times 10^{-3} h^{-3}\ \mathrm{Mpc}^3$.

First, we examine the effect of the IC residuals, halo radius rescaling, and the transfer function in real space, then in redshift space. Finally, we check that a transfer function measured on one pair of simulations can be applied to another with similar accuracy.

6.3.2 Real Space

Figure 6.1 shows the results of the warping procedure on the real-space power spectrum monopole and cross correlation. First, we note that the halo bias evolves rapidly from the original to target simulations—for a 1% change in σ_8, we should see a 2% change in the halo power spectrum, assuming constant halo bias. (We do indeed measure exactly a 2% change in the large-scale matter clustering.) Instead, we see almost no change; the halo bias has almost exactly canceled the change in σ_8. Thus, when we apply the IC residuals from the large-scale modes, the halo power spectrum overshoots that of the target cosmology.

The bias cancellation does not mean that the catalog can be used as a realization of the new cosmology without any modification, of course. This is seen most immediately in the redshift space $\ell = 0$ and $\ell = 2$ clustering (Figures 6.3 & 6.4) where biases of 0.5–1.0% are observed; our redshift-space warping will successfully deal with those in Section 6.3.3.

Despite the power spectrum overshoot due to applying the IC residuals, we see the small-scale cross correlation actually increases. The cross correlation is already good on all scales—the error is smaller than 1.5% for $k < 1 h\,\mathrm{Mpc}^{-1}$—but this gives us confidence

that the IC residuals are moving halos to more closely lie on top of their counterparts in the target cosmology.

The next step is to rescale halo radii according to r_{50}, the median halo radius, as a function of abundance. The halo r_{50} versus abundance for the two cosmologies is given in Figure 6.2. This also shows the trend line that we fit to the change in median r_{50}; we move halo particles radially according to this fit. The scatter in a given bin is quite large but the ratio of the medians appears robust.

Figure 6.1 shows the effect of the radius rescaling on the power; as expected, the only impact is on small scales. The radius changes are generally quite small (less than 0.5%) which translates into a $< 0.1\%$ shift in power even at our $k_{\rm max} = 1h\,{\rm Mpc}^{-1}$.

The direction of this change is such that agreement with the target simulation actually decreases. This is because most halos in the original cosmology have larger r_{50} than their counterparts in the target cosmology, despite the smaller σ_8. This may be an effect of halos forming earlier and thus having higher concentrations with a higher σ_8. This change is small enough that the cross-correlation is unaffected and the transfer function can easily compensate. We prefer to match the 1-halo abundance statistics and deal with the consequences in the 2-halo clustering rather than neglect the former; otherwise, we are likely to end up with a catalog that looks accurate for this particular halo mass cut/HOD but not any other.

Finally, we fit the real-space transfer function using 6 steps of Powell's method (Section 6.2.4). This brings the monopole power of the warped simulation into 0.1% agreement with the target to $k_{\rm max} = 1h\,{\rm Mpc}^{-1}$; this is more precise than the sample variance of the box across the whole fitted range.

We note that the measured transfer function is a relatively smooth function of k and would thus likely be amenable to parametrization as a low-dimensional function instead of 10 independent segments. This would greatly accelerate convergence of the non-linear fit and probably eliminate the minor "sawtooth" effect observed at high k due to use of a constant value to inside each bin.

We observe a tiny decrease in the cross correlation as a result of the transfer function. Since the two-point clustering is the relevant quantity for most cosmology analysis, this very small loss of cross correlation is relatively unconcerning. This is most likely an effect of halo mergers and splits that are not explicitly modeled here; such a problem would be compounded by our use of FoF halos, which are known to over-merge (e.g. More et al. 2011). We will investigate this in future work with more sophisticated halo catalogs.

6.3.3 Redshift Space

Figures 6.3 & Figure 6.4 show the results of the velocity warping procedure on the redshift-space monopole and quadrupole, respectively. We see in both that the real-space warping restores some power on large scales, but the clustering amplitude is offset on small scales. The first step of the velocity warping, matching σ_v, increases both monopole and quadrupole agreement. In Figure 6.5, we can see that the velocity dispersions are higher in the target cosmology, so boosting the dispersions in the original cosmology will tend to decrease the monopole and increase the quadrupole as galaxies are spread out along the line of sight.

The next step of the redshift-space warping is to fit the velocity transfer function, but unlike in real space, this transfer function must simultaneously satisfy the monopole

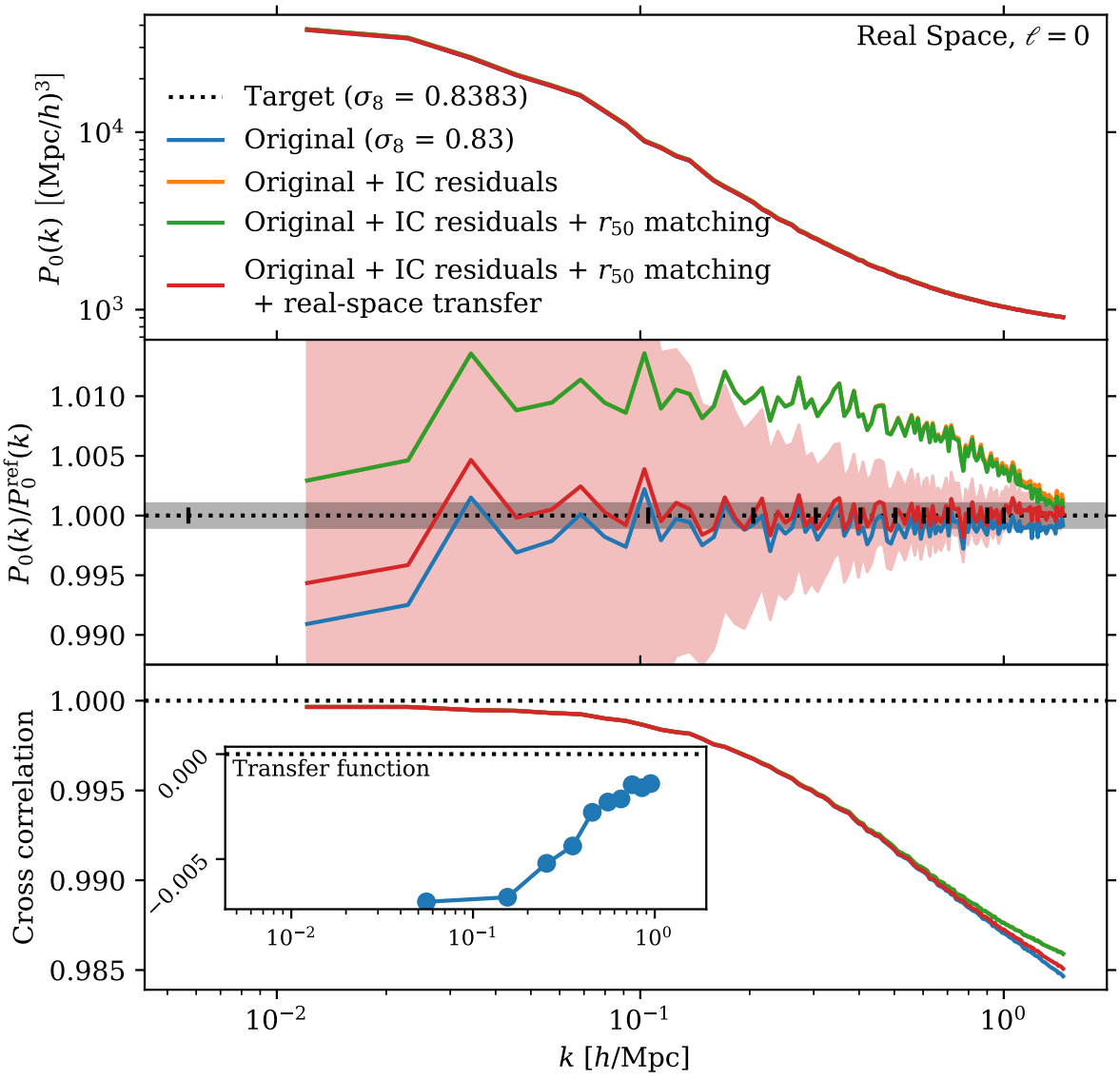

Figure 6.1: The results of the real space warping in the real-space monopole at $z = 0.5$. The dotted lines are from the target simulation; each of the solid lines shows the original simulation during a stage of the warping. Even with no modification, the original catalog matches the target catalog to a striking degree due to the halo bias changing to cancel the change in σ_8. Thus, displacing halos by the difference in the ICs overshoots. The final step of applying the transfer function brings the power into 0.1% agreement with the target even past our chosen $k_{\rm max} = 1h\,{\rm Mpc}^{-1}$. The broad shaded region indicates the sample variance error on the power spectrum; our fitting is consistently better than this variance. The edges of the transfer function bins are marked with vertical ticks in the rectangular shaded region which indicates our target of 0.1% precision. The color scheme is matched to the redshift-space warping plots (Figures 6.3 & 6.4).

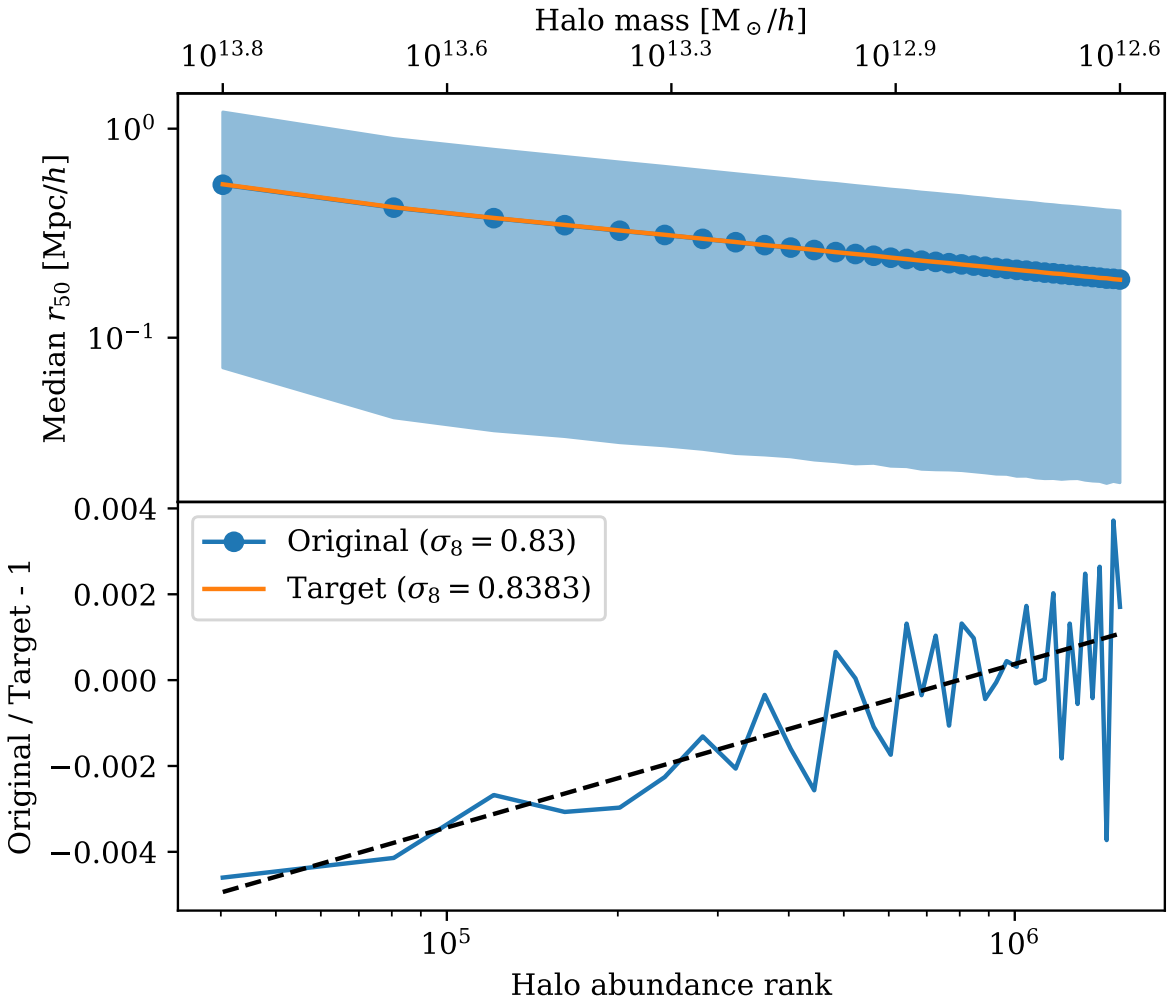

Figure 6.2: *Top panel:* median halo radius r_{50} versus abundance in two cosmologies. *Bottom panel:* the fractional difference in the values in the top panel. The relative radius changes monotonically with mass; we fit a line to the trend (bottom panel, dashed line) and move halo particles radially according to this fit. The fact that this fit passes through zero indicates that massive halos are slightly larger and less massive halos are slightly smaller in the target cosmology. The shaded region in the upper panel indicates the 25th to 75th percentiles of the r_{50} distribution in each bin.

and quadrupole. Even under with this constraint, the transfer function does a very good job: the power matches the target within the sample variance of both multipoles to our chosen $k_{\rm max} = 1.0h\,{\rm Mpc}^{-1}$. The quadrupole is inherently nosier than the monopole, but it is clear that the fit is bringing the simulations into better agreement on all but the smallest scales.

The slight negative offset in the small-scale quadrupole pairs with the slight positive offset in the small-scale monopole. This likely indicates tension between the two, which is unsurprising given that the velocity transfer function strictly operates on two-halo velocities and can make no modifications to internal halo structure. On small scales, the power is dominated by internal halo velocity dispersions, so attempting to reproduce this with bulk halo motions is likely to fail. This is likely the cause of the slight loss of cross-correlation on small scales as well.

6.3.4 Transferring the Transfer Function

The utility of this warping methodology is based on the assumption that a transfer function measured on one pair of phase-matched simulations can be applied to a simulation with different initial phases. We now test that assumption. Figure 6.6 shows the application of the real-space transfer function (Figure 6.1) and redshift-space (velocity) transfer functions (Figures 6.3 & 6.4) to a new simulation. The results are very good: the real space monopole matches the target better than sample variance across the whole k range and better than 0.3% across almost all of the k range. The results are similar in the redshift-space monopole and quadrupole: a mild degradation compared to the original phases but better than sample variance limits to $k_{\rm max}$.

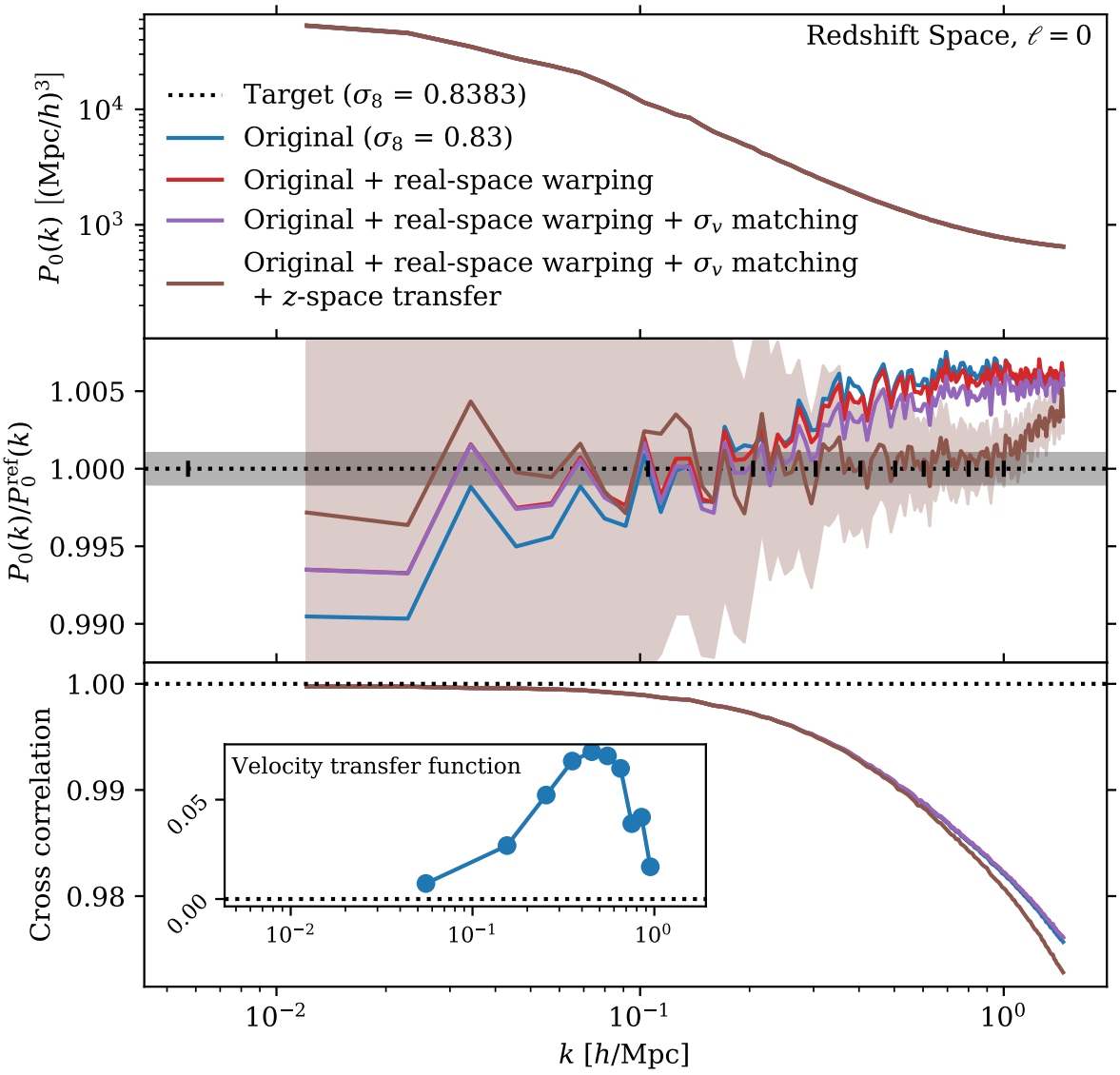

Figure 6.3: The results of the warping on the redshift-space power spectrum monopole. The real-space warping (i.e. the end result of Figure 6.1) helps bring the large-scale power into alignment but fails on small scales. Matching halo velocity dispersions and fitting a velocity transfer function match the power to the target within the sample variance to our chosen $k_{\rm max} = 1 h{\rm Mpc}^{-1}$. See Section 6.3.3. The color scheme is matched to the other warping plots.

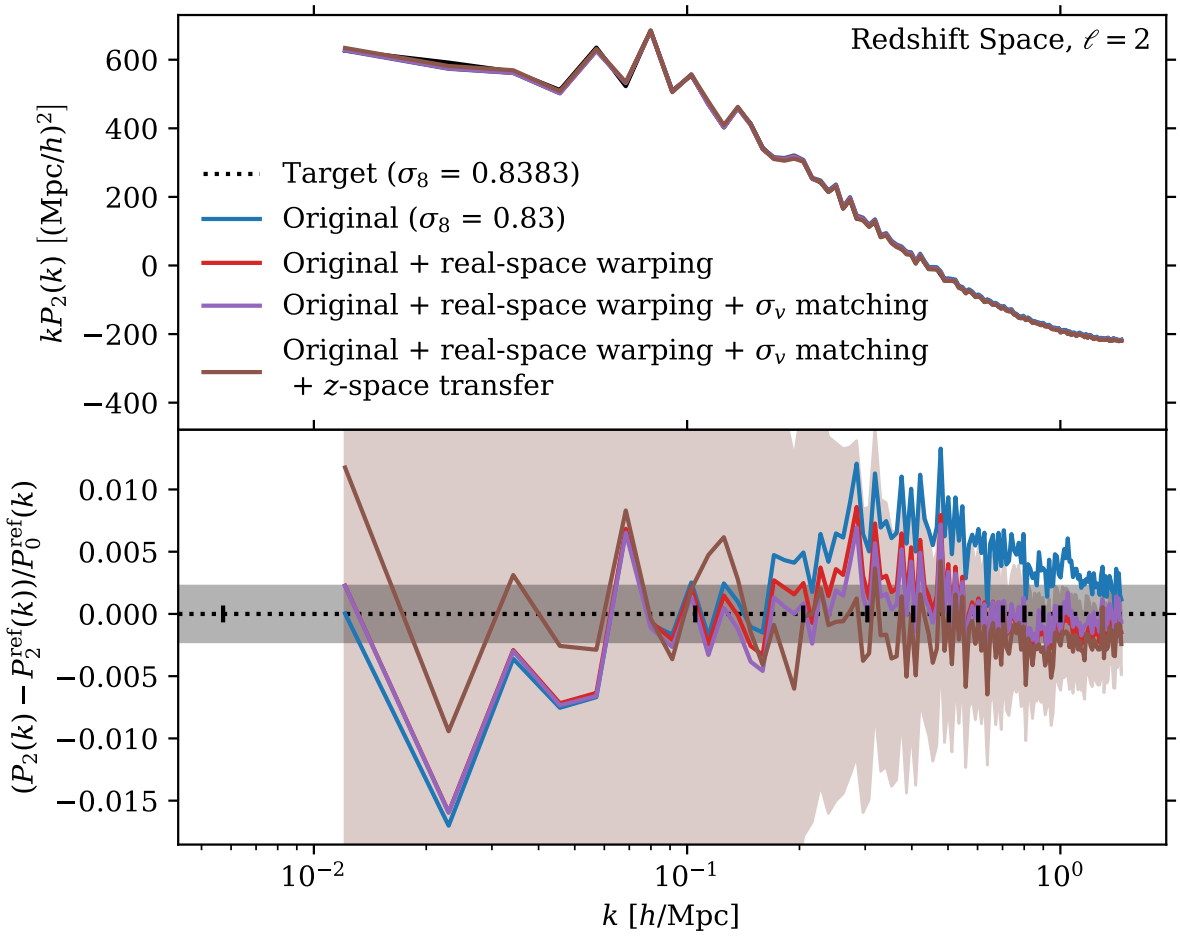

Figure 6.4: The same results for warping in redshift-space as Figure 6.3 but seen in the quadrupole. Note that the second panel is the difference in the quadrupole relative to the monopole, due to the zero crossing of the quadrupole power around $k = 0.2h\text{Mpc}^{-1}$. The horizontal shaded bar thus highlights the $\sqrt{5} \times 0.1\%$ error region, since the quadrupole variance is 5 times that of the monopole. Overall, these results show much the same trends as the redshift-space monopole which are that the real-space warping helps but the σ_v matching and velocity transfer are necessary for precision matching. The color scheme is matched to the other warping plots.

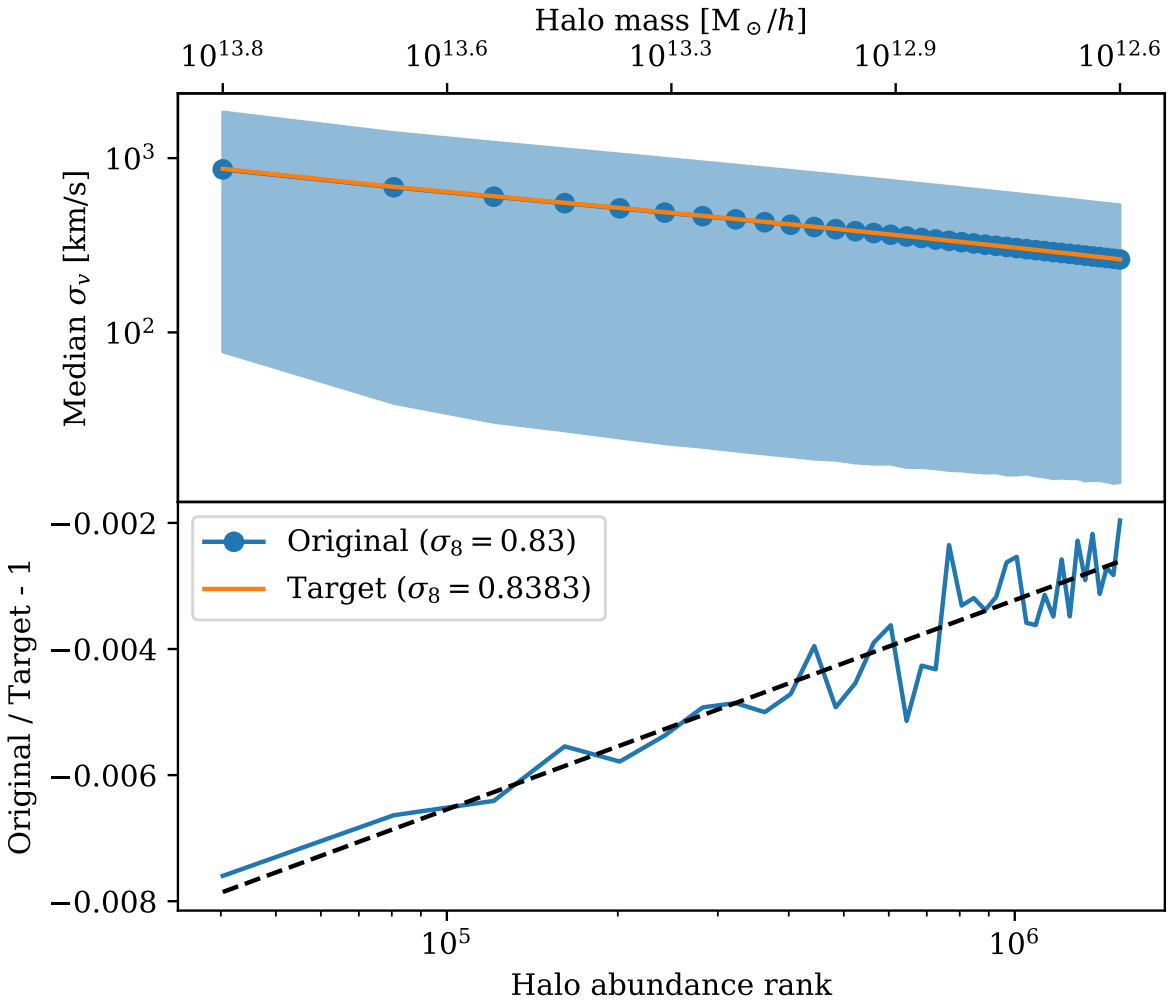

Figure 6.5: *Top panel:* median halo velocity dispersion σ_v versus abundance in two cosmologies. *Bottom panel:* the fractional difference in the values in the top panel. As with the radius (Figure 6.2), the relative σ_v is well fit by a line (bottom panel, dashed). Unlike the radius, however, the sign of the difference is the same across the whole abundance range. Thus, σ_v is always larger in the target cosmology for the mass range examined here. The shaded region in the upper panel indicates the 25th to 75th percentiles of the σ_v distribution in each bin.

The degradation in the real-space monopole appears largely as smooth deflection of peak amplitude $\sim 0.3\%$ from $k = 0.1$–$1 h\,\mathrm{Mpc}^{-1}$. This may be due to different non-linear couplings of the large-scale modes to small-scale structure in this pair of simulations compared with the pair on which the transfer function was measured. At all scales this effect is sub-dominant to sample variance, however.

6.4 Discussion and Future Directions

We have developed a warping framework for changing the cosmology of a simulation, focusing on highly accurate warped catalogs for small changes in cosmology rather than roughly accurate catalogs for large changes in cosmology. We have used a halo displacement technique to modulate the clustering amplitude and thus avoided difficulties associated with an Eulerian re-weighting scheme (Section 6.A). We have shown that the real-space power spectrum monopole can be controlled to 0.1% precision, or 0.3% when measured on one pair of simulations and applied to another. The redshift space monopole and quadrupole are noisier and but can still be controlled to about 0.3%. In all cases, this error is better than the sample variance limit for a $1.1 h^{-1}\,\mathrm{Gpc}$ box.

This work has focused on a simple change in cosmology (a 1% change in σ_8) whose effects on the power spectrum are readily understood. Even in this simple case, we found rapidly evolving halo bias, hence the need for a directly-fit transfer function rather than one motivated from initial conditions or first principles. The next step of this work will be to test the effect of more complicated cosmology changes that also modulate the shape of the power spectrum.

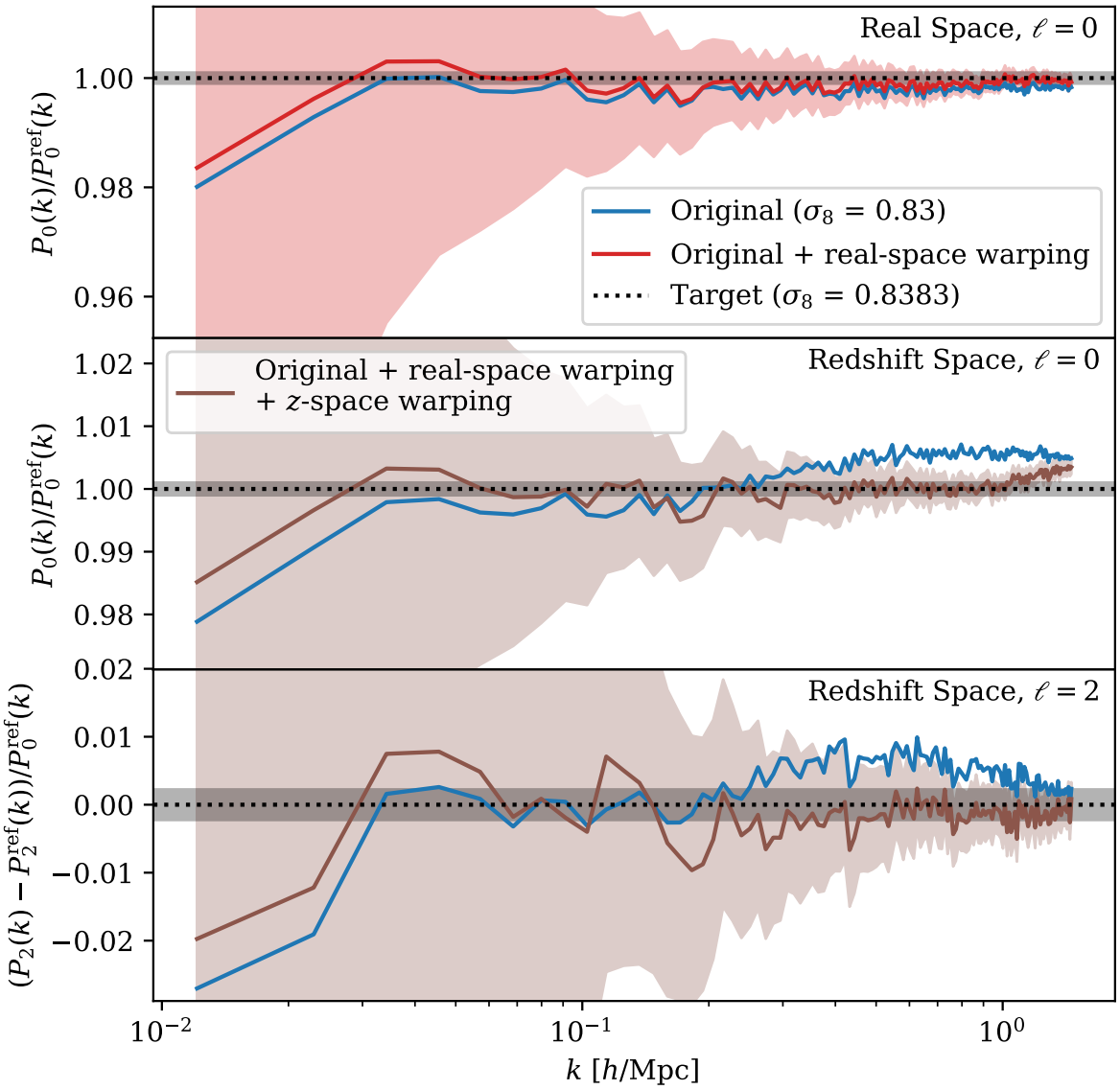

Figure 6.6: A summary of the results of taking the real-space and velocity transfer functions measured on one pair of simulations and applying them to another pair with different initial phases. The top panel shows the ratio of the real-space monopole with the target; the warped result is better than sample variance across the whole k range. The middle panel shows the redshift-space monopole, which is similarly accurate up to our chosen $k_{\rm max} = 1h\,{\rm Mpc}^{-1}$. The bottom panel shows the difference of the redshift-space quadrupole relative to the monopole (due to the zero-crossing in the quadrupole; see Figure 6.4). The quadrupole is noiser but matches better than sample variance across the whole k range. The horizontal shaded bar in the first two panels shows our target of 0.1%; in the last panel, this is $\sqrt{5} \times 0.1\%$ since it is relative to the monopole. The real-space transfer function measurement is shown in Figure 6.1; the redshift-space (velocity) function measurement is shown in Figures 6.3 & 6.4.

The parametrization of $T(k)$ as 10 independent bins was chosen to allow full flexibility in the shape of the transfer function. The measured transfer functions in this work were generally smooth which should admit lower-dimensional parameterizations; this is a natural next step for this work. This will significantly accelerate the optimization step. However, we expect that more complicated changes to the cosmology will cause shape variations in the transfer function, so we must ensure that any simpler parametrization can capture these as well.

We used friends-of-friends halos here for their simplicity, but FoF has well-known pathologies that may be causing some mild difficulties in the halo property rescaling and small-scale transfer function. We expect this method will be applicable to other halo finders, such as ROCKSTAR (Behroozi et al. 2013) or spherical overdensity, and will test this in future work.

Finally, we used a pseudo-HOD prescription when assigning galaxies to halo particles in order to suppress shot noise, but we must test that the warped catalog behaves the same as the target catalog for a range of real HODs and HOD parameters.

We believe this technique will be suitable for applications that require high-quality mocks for small variations in cosmology, such as blind mock challenges. Once we validate our method on more complicated cosmology changes, we will be able to produce suites of simulations suitable for warping. The design of such a suite might be similar to that of our Abacus Cosmos simulations, with 20 realizations of a fiducial cosmology and 40 phase-matched simulations with different cosmologies. Such a design would allow us produce 800 mock catalogs from 40 transfer function measurements. Furthermore, it may be possible to interpolate the transfer functions between cosmologies. This will maximize the utility

of simulations in blind challenges, as we will be able to re-blind the same simulation many times.

Acknowledgements

LHG would like to thank Harshil Kamdar for work on an earlier iteration of this project. We would also like to thank Doug Ferrer, Phil Pinto, and Marc Metchnik as co-authors of Abacus, and Nina Maksimova, Duan Yutong, and Sownak Bose for helpful discussions. This work has been supported by grant AST-1313285 from the National Science Foundation and by grant DE-SC0013718 from the U.S. Department of Energy. DJE is further supported as a Simons Foundation investigator.

6.A Eulerian Transfer: Failed Warping Procedure

Rather than a non-linear fit to a transfer function on the late-time displacements, we initially tried to fit a transfer function directly on the Eulerian pseudo-HOD density field. This approach had the advantage of being linear and solvable via a least-squares minimization. The goal was to find the best-fit $T(k)$ such that

$$\tilde{\delta}'(\mathbf{k}) = T(k)\tilde{\delta}(\mathbf{k}). \tag{6.A}$$

As before, the prime indicates a quantity in the target cosmology. This can be posed as a complex least-squares minimization of

$$\sum_i |\tilde{\delta}'_i - T(k)\tilde{\delta}_i|^2 \tag{6.B}$$

in each bin of k. Imposing the constraint that $T(k)$ must be real gives the solution

$$\hat{T}(k) = \frac{\sum_i \mathrm{Re}(\tilde{\delta_i'}^* \tilde{\delta}_i)}{\sum_i |\tilde{\delta}_i|^2}, \tag{6.C}$$

where the asterisk denotes complex conjugation. The halos (or the halo occupation statistics) would then be re-weighted by this transfer function to reproduce the target density field by construction.

This had two problems. The first was that any roll-off in cross-correlation towards high k would manifest as a suppressed transfer function, as evidenced by the fact that $\hat{T}(k)$ can be rewritten as the product of two terms, the first of which is the cross-correlation:

$$\hat{T}(k) = \left[\frac{\sum_i \mathrm{Re}(\tilde{\delta_i'}^* \tilde{\delta}_i)}{\sum_i |\tilde{\delta}_i'||\tilde{\delta}_i|} \right] \left[\frac{\sum_i |\tilde{\delta}_i'||\tilde{\delta}_i|}{\sum_i |\tilde{\delta}_i|^2} \right]. \tag{6.D}$$

A cross-correlation of less than 1 was observed (Figure 6.1) and suppressed the best-fit transfer function. The second term is related to the relative mode amplitudes and can be greater than 1, but any noise will reduce the covariance of the amplitudes and thus suppress the ratio. We observed this effect as well. A least-squares fit on $|\delta|$ or $|\delta|^2$ instead of δ would suffer from this same amplitude cross-correlation effect.

The second and more severe problem was that the re-weighted density field had no real-space positivity constraint, as it was constructed in Fourier space. Thus, applying the transfer function caused large regions of $\delta < -1$ in the simulation voids. The problem was severe enough that clipping these values to -1 distorted the power spectrum. Re-weighting halos with negative values was deemed too unphysical to be tenable.

As a result of these difficulties with the Eulerian transfer function, we developed the transfer formalism with displacements used in the rest of this work.

Chapter 7

Conclusions

Large-scale structure will be a leading probe of cosmology in the upcoming decade. As the number of unexplored modes on the two-dimensional surface of last scattering decreases, the information encoded in the 3D density field of the universe will become ever more important. We will not realize this potential if the modeling of the non-linear cosmological density field is not sufficiently precise, however. This places two demands on the modeling: unbiased results and large statistical samples.

In this thesis, we have addressed both of these challenges our cosmological N-body code Abacus. With a combination of new mathematical techniques to solve periodic Newtonian gravity from Metchnik (2009) and commodity computer hardware, we have executed simulations of unprecedented speed that are simultaneously orders of magnitude more accurate than results from comparable codes. The efficiency of Abacus tremendously reduces the computational hurdle for reaching the large simulation volumes required for validation of surveys like DESI, and the accuracy of the method is a substantial step towards ensuring they are unbiased.

Furthermore, the accuracy of Abacus has allowed detailed comparisons of the results of N-body simulations with analytic predictions. For example, in Chapter 3 we were able to recover linear theory at $z = 0$ better than 0.01% (while other codes disagreed at 0.5%), and in Chapter 4, we were able to validate our 2LPT implementation by directly simulating the growth of structure starting from $z = 4999$.

This is just the beginning. With Abacus, we finally have a tool that allows us to explore the extent to which N-body simulations actually reflect the physical system that they purport to model—the Vlasov-Poisson distribution of a collisionless cold dark matter species. Detailed comparisons with scale-free simulations (Appendix A) and other techniques will allow us to probe the limits of N-body deep in the non-linear regime. Only then will we be able to say whether our models of CDM large-scale structure are biased or not.

The public release of Abacus will be transformative for testing analytic methods in large-scale structure, especially techniques like perturbation theory and effective field theory where one attempts to directly model the growth of CDM structure. Numerical uncertainties will finally be able to be removed from these comparisons (even if uncertainties in how well N-body models the real universe cannot yet be removed).

Of course, the allowable cosmological parameter space is shrinking as the parameters of ΛCDM are increasingly well measured. Thus, it is worth considering if large suites of N-body simulations are really needed if there is "nothing left to measure." So long as dark matter and dark energy's fundamental natures remain unknown, ΛCDM will be an incomplete theory. Exploring extensions to ΛCDM will thus be an important part of the dark energy program of the next decade and beyond. The dimensionality of the parameter

space constrained by data will be increasing and theoretical models must keep pace.

Full N-body simulations will not be needed for every survey validation task, especially as approximate methods improve. The future of large-scale structure modeling will likely be a mix of approximate methods for testing observational systematics and N-body for theoretical systematics, with emulation techniques for performing inference in the many-dimensional parameter space that will be constrained by upcoming surveys. ABACUS is a small piece of this large puzzle but one that we hope will help increase our understanding of the universe at large.

Appendix A

Scale-Free Simulations

A.1 Background

Scale-free simulations are a classic test problem in cosmology in which an N-body simulation is initialized with a power-law power spectrum and the background cosmology is $\Omega_M = 1$ (Einstein-de Sitter, EdS). The resulting gravitational clustering is expected to be self-similar in time (Efstathiou et al. 1988). In other words, the clustering on small scales at early times is expected to be statistically identical to clustering on large scales at late times. The only scales imprinted on the simulation are the finite particle mass, finite box size, finite starting redshift, softening length, and initial particle arrangement (e.g. lattice, used here, or glass). Thus, any deviation from self-similarity must be due to one of these or inaccurate numerics. This should be true even in the deeply non-linear regime which is the true strength of this method: it is the only systematic test that allows for precise identification of resolution-dependent effects of arbitrary complexity.

Scale-free tests have traditionally been muddled by insufficient code accuracy or thwarted by the computational requirements. A steeper power law shrinks the well-resolved region between the box scale and particle scale, so a testing a small-scale ΛCDM-like power law of index $n = -2.5$ requires large N and box size. ABACUS is uniquely well-suited for these kinds of tests with its combination of accuracy and speed. Thus, we can cleanly separate the question of code accuracy and physical correctness.

The dimensionless units in which the simulations are self-similar are given by the "non-linear length scale":

$$s_{\rm NL} \propto a^{2/(3+n)}. \tag{A.1}$$

The associated non-linear mass scale is simply given by the cube of the length scale:

$$M_{\rm NL} \propto a^{6/(3+n)}. \tag{A.2}$$

A.2 Simulations

We have run three $N = 1024^3$ simulations for $n = -2$ with decreasing values of the softening length: $\epsilon = 1/15,\ 1/30,\ 1/60$, where the particle spacing is unity. We will present results from the intermediate softening in this appendix; full results will be presented in a forthcoming paper.

The simulations are initialized with the tophat standard deviation at the particle spacing scale $\sigma_l = 0.03$. Outputs begin when $\sigma_l = 0.56$ and proceed for 38 time slices. Each slice is output at a time, or epoch, where the non-linear mass scale has increased by a factor of $M_{\rm NL}^{0.5}$ from the previous slice. The outputs thus span more than 6 octaves in length scale and 18 octaves of mass scale—factors of ~ 72 and $\sim 370{,}000$.

A.3 Two-Point Correlation Function

The binning of the two-point correlation function analysis (2PCF) was set up in a scale-free manner, such that when rescaled by $s_{\rm NL}$ the bins lie exactly on top of their self-similar counterparts. This facilitates robust comparisons rather than using spline or some other interpolation procedure on a set of fixed comoving bins.

Since we are probing a factor of ~ 72 in non-linear scale length, the largest bins at late times get quite large and thus expensive to compute. We therefore exclude bins above a certain comoving scale as those are relatively well-converged anyway, at least with respect to softening and particle mass—not so with respect to finite box size! But finite box size tests may be better handled by the power spectrum analysis.

We show the raw and rescaled 2PCF results in Figures A.1 & A.2. The phenomenology is rich: in the raw results, at early times one can see the imprint of the initial lattice configuration. The system eventually "Poisson-izes" and shows smooth, scale-free evolution over a range of scales. At late times, there are small but coherent deflections in the large-scale results; this could be due to finite-box-size effects, although it is difficult to say, given that the scale is 1/200 the box size.

On small scales at all times, we see suppression from the softening extending to many times the softening scale. Quantitatively identifying the point at which the results converge to the scale-free solution (to within, say, 1%) sets an effective resolution scale. This will be measured in upcoming work.

Another way to test the self-similarity is to compare each epoch with the self-similar prediction from a later epoch. This test is shown in Figure A.3. The "later epoch" is

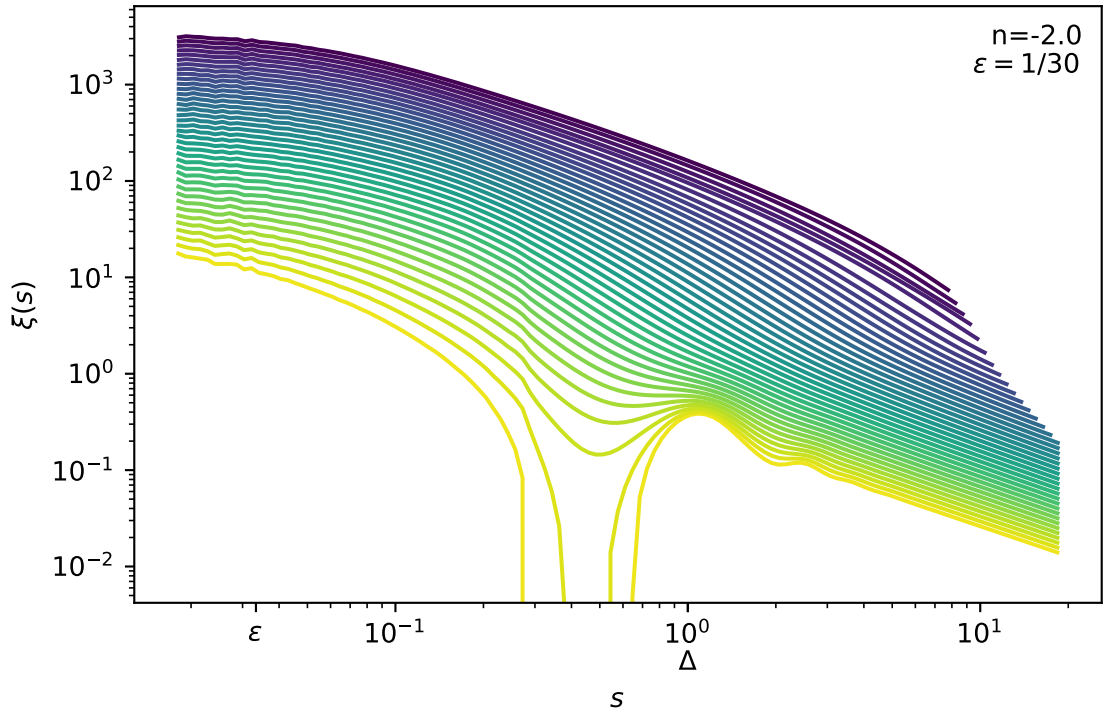

Figure A.1: Raw two-point correlation functions for power-law index $n = -2$ and softening length $\varepsilon = 1/30$ (where the initial particle spacing has length unity). Later slices appear in darker colors; the legend is given alongside the rescaled correlation functions in Figure A.2. The wiggles at early times near the particle spacing scale are the "memory" of the initial lattice configuration of the system.

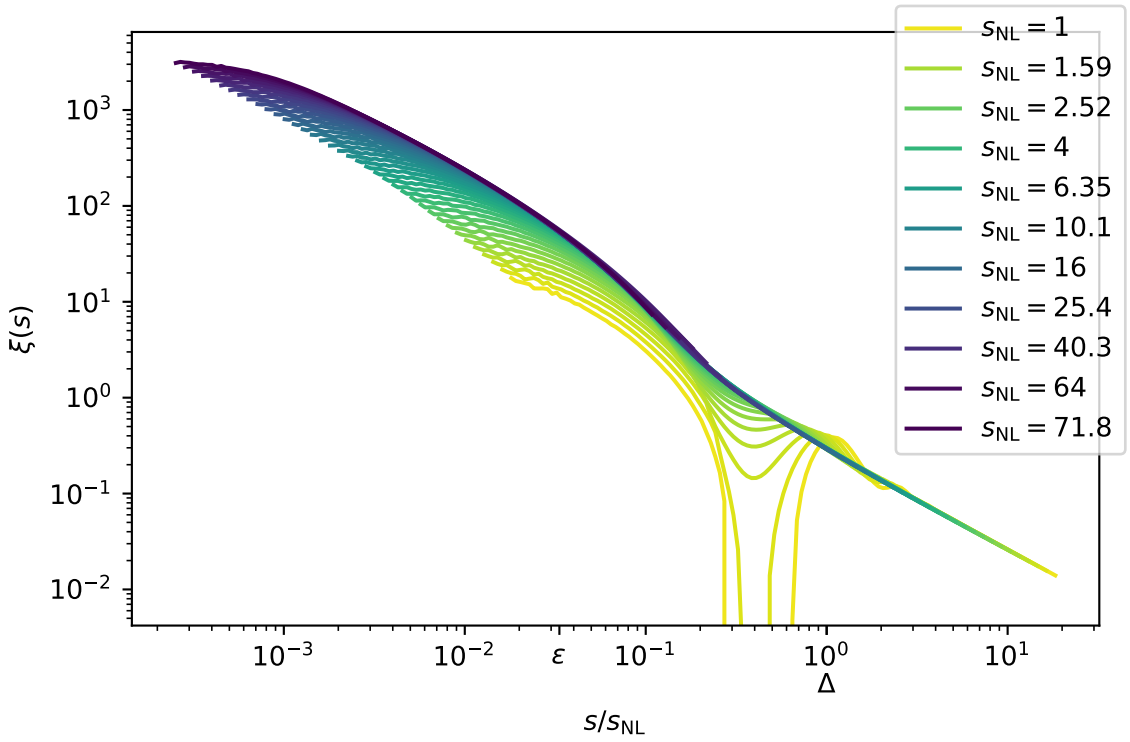

Figure A.2: Rescaling of the correlation functions in Figure A.1. $\boldsymbol{\varepsilon}$ and $\boldsymbol{\Delta}$ are marked on the abscissa for $s_{\rm NL} = 1$. All of the lines would lie on top of one another if there were no finite scales in the simulation; the fact that they do not means that self-similarity is being broken. The flattening towards small separation is likely due to a combination of softening and finite particle mass. The lower non-linear length scale $s_{\rm NL}$ corresponds to earlier times.

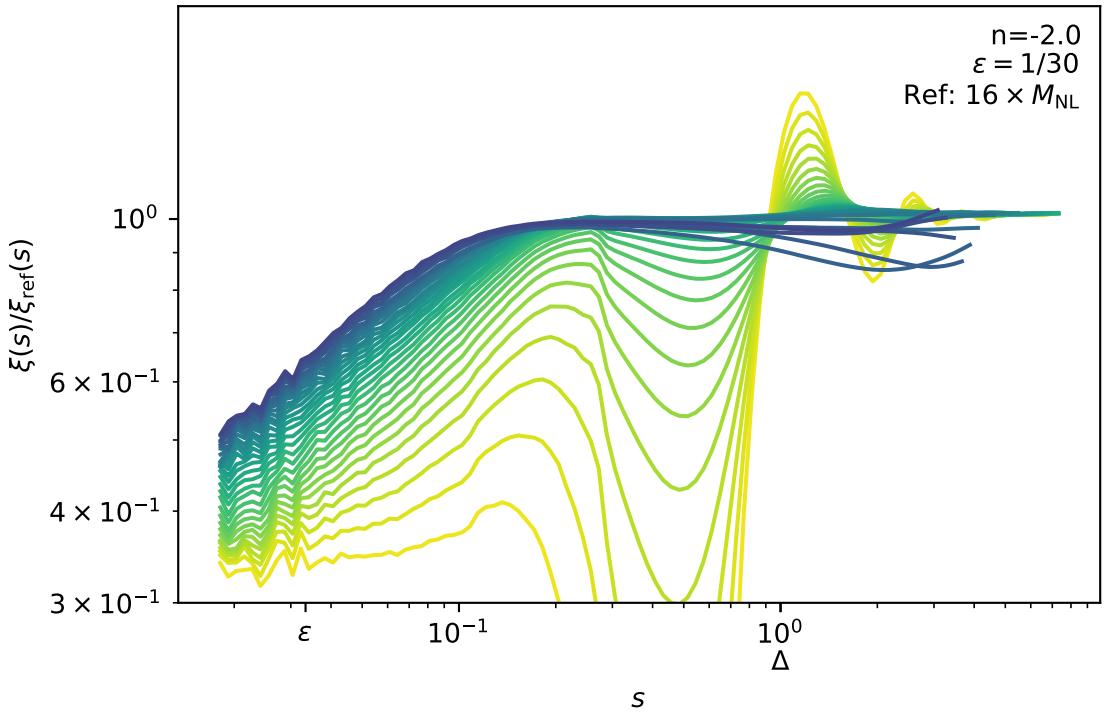

Figure A.3: The ratio of the 2PCF at a given epoch with its prediction from a later epoch scaled self-similarly to the current epoch. In detail, each epoch is compared with 8 epochs later, or a factor of 16 in $M_{\rm NL}$. Due to the presence of softening and finite particle mass, the 2PCF does not converge to the self-similar solution even in the late time, strongly clustered regime (dark lines, small scales).

chosen to be 8 epochs ahead of the current epoch, such that the non-linear mass scale has increased by a factor of 16. After early transients resolve (likely due to lack of power below the particle spacing scale in the initial conditions), the 2PCF appears to converge to a stable result. This stable result, however, does not match the self-similar prediction. This is expected due to the presence of softening and finite particle mass; disentangling the contribution of the two will require analysis of simulations with different softening lengths (or mass resolutions).

A.4 Power Spectrum

The binning of the power spectrum analysis was set up in a scale-free manner, similar to the binning for the 2PCF. Furthermore, we also changed the size of the FFT mesh itself in a scale-free manner. This is because the FFT power spectrum estimator is not perfect on small scales: there are aliasing and windowing artifacts that remain even after dividing out the alias/window functions. If $k_{\rm NL}$ is changing with respect to $k_{\rm Nyquist}$, these artifacts would appear as spurious evolution of the rescaled power spectrum. But the effects are relatively smooth with respect to the ratio $k/k_{\rm Nyquist}$, so if one can arrange $k_{\rm NL}/k_{\rm Nyquist}$ to be constant, then these effects should divide out to a good approximation.

However, the dynamic length range of 72 is punishing, since the FFT memory requirements grow with the inverse cube of the targeted length scale. In other words, if we set up a 3500^3 FFT mesh at the first time slice, then at the last time slice the FFT mesh is only 50^3! We would like to probe to higher k at intermediate and late times without losing the property of holding $k_{\rm NL}/k_{\rm Nyquist}$ fixed.

This leads to the idea of power spectrum "tracks": at half a dozen log-spaced "anchor" redshifts, we start a 3500^3 FFT mesh and proceed with smaller meshes in a scale-free manner towards lower redshift. These form a track of power spectrum measurements across time slices that can safely be compared to one another, since $k_{\rm NL}/k_{\rm Nyquist}$ is held constant. The last time slice thus has 6 FFT measurements.

In detail, $k_{\rm NL}/k_{\rm Nyquist}$ cannot be held perfectly constant since we must have an integer number of FFT mesh cells. But the Nyquist effects are fairly smooth, so getting the scaling correct to first order should suppress the worst effects.

Figures A.4 & A.5 show the raw and rescaled power spectrum results. Compared to the 2PCF analysis, the challenge of the power spectrum analysis becomes clear, especially in the second figure: the comparison is very noisy at large scales due to the mode quantization from the periodic box. The fundamental mode is fixed in comoving units but evolves in scale-free units, so the cross-epoch comparison is difficult. The results are presented here without smoothing over bins, but even aggressive smoothing/coarsifying does not remove the effect entirely. We will have to consider carefully how to interpret these results, perhaps through interpolation or model fitting.

A.5 Upcoming Work

The next step in this work is to identify converged scales and their evolution with scale factor, and repeat the analysis for different spectral index and softening length. The interpretation of the power spectrum results will require more consideration, but the results of the 2PCF analysis should be an excellent starting point.

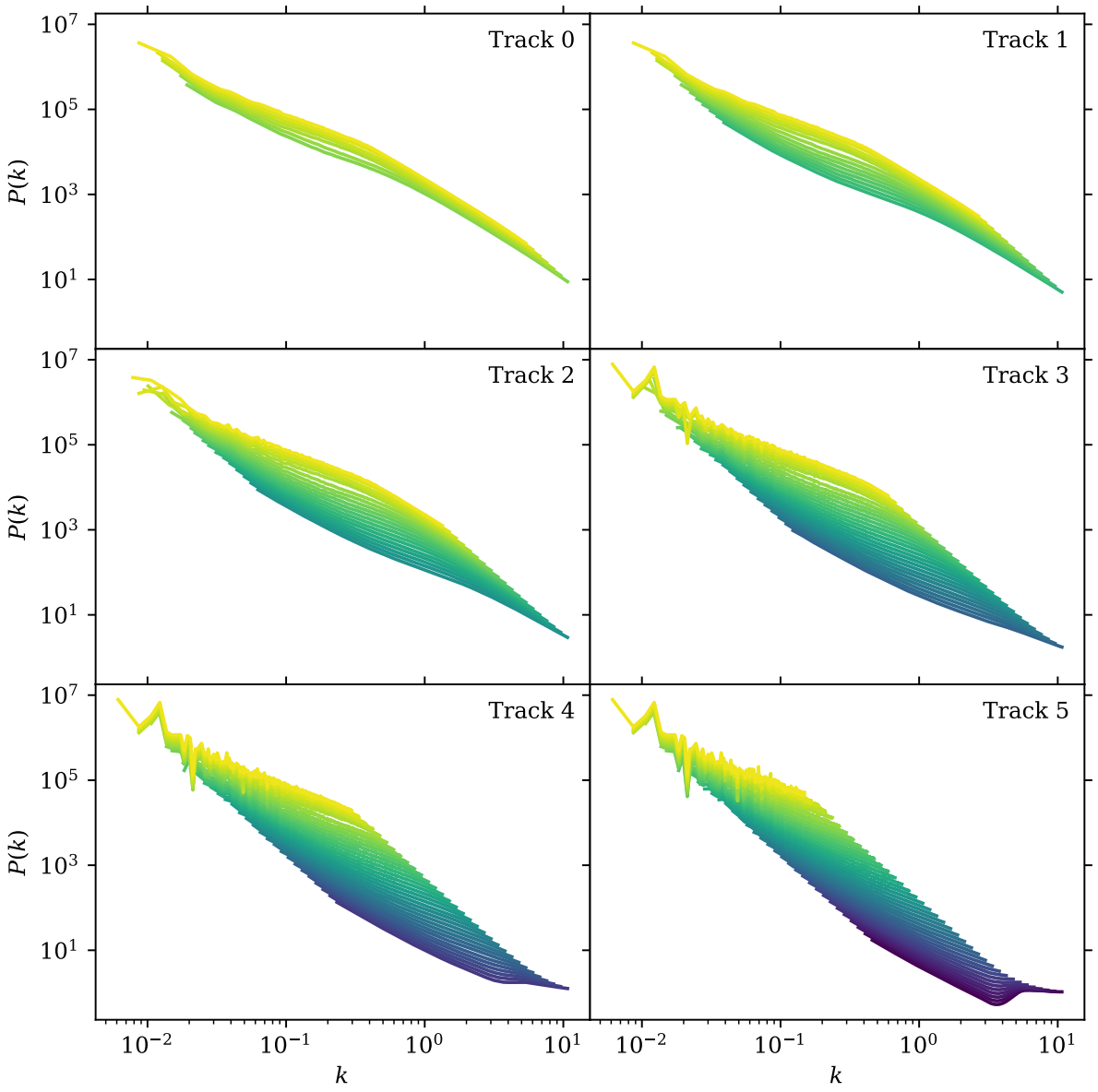

Figure A.4: Raw power spectrum results with multi-track scale-free binning. Within a track, the ratio $k_{\mathrm{NL}}/k_{\mathrm{Nyquist}}$ is kept fixed by progressively shrinking the FFT mesh size. Multiple tracks are measured so that late times can be probed to small scales, even though k_{Nyquist} is shrinking. Noise from the mode quantization is visible on large scales; the emergence of Poisson noise can be seen on small scales.

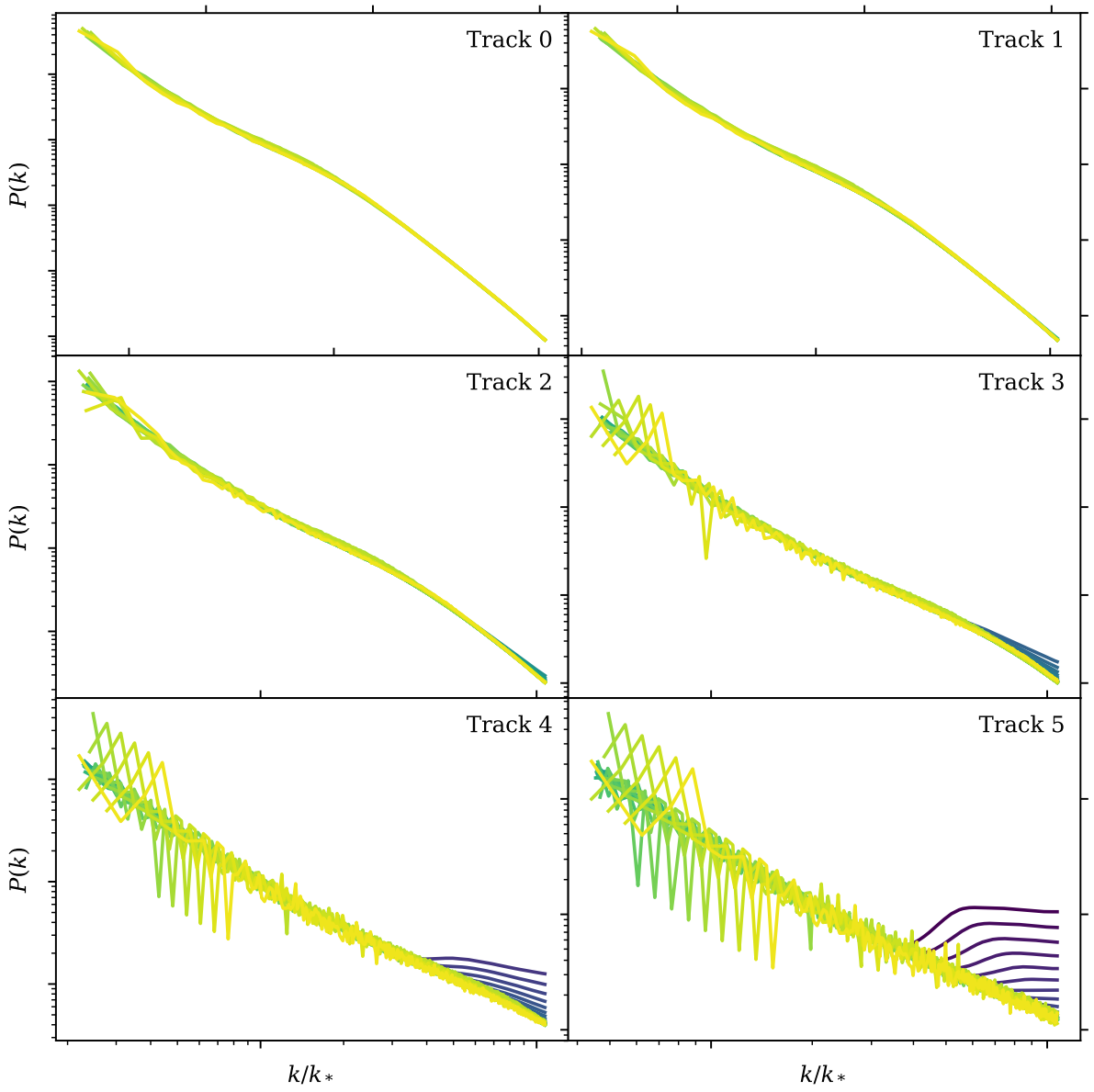

Figure A.5: Same as Figure A.4 but in scale-free units. N.B. the axes do not span the same k or $P(k)$ range in each panel; they are left independent so that the full dynamic range can be presented in each. See the previous figure for the absolute magnitudes of the results.

These scale-free tests necessarily use a power-law power spectrum and EdS background. Of course, neither of these are strictly true in ΛCDM. But simulating multiple n at least gives bounding behaviors; interpolating between n to the effective spectral index of a given scale in ΛCDM seems a promising route to estimating systematic errors. And ΛCDM does behave like EdS at early times; it's unclear how much differences would matter even at late times as long as the comparison is done by matching the non-linear ΛCDM mass scale to the EdS value.

Many more tests besides two-point statistics are possible on these simulations. Indeed, any statistic one can dream up to measure at one time should rescale to another time. The most informative statistics for deciding the trustworthiness of N-body simulations is still an area of investigation.

Appendix B

Numerically Stable Computation of σ_8 with a Power-Law Power Spectrum

Cosmologies with a power-law power spectrum ($P(k) = k^n$) have an analytic form for $\sigma^2(R)$ with a spherical tophat window W_T:

$$\begin{aligned}\sigma^2(R) &= \int \frac{d^3k}{(2\pi)^3} \tilde{W}_T^2(kR) P(k) \\ &= \int_0^\infty \frac{k^2 dk}{2\pi^2} \left[\frac{3(\sin(kR) - kR\cos(kR))}{(kR)^3}\right]^2 k^n \\ &= \frac{9(n+1)2^{-n-1}R^{-n-3}\sin\left(\frac{\pi n}{2}\right)\Gamma(n-1)}{\pi^2(n-3)},\end{aligned} \tag{B.1}$$

where we used the Fourier transform $\tilde{W}_T(kR)$ of the spherical tophat window. The result in the last line (obtained via Mathematica) is only valid for $-3 < n < 1$; otherwise, the variance is undefined.

However, the result is numerically unstable if n is an integer in this range, since the function $\Gamma(n-1)$ in the numerator goes to infinity if $n-1$ is 0 or a negative integer. Exact integer values of n arise commonly in studies of scale-free cosmologies (EdS background and power-law power spectrum), so it is convenient to have a closed-form expression for the normalization of such a power spectrum, especially for generating initial conditions for N-body simulations.

The $\sigma^2(R)$ function itself is perfectly smooth and finite in this range, as seen in Fig. B.1.

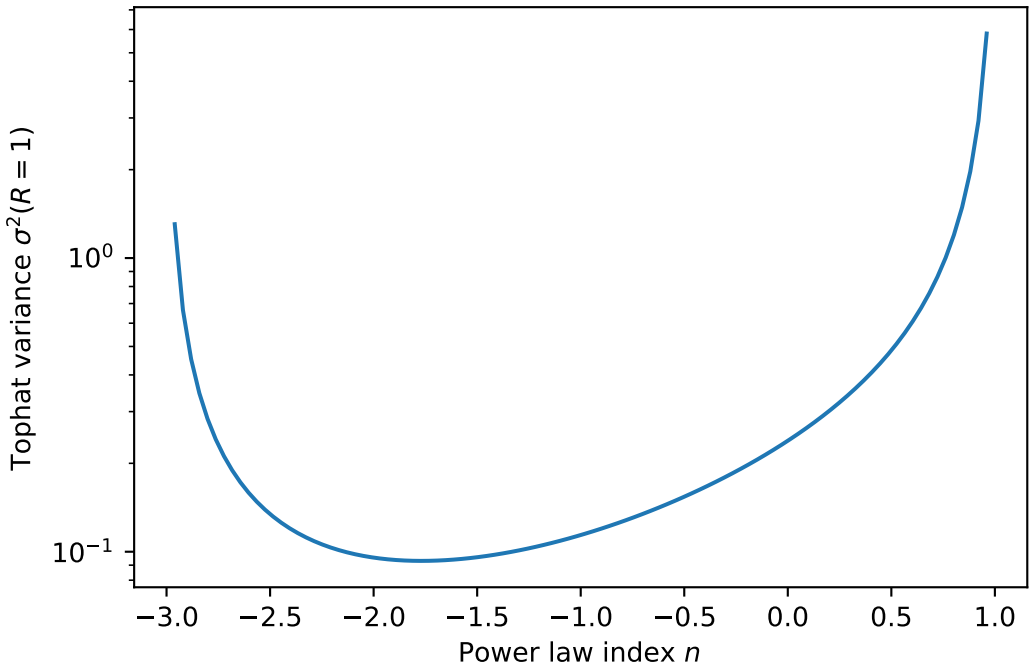

Figure B.1: $\sigma^2(1)$ versus power-law index n using Eq. B.1. For plotting purposes, we avoided exact integers in the n range.

So, to find a numerically stable variant of the above expression, we need to re-write it to avoid evaluating $\Gamma(n-1)$ at negative integers.

Using some properties of the Γ function (DLMF, Section 5.5), we can arrive at the following expressions, each of which is numerically stable around even or odd integer poles:

Even poles $(n = 0, -2)$

$$\begin{aligned}\sigma^2(R) &= \left[\frac{9R^{-n-3}2^{-n-1}}{\pi^2(n-3)}\right]\left[(n+1)\sin\left(\frac{\pi n}{2}\right)\Gamma(n-1)\right] \\ &= \left[\frac{9R^{-n-3}2^{-n-1}}{\pi^2(n-3)}\right]\left[\frac{-(n+1)\pi}{2\Gamma(2-n)}\sec\left(\frac{\pi n}{2}\right)\right]\end{aligned} \tag{B.2}$$

Odd pole $(n = -1)$

$$\sigma^2(R) = \left[\frac{9R^{-n-3}2^{-n-1}}{\pi^2(n-3)}\right]\left[\frac{\sin\left(\frac{\pi n}{2}\right)\Gamma(n+2)}{n(n-1)}\right] \tag{B.3}$$

It's worth noting that these expressions are exact; we have not used any approximations or Taylor expansions. Each is simply more convenient to evaluate numerically near a given pole. Because of this, the transition point between functions is basically arbitrary; it's convenient to round to the nearest pole and check if it's even or odd.

Table B.1 compares the values of $\sigma^2(8)$ computed with the "naive" form from Mathematica (Eq. B.1) with the "stable" forms Eqs. B.2 & B.3. The spot-checks in the areas where the naive, even-pole, and odd-pole forms are all valid show that the identities were implemented correctly.

These stable expressions for $\sigma^2(R)$ are used by the Abacus initial conditions generator[1] when requesting a power-law cosmology.

[1] `https://github.com/lgarrison/zeldovich-PLT`

n	Naive	Even	Odd	Stable
-2.90	0.432508	0.432508	0.432508	0.432508
-2.50	0.0474965	0.0474965	0.0474965	0.0474965
-2.00	-inf	0.0119366	inf	0.0119366
-1.00	nan	0	0.00178104	0.00178104
0.00	nan	0.000466274	nan	0.000466274
0.75	0.000392073	0.000392073	0.000392073	0.000392073

Table B.1:: Numerical comparison of analytically-equivalent expressions for σ_8^2. Note how all except "Stable" fail at some integer n. The stable result is formed by taking the "Even" result near even n and "Odd" result near odd n.

Appendix C

Importance Sampling for the Covariance of the 2PCF

C.1 Abstract

The Gaussian part of the covariance of the two-point correlation function can be expressed as the auto-convolution of the 2PCF which can be computed with Monte Carlo integration. We derive an importance sampling kernel that accelerates the convergence of this integral.

C.2 Gaussian Covariance of the 2PCF

We define the two-point correlation function (2PCF) as the autocorrelation of the density contrast $\delta(\mathbf{r})$:

$$\xi(\mathbf{s}) = \frac{1}{V}\int d^3\mathbf{r}\,\delta(\mathbf{r}+\mathbf{s})\delta(\mathbf{r}), \tag{C.1}$$

where we are working in a periodic domain of volume V. We define the covariance of ξ with respect to the two separations $\mathbf{s}_1$, $\mathbf{s}_2$ as

$$C_{12} = \langle \xi(\mathbf{s}_1)\xi(\mathbf{s}_2)\rangle - \langle \xi(\mathbf{s}_1)\rangle\langle \xi(\mathbf{s}_2)\rangle, \tag{C.2}$$

where the expectation values are taken with respect to different realizations of the initial conditions.

Substituting Eq. C.1 into Eq. C.2 and grouping δ terms,

$$C_{12} = \frac{1}{V^2}\int d^3\mathbf{r}_1 \int d^3\mathbf{r}_2 \,\langle \delta(\mathbf{r}_1)\delta(\mathbf{r}_1+\mathbf{s}_1)\delta(\mathbf{r}_2)\delta(\mathbf{r}_2+\mathbf{s}_2)\rangle - \langle \xi(\mathbf{s}_1)\rangle\langle \xi(\mathbf{s}_2)\rangle, \tag{C.3}$$

where we have brought the expectation value inside the integrals. Note that $\langle \xi(\mathbf{s})\rangle = \xi(\mathbf{s})$.

The expectation of the product of four Gaussian deviates can be expanded by Wick's theorem:

$$\begin{aligned}
\langle \delta(\mathbf{r}_1)\delta(\mathbf{r}_1+\mathbf{s}_1)\delta(\mathbf{r}_2)\delta(\mathbf{r}_2+\mathbf{s}_2)\rangle = \;& \langle \delta(\mathbf{r}_1)\delta(\mathbf{r}_1+\mathbf{s}_1)\rangle\langle \delta(\mathbf{r}_2)\delta(\mathbf{r}_2+\mathbf{s}_2)\rangle \\
& + \langle \delta(\mathbf{r}_1)\delta(\mathbf{r}_2)\rangle\langle \delta(\mathbf{r}_1+\mathbf{s}_1)\delta(\mathbf{r}_2+\mathbf{s}_2)\rangle \\
& + \langle \delta(\mathbf{r}_1)\delta(\mathbf{r}_2+\mathbf{s}_2)\rangle\langle \delta(\mathbf{r}_1+\mathbf{s}_1)\delta(\mathbf{r}_2)\rangle \\
= \;& \xi(\mathbf{s}_1)\xi(\mathbf{s}_2) \\
& + \xi(\mathbf{r}_1-\mathbf{r}_2)\xi(\mathbf{s}_1-\mathbf{s}_2-(\mathbf{r}_2-\mathbf{r}_1)) \\
& + \xi(\mathbf{s}_2-(\mathbf{r}_1-\mathbf{r}_2))\xi(\mathbf{s}_1-(\mathbf{r}_2-\mathbf{r}_1)),
\end{aligned} \tag{C.4}$$

where we have used $\xi(\mathbf{s}) = \langle \delta(\mathbf{r})\delta(\mathbf{r}+\mathbf{s})\rangle$ which is equivalent to Eq. C.1 assuming homogeneity.

Working term by term, we substitute back into the double integral in Eq. C.3:

$$\frac{1}{V^2}\int d^3\mathbf{r}_1 \int d^3\mathbf{r}_2 \,\xi(\mathbf{s}_1)\xi(\mathbf{s}_2) = \xi(\mathbf{s}_1)\xi(\mathbf{s}_2). \tag{C.5}$$

We immediately see that this cancels the second term in Eq. C.3.

The second term becomes

$$\frac{1}{V^2}\int d^3\mathbf{r}_1 \int d^3\mathbf{r}_2\, \xi(\mathbf{r}_1-\mathbf{r}_2)\xi(\mathbf{s}_1-\mathbf{s}_2-(\mathbf{r}_2-\mathbf{r}_1)). \tag{C.6}$$

We now make a change of integration variables: $\mathbf{s}' \equiv \mathbf{r}_2-\mathbf{r}_1$ and $\Delta_{12} \equiv \mathbf{s}_1-\mathbf{s}_2$:

$$\frac{1}{V^2}\int d^3\mathbf{r}_1 \int d^3\mathbf{s}'\, \xi(\mathbf{s}')\xi(\Delta_{12}-\mathbf{s}'), \tag{C.7}$$

where we have used $\xi(\mathbf{s}) = \xi(-\mathbf{s})$. The origin of the inner integral is shifted by $-\mathbf{r}_1$, but in a periodic box this has no effect, so the integral is still over the same volume. This decouples the two integrals, so the outer integral resolves to a factor of V.

The third term in Eq. C.4 gives a similar result to the second (Eq. C.7), with the only difference being the replacement of Δ_{12} by $\Sigma_{12} = \mathbf{s}_1+\mathbf{s}_2$. Later, in the bin-averaging process, these two terms will become identical, but we will not explicitly show that here.

In summary, thus far we have

$$C_{12} = \frac{1}{V}\int d^3\mathbf{s}'\, \xi(\mathbf{s}')\xi(\Delta_{12}-\mathbf{s}') + \frac{1}{V}\int d^3\mathbf{s}'\xi(\mathbf{s}')\xi(\Sigma_{12}-\mathbf{s}'). \tag{C.8}$$

Both of these integrals are 3D auto-convolutions of ξ. We will compute these integrals with Monte Carlo sampling in the next section.

C.3 Monte Carlo Integration

We wish to compute convolution integrals of the form

$$(\xi\otimes\xi)(\mathbf{s}) = \int d^3\mathbf{s}'\, \xi(\mathbf{s}')\xi(\mathbf{s}-\mathbf{s}'). \tag{C.9}$$

Since we may be interested in covariance on small scales and with non-uniform binning, we will approach this integral with Monte Carlo sampling.

We can write the 3D integral in spherical coordinates using $\mathbf{s}$ as the azimuthal axis without loss of generality:

$$(\xi \otimes \xi)(\mathbf{s}) = \int_0^{2\pi} d\phi \int_0^S ds' \int_{-1}^1 d\mu \, s'^2 \xi(\mathbf{s}') \xi(\mathbf{s} - \mathbf{s}'), \tag{C.10}$$

where $\mu = \cos(\theta)$ and θ is the polar angle. S is the maximum radius, which we will eventually take to infinity but leave as a bookkeeping notation for now.

For brevity, we will only consider the isotropic case $\xi(\mathbf{s}) = \xi(|\mathbf{s}|)$ in the rest of this appendix. In terms of the integration variables, $|\mathbf{s}'| = s'$ and $|\mathbf{s} - \mathbf{s}'| = \sqrt{s^2 + s'^2 - 2ss'\mu}$. We can thus resolve the ϕ integral to 2π:

$$(\xi \otimes \xi)(\mathbf{s}) = 2\pi \int_0^S ds' \int_{-1}^1 d\mu \, s'^2 \xi(s') \xi\left(\sqrt{s^2 + s'^2 - 2ss'\mu}\right). \tag{C.11}$$

If we consider the likely form of $\xi(s)$, the challenge of computing this integral with Monte Carlo sampling becomes clear. To zeroth order, $\xi(s) \propto 1/s^2$, leading to

$$(\xi \otimes \xi)(\mathbf{s}) \propto \int_0^S ds' \int_{-1}^1 d\mu \, s'^2 \frac{1}{s'^2} \frac{1}{s^2 + s'^2 - 2ss'\mu}. \tag{C.12}$$

The first two terms in the integrand cancel, leaving all of the integrand weight to occur near the pole $s'^2 + s'^2 - 2ss'\mu = 0$, where $\mathbf{s} = \mathbf{s}'$.

To efficiently evaluate this integral, we would like to sample (s', μ) from $1/(s^2 + s'^2 - 2ss'\mu)$ rather than a uniform distribution, such that more samples appear near the pole, thus suppressing the Monte Carlo variance. This is the idea of importance sampling.

First, let us define

$$g(s', \mu) = \frac{1}{s^2 + s'^2 - 2ss'\mu} \tag{C.13}$$

as our target importance sampling kernel. This is a 2D (unnormalized) joint probability distribution function that we can rewrite as the product of a marginalized and a condi-

tional 1D distribution:

$$g(s',\mu) = g(s'|\mu)g(\mu) = g(\mu|s')g(s'). \tag{C.14}$$

We are free to choose either form based on numerical convenience. We will try both to see if one is more convenient.

Let us first consider the marginal in the second form:

$$g(s') = \int_{-1}^{1} d\mu\, g(s',\mu) = \frac{1}{2ss'} \log\left[\frac{(s+s')^2}{(s'-s)^2}\right]. \tag{C.15}$$

This is a closed form; now we need to sample from it. We will try to do so with a probability integral transform: calculating the cumulative distribution function (CDF) $g(<s')$, setting it equal to a uniform variable X, and solving for s'.

The CDF has a closed form, but is not invertible:

$$g(<s') = \frac{2}{\pi}\left[-\mathrm{Li}_2\left(-\frac{s'}{s}\right) + \mathrm{Li}_2\left(\frac{s'}{s}\right)\right], \tag{C.16}$$

where Li_2 is the dilogarithm. There is no clear way to invert the above equation for s', so we will not bother to compute the conditional $g(\mu|s')$.

The marginal in the first form of Eq. C.14 is:

$$g(\mu) = \int_0^\infty ds'\; g(s',\mu) = \frac{\pi - \cos^{-1}(\mu)}{s'\sqrt{1-\mu^2}}. \tag{C.17}$$

The CDF is

$$g(<\mu) = \frac{\cos^{-1}(\mu)^2 + 2\pi\sin^{-1}(\mu)}{\pi^2}. \tag{C.18}$$

This is invertible, unlike $g(<s')$:

$$g(<\mu) = \frac{\cos^{-1}(\mu)^2 + 2\pi\sin^{-1}(\mu)}{\pi^2} = X$$
$$\mu = -\cos(\pi\sqrt{X}) \tag{C.19}$$

Now let us try to sample from the associated marginal by computing the CDF:

$$g(<s'|\mu) = \frac{\sin^{-1}(\mu) - \tan^{-1}[(-s' + \mu s')/(s'\sqrt{1-\mu^2})]}{\pi - \cos^{-1}(\mu)}. \tag{C.20}$$

Despite appearances, this too is invertible:

$$\begin{aligned} g(<s'|\mu) &= X \\ s' &= s\left(\mu + \sqrt{1-\mu^2}\tan[(\pi - \cos^{-1}(\mu))X - \sin^{-1}(\mu)]\right). \end{aligned} \tag{C.21}$$

Thus, combining Eqs. C.19 and C.21, we have a fully analytic mapping from the unit square to our desired importance sampling kernel $g(s,\mu)$. Testing this sampling kernel on simulation data yields nearly a factor of 600 in variance reduction.

In summary, this importance sampling kernel maps an infinite domain with a pole to a smooth, finite domain. The result is nearly $600\times$ faster integral convergence. Furthermore, the coordinate mapping is fully analytic, so Sobol sampling and other techniques that surpass Monte Carlo sampling in variance reduction should now apply. While we have only considered the isotropic case in the above, extension to the anisotropic case may be possible. Applying bin selection operators to compute the binned covariance (and not just the point-wise covariance) is a natural next step in this work.

Appendix D

Lagrangian Perturbation Theory for Initial Conditions

Initial conditions codes for cosmological N-body simulations typically use Lagrangian perturbation theory (LPT; Zel'dovich 1970) to generate particle displacements and velocities from an initial homogeneous lattice configuration. In linear perturbation theory, the spatial and temporal parts of this problem are decoupled such that the particle displacements can be written as:

$$\boldsymbol{\psi}(a, \mathbf{q}) = D(a)\boldsymbol{\psi}(\mathbf{q}), \tag{D.1}$$

where a is the scale factor, $D(a)$ is the linear growth factor and $\boldsymbol{\psi}(\mathbf{q})$ is the displacement for the particle with initial (lattice) location $\mathbf{q}$. This initial location $\mathbf{q}$ is also known as the Lagrangian coordinate.

The linear growth factor $D(a)$ is readily obtained through integration of the linear growth equation. In this appendix, we will focus instead on how the spatial part $\boldsymbol{\psi}(\mathbf{q})$ is

obtained.

The most common method is to solve Poisson's equation in Fourier space:

$$\psi(\mathbf{k}) = -\frac{i\mathbf{k}}{k^2}\delta(\mathbf{k}), \tag{D.2}$$

$$\psi(\mathbf{q}) = \mathscr{F}^{-1}[\psi(\mathbf{k})], \tag{D.3}$$

where $\delta = \rho/\bar{\rho} - 1$ is the overdensity with ρ as the density and $\bar{\rho}$ as the mean density, $\mathbf{k}$ is the wavevector with amplitude k, and i is the imaginary unit. The inverse Fourier transform is indicated by $\mathscr{F}^{-1}$. This is typically taken as a starting point in most papers on initial conditions (e.g. Garrison et al. 2016, Chapter 4 of this thesis). The purpose of this appendix is to demonstrate pedagogically how Equation D.2 arises.

LPT begins with the ansatz that all density fluctuations arise from displacement of mass elements from their initial locations $\mathbf{q}$ at $t = 0$ (or $z = \infty$) to their current locations

$$\mathbf{x} = \mathbf{q} + \psi(\mathbf{q}). \tag{D.4}$$

$\mathbf{x}$ is known as the Eulerian position.

We can follow the evolution of Eulerian density δ by local conservation of mass:

$$\rho d\mathbf{x} = \rho_L d\mathbf{q}$$

$$\bar{\rho}(1+\delta)d\mathbf{x} = \bar{\rho}_L(1+\delta_L)d\mathbf{q}$$

$$(1+\delta)d\mathbf{x} = d\mathbf{q}, \tag{D.5}$$

where ρ_L and δ_L are the densities in the Lagrangian frame. To arrive at the last expression, we used $\bar{\rho} = \bar{\rho}_L$ by global conservation of mass, and $\delta_L = 0$ from our anzatz that all Eulerian density fluctuations arise from displacements such that density is uniform in the Lagrangian frame.

We wish to find a set of displacements $\mathbf{q}$ that will produce Eulerian density δ. To do so, we compute the Jacobian of the mapping from Lagrangian to Eulerian coordinates:

$$\begin{aligned} J &\equiv \left| \frac{d\mathbf{x}_i}{d\mathbf{q}_j} \right|, \\ &= \left| \frac{d(\mathbf{q}_i + \psi_i)}{d\mathbf{q}_j} \right|, \\ &= \left| \delta_{ij} + \frac{d\psi_i}{d\mathbf{q}_j} \right|, \end{aligned} \tag{D.6}$$

where we used Equation D.4 in the second line and δ_{ij} is the Kronecker delta. The Jacobian matrix in D.6 is also known as the ?.

From Equation D.5 and the definition of the Jacobian, we also have

$$1 + \delta = J^{-1}. \tag{D.7}$$

Now we apply the perturbative approximation $d\psi/d\mathbf{q} \ll 1$ and evaluate the determinant in Equation D.6. To first order, only the diagonal terms in the Jacobian survive due to the Kronecker delta:

$$\begin{aligned} \left| \delta_{ij} + \frac{d\psi_i}{d\mathbf{q}_j} \right| &\approx \left(1 + \frac{d\psi_0}{d\mathbf{q}_0}\right)\left(1 + \frac{d\psi_1}{d\mathbf{q}_1}\right)\left(1 + \frac{d\psi_2}{d\mathbf{q}_2}\right) \\ &\approx 1 + \frac{d\psi_0}{d\mathbf{q}_0} + \frac{d\psi_1}{d\mathbf{q}_1} + \frac{d\psi_2}{d\mathbf{q}_2} \\ &\approx 1 + \nabla_{\mathbf{q}} \cdot \psi \end{aligned} \tag{D.8}$$

Using Equation D.7 to connect this to the density, we have

$$\begin{aligned} 1 + \delta &\approx \frac{1}{1 + \nabla \cdot \psi}, \\ \delta &\approx -\nabla \cdot \psi, \end{aligned} \tag{D.9}$$

where we used the Taylor approximation $(1+x)^{-1} \approx (1-x)$ for small x (thus only keeping terms to first order as we did in Equation D.8). Thus, to linear order, the Eulerian density

field is given by the divergence of the Lagrangian displacement field. Intuitively, regions of particle convergence will be regions of high density, while regions of particle divergence will be regions of low density.

How do we invert Equation D.9 to get a set of displacements from a density field? Now we can apply Poisson's equation and assume an irrotational field:

$$\nabla^2\phi = 4\pi G\bar{\rho}\,\delta, \tag{D.10}$$

$$\nabla\cdot(\nabla\phi) \approx -4\pi G\bar{\rho}(\nabla\cdot\psi),$$

$$\nabla\phi \approx -4\pi G\bar{\rho}\,\psi, \tag{D.11}$$

so the gradient of the potential gives the Lagrangian displacements. The potential is given by Equation D.10 which is easily solved in Fourier space:

$$\phi(\mathbf{k}) = -\frac{4\pi G\bar{\rho}\,\delta(\mathbf{k})}{k^2}. \tag{D.12}$$

Combining this with Equation D.11 in Fourier space yields:

$$-4\pi G\bar{\rho}\,\psi(\mathbf{k}) = -i\mathbf{k}\phi(\mathbf{k}),$$

$$-4\pi G\bar{\rho}\,\psi(\mathbf{k}) = i\mathbf{k}\frac{4\pi G\bar{\rho}\,\delta(\mathbf{k})}{k^2},$$

$$\psi(\mathbf{k}) = -\frac{i\mathbf{k}}{k^2}\delta(\mathbf{k}), \tag{D.13}$$

which is Equation D.2, our desired result.

Appendix E

The RR Term in Particle Auto-Correlations

E.1 RR in Unweighted Clustering Statistics

When computing a two-point correlation function estimator like $\xi(r) = \mathrm{DD}/\mathrm{RR} - 1$, the RR term can be computed analytically if the domain is a periodic box. Often, this is done as

$$\mathrm{RR}_i = NV_i\bar{\rho} \tag{E.1}$$

$$= NV_i\frac{N}{L^3} \tag{E.2}$$

where RR_i is the expected number of random-random pairs in bin i, N is the total number of points, V_i is the volume (or volume or area if 2D) of bin i, L is the box size, and $\bar{\rho}$ is the average density in the box.

However, using $\bar{\rho} = N/L^3$ is only correct for continuous fields, not sets of particles.

When sitting on a particle, only $N-1$ particles are available to be in a bin at some non-zero distance. The remaining particle is the particle you're sitting on, which is always at distance 0. Thus, the correct expression is

$$\mathrm{RR}_i = NV_i \frac{N-1}{L^3}. \tag{E.3}$$

Using density N instead of $N-1$ introduces a bias of order $1/N$ into the estimator. We can easily check this empirically. Consider generating a set of 10 particles with independent, uniformly-random positions in a box of unit size. Because their positions are independent and random, we should measure a correlation function of zero. Of course, for any given realization, we will see large Poisson fluctuations, so we can repeat this experiment 100K times to boost the signal-to-noise.

Executing this experiment with four evenly spaced bins from 0.1 to 0.4 yields:

- DD/RR using density N: $[0.89871, 0.89998, 0.90135, 0.89938]$
- DD/RR using density $N-1$: $[0.99857, 0.99998, 1.00151, 0.99931]$

Thus, we see that density of N yields a $1/N = 10\%$ biased answer, while $N-1$ yields an unbiased answer.

Of course, this is a tiny correction for large N problems, but important for small N. When comparing results from different correlation function codes, this may be one source of systematic discrepancy.

Cross-correlations of two different particle sets don't suffer from this problem; the particle you're sitting on is never part of the set of particles under consideration for pair-making.

This $N-1$ correction is implemented in the CORRFUNC code Sinha & Garrison (2017b).

E.2 RR in Weighted Clustering Statistics

We can extend the above discussion to weighted correlation functions in which each particle is assigned a weight, and the pair weight is taken as the product of the particle weights.

Let w_j be the weight of particle j, and W be the sum of the weights. The correspondence to an "unclustered" distribution is less clear than in the uniform case, but we will define it as the case of N particles uniformly distributed, where each is assigned the mean weight $\bar{w}$. We thus have

$$\begin{aligned} \mathrm{RR}_i &= \sum_{j=1}^{N} \bar{w}(W-\bar{w})\frac{V_i}{L^3} \\ &= (W^2 - \bar{w}W)\frac{V_i}{L^3} \\ &= W^2\left(1-\frac{1}{N}\right)\frac{V_i}{L^3}. \end{aligned} \tag{E.4}$$

When the particles all have $w_j = 1$, then $W = N$ and we recover the unweighted result from above.

Alternatively, we could define the unclustered distribution by redistributing the particles uniformly but preserve their individual weights. In this case, we would find

$$\mathrm{RR}_i = \sum_{j=1}^{N} w_j(W-w_j)\frac{V_i}{L^3} \tag{E.5}$$

$$= \left(W^2 - \sum_{j=1}^{N} w_j^2\right)\frac{V_i}{L^3}. \tag{E.6}$$

One should probably define the unclustered distribution based on the physical meaning of the assigned weights. But in most practical cases, this can be ignored unless N is small or the variance of the weights is large.

References

Aarseth, S. J. 1963, MNRAS, 126, 223

—. 2003, Gravitational N-Body Simulations (Cambridge University Press)

Aarseth, S. J., Gott, J. R., I., & Turner, E. L. 1979, ApJ, 228, 664

Abazajian, K. N., Adshead, P., Ahmed, Z., et al. 2016, arXiv e-prints, arXiv:1610.02743

Abbott, T. M. C., Abdalla, F. B., Alarcon, A., et al. 2018a, Phys. Rev. D, 98, 043526

—. 2018b, Phys. Rev. D, 98, 043526

Abel, T., Hahn, O., & Kaehler, R. 2012, MNRAS, 427, 61

Abell, G. O. 1965, Annual Review of Astronomy and Astrophysics, 3, 1

Adamek, J., Daverio, D., Durrer, R., & Kunz, M. 2013, Phys. Rev. D, 88, 103527

—. 2016, Journal of Cosmology and Astro-Particle Physics, 2016, 053

Adamek, J., Durrer, R., & Kunz, M. 2014, Classical and Quantum Gravity, 31, 234006

Albrecht, A., Bernstein, G., Cahn, R., et al. 2006, arXiv e-prints, astro

Anderson, L., Aubourg, É., Bailey, S., et al. 2014, MNRAS, 441, 24

Angulo, R. E., Hahn, O., & Abel, T. 2013, MNRAS, 434, 1756

Angulo, R. E., & White, S. D. M. 2010, Monthly Notices of the Royal Astronomical Society, 405, 143

Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123

Bahcall, N. A. 1977, Annual Review of Astronomy and Astrophysics, 15, 505

Banerjee, A., Powell, D., Abel, T., & Villaescusa-Navarro, F. 2018, Journal of Cosmology and Astro-Particle Physics, 2018, 028

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

Bartelmann, M., & Schneider, P. 2001, Phys. Rep., 340, 291

Baumgardt, H., & Makino, J. 2003, MNRAS, 340, 227

Behroozi, P., Knebe, A., Pearce, F. R., et al. 2015, MNRAS, 454, 3020

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

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

Bernardeau, F., Colombi, S., Gaztañaga, E., & Scoccimarro, R. 2002, Phys. Rep., 367, 1

Bertschinger, E., & Gelb, J. M. 1991, Computers in Physics, 5, 164

Binney, J., & Knebe, A. 2002, MNRAS, 333, 378

Bird, S., Ali-Haïmoud, Y., Feng, Y., & Liu, J. 2018, MNRAS, 481, 1486

Brand, J., & Blitz, L. 1993, A&A, 275, 67

Burden, A., Percival, W. J., Manera, M., et al. 2014, MNRAS, 445, 3152

Caldwell, R. R., Dave, R., & Steinhardt, P. J. 1998, Phys. Rev. Lett., 80, 1582

Carr, B., Kühnel, F., & Sandstad, M. 2016, Phys. Rev. D, 94, 083504

Centrella, J., & Melott, A. L. 1983, Nature, 305, 196

Chambers, J. E., & Migliorini, F. 1997, in AAS/Division for Planetary Sciences Meeting Abstracts #29, AAS/Division for Planetary Sciences Meeting Abstracts, 27.06

Chisari, N. E., & Zaldarriaga, M. 2011, Phys. Rev. D, 83, 123505

Chuang, C.-H., Prada, F., Pellejero-Ibanez, M., et al. 2016, MNRAS, 461, 3781

Cimatti, A., Robberto, M., Baugh, C., et al. 2009, Experimental Astronomy, 23, 39

Comparat, J., Prada, F., Yepes, G., & Klypin, A. 2017, ArXiv e-prints, arXiv:1702.01628

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

Cooley, J. W., & Tukey, J. W. 1965, Mathematics of Computation, 19, 297

Copeland, E. J., Sami, M., & Tsujikawa, S. 2006, International Journal of Modern Physics D, 15, 1753

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

Dai, B., Feng, Y., & Seljak, U. 2018, Journal of Cosmology and Astro-Particle Physics, 2018, 009

Davis, M., Efstathiou, G., Frenk, C. S., & White, S. D. M. 1985, ApJ, 292, 371

Davis, M., Huchra, J., Latham, D. W., & Tonry, J. 1982, ApJ, 253, 423

D'Ercole, A., Vesperini, E., D'Antona, F., McMillan, S. L. W., & Recchi, S. 2008, MNRAS, 391, 825

DeRose, J., Wechsler, R. H., Tinker, J. L., et al. 2018, ArXiv e-prints, arXiv:1804.05865

DESI Collaboration, Aghamousa, A., Aguilar, J., et al. 2016, arXiv e-prints, arXiv:1611.00036

Desjacques, V., Jeong, D., & Schmidt, F. 2018, Phys. Rep., 733, 1

Diemand, J., Moore, B., Stadel, J., & Kazantzidis, S. 2004, MNRAS, 348, 977

DLMF. ????, *NIST Digital Library of Mathematical Functions*, http://dlmf.nist.gov/, Release 1.0.22 of 2019-03-15, f. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller and B. V. Saunders, eds.

Dubinski, J., Kim, J., Park, C., & Humble, R. 2004, New Astron., 9, 111

Eastwood, J. W., & Hockney, R. W. 1974, Journal of Computational Physics, 16, 342

Eastwood, J. W., Hockney, R. W., & Lawrence, D. N. 1980, Computer Physics Communications, 19, 215

Efstathiou, G., Davis, M., White, S. D. M., & Frenk, C. S. 1985, The Astrophysical Journal Supplement Series, 57, 241

Efstathiou, G., & Eastwood, J. W. 1981, MNRAS, 194, 503

Efstathiou, G., Frenk, C. S., White, S. D. M., & Davis, M. 1988, MNRAS, 235, 715

Einstein, A. 1917, Sitzungsber. Preuss. Akad. Wiss. Berlin (Math. Phys.), 1917, 142

Eisenstein, D. J., & Hu, W. 1998, ApJ, 496, 605

Eisenstein, D. J., Seo, H.-J., Sirko, E., & Spergel, D. N. 2007, ApJ, 664, 675

Eisenstein, D. J., Zehavi, I., Hogg, D. W., et al. 2005, ApJ, 633, 560

El-Zant, A. A. 2006, MNRAS, 370, 1247

Emberson, J. D., Yu, H.-R., Inman, D., et al. 2017, Research in Astronomy and Astrophysics, 17, 085

Ewald, P. P. 1921, Annalen der Physik, 369, 253

Feng, Y., Chu, M.-Y., Seljak, U., & McDonald, P. 2016, MNRAS, 463, 2273

Fidler, C., Rampf, C., Tram, T., et al. 2015, Phys. Rev. D, 92, 123517

Fischer, P., & Tyson, J. A. 1997, AJ, 114, 14

Foreman, S., Perrier, H., & Senatore, L. 2016, Journal of Cosmology and Astro-Particle Physics, 2016, 027

Frieman, J., & Dark Energy Survey Collaboration. 2013, in American Astronomical Society Meeting Abstracts, Vol. 221, American Astronomical Society Meeting Abstracts #221, 335.01

Garrison, L. H., & Eisenstein, D. J. 2019, MNRAS, 485, 2407

Garrison, L. H., Eisenstein, D. J., Ferrer, D., Metchnik, M. V., & Pinto, P. A. 2016, MNRAS, 461, 4125

Garrison, L. H., Eisenstein, D. J., Ferrer, D., et al. 2018, The Astrophysical Journal Supplement Series, 236, 43

Geller, M. J., & Huchra, J. P. 1983, The Astrophysical Journal Supplement Series, 52, 61

Green, S. R., & Wald, R. M. 2012, Phys. Rev. D, 85, 063512

Gunn, J. E. 1967, ApJ, 147, 61

Guo, Q., White, S., Li, C., & Boylan-Kolchin, M. 2010, MNRAS, 404, 1111

Habib, S., Morozov, V., Frontiere, N., et al. 2013, in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '13 (New York, NY, USA: ACM), 6:1–6:10

Habib, S., Pope, A., Finkel, H., et al. 2016, New Astron., 42, 49

Hada, R., & Eisenstein, D. J. 2018, MNRAS, 478, 1866

Hahn, O., Abel, T., & Kaehler, R. 2013, MNRAS, 434, 1171

Hahn, O., & Angulo, R. E. 2016, MNRAS, 455, 1115

Hamilton, A. J. S. 1998, in Astrophysics and Space Science Library, Vol. 231, The Evolving Universe, ed. D. Hamilton, 185

Hand, N., Feng, Y., Beutler, F., et al. 2018, AJ, 156, 160

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

Heitmann, K., Higdon, D., White, M., et al. 2009, ApJ, 705, 156

Heitmann, K., Lukić, Z., Fasel, P., et al. 2008, Computational Science and Discovery, 1, 015003

Heitmann, K., Bingham, D., Lawrence, E., et al. 2016, ApJ, 820, 108

Hernandez, D. M., & Bertschinger, E. 2015, MNRAS, 452, 1934

Hernquist, L., & Katz, N. 1989, ApJS, 70, 419

Hildebrandt, H., Viola, M., Heymans, C., et al. 2017, MNRAS, 465, 1454

Hinshaw, G., Larson, D., Komatsu, E., et al. 2013, ApJS, 208, 19

Hobson, M., Efstathiou, G., & Lasenby, A. 2006, General Relativity: An Introduction for Physicists (Cambridge University Press)

Hockney, R. W., & Eastwood, J. W. 1981, Computer Simulation Using Particles (McGraw-Hill)

—. 1988, Computer simulation using particles (Hilger)

Holmberg, E. 1941, ApJ, 94, 385

Hu, W. 1999, ApJ, 522, L21

Hut, P., & Rees, M. J. 1992, MNRAS, 259, 27P

Jackson, J. C. 1972, MNRAS, 156, 1P

Jarvis, M. 2015, TreeCorr: Two-point correlation functions, Astrophysics Source Code Library, ascl:1508.007

Jenkins, A. 2010, MNRAS, 403, 1859

Jeong, D. 2010, PhD thesis, University of Texas at Austin

Jeong, D., Schmidt, F., & Hirata, C. M. 2012, Phys. Rev. D, 85, 023504

Jing, Y. P. 2005, ApJ, 620, 559

Jones, E., Oliphant, T., Peterson, P., et al. 2001–, SciPy: Open source scientific tools for Python, [Online; accessed <today>]

Joyce, M., & Marcos, B. 2007a, Phys. Rev. D, 76, 103505

—. 2007b, Phys. Rev. D, 75, 063516

Joyce, M., Marcos, B., & Baertschiger, T. 2009, MNRAS, 394, 751

Joyce, M., Marcos, B., Gabrielli, A., Baertschiger, T., & Sylos Labini, F. 2005, Physical Review Letters, 95, 011304

Jungman, G., Kamionkowski, M., & Griest, K. 1996, Phys. Rep., 267, 195

Kaiser, N. 1987, MNRAS, 227, 1

Kereš, D., Katz, N., Weinberg, D. H., & Davé, R. 2005, MNRAS, 363, 2

Kilbinger, M. 2015, Reports on Progress in Physics, 78, 086901

Klypin, A., & Prada, F. 2017, ArXiv e-prints, arXiv:1701.05690

Klypin, A., Prada, F., & Comparat, J. 2017, ArXiv e-prints, arXiv:1711.01453

Klypin, A. A., & Shandarin, S. F. 1983, MNRAS, 204, 891

Klypin, A. A., Trujillo-Gomez, S., & Primack, J. 2011, ApJ, 740, 102

Knebe, A., Pearce, F. R., Lux, H., et al. 2013, MNRAS, 435, 1618

Kravtsov, A. V., Berlind, A. A., Wechsler, R. H., et al. 2004, ApJ, 609, 35

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

Laureijs, R. 2009, ArXiv e-prints, arXiv:0912.0914

Laureijs, R., Amiaux, J., Arduini, S., et al. 2011, ArXiv e-prints, arXiv:1110.3193

Lawrence, E., Heitmann, K., White, M., et al. 2010, ApJ, 713, 1322

Lawrence, E., Heitmann, K., Kwan, J., et al. 2017, ArXiv e-prints, arXiv:1705.03388

Leauthaud, A., Saito, S., Hilbert, S., et al. 2017, MNRAS, 467, 3024

Lemson, G., & Virgo Consortium, t. 2006, ArXiv Astrophysics e-prints, astro-ph/0608019

Lesgourgues, J. 2011, arXiv e-prints, arXiv:1104.2932

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

Lewis, A., Challinor, A., & Lasenby, A. 2000, Astrophys. J., 538, 473

L'Huillier, B., Park, C., & Kim, J. 2014, New Astron., 30, 79

Lissauer, J. J., Ragozzine, D., Fabrycky, D. C., et al. 2011, The Astrophysical Journal Supplement Series, 197, 8

Liu, J., Bird, S., Zorrilla Matilla, J. M., et al. 2018, Journal of Cosmology and Astro-Particle Physics, 2018, 049

LSST Dark Energy Science Collaboration. 2012, ArXiv e-prints, arXiv:1211.0310

LSST Science Collaboration, Abell, P. A., Allison, J., et al. 2009, ArXiv e-prints, arXiv:0912.0201

Madau, P., Meiksin, A., & Rees, M. J. 1997, ApJ, 475, 429

Mandelbaum, R., Hirata, C. M., Seljak, U., et al. 2005, MNRAS, 361, 1287

Mansfield, P., & Kravtsov, A. V. 2019, arXiv e-prints, arXiv:1902.00030

Marcos, B., Baertschiger, T., Joyce, M., Gabrielli, A., & Sylos Labini, F. 2006, Phys. Rev. D, 73, 103507

Markevitch, M., Gonzalez, A. H., Clowe, D., et al. 2004, ApJ, 606, 819

McBride, C., Berlind, A., Scoccimarro, R., et al. 2009, in Bulletin of the American Astronomical Society, Vol. 41, American Astronomical Society Meeting Abstracts #213, 253

McClintock, T., Rozo, E., Becker, M. R., et al. 2018, ArXiv e-prints, arXiv:1804.05866

McKinney, W. 2010, in Proceedings of the 9th Python in Science Conference, 51–56

Mead, A. J., & Peacock, J. A. 2014, Monthly Notices of the Royal Astronomical Society, 440, 1233

Mead, A. J., Peacock, J. A., Heymans, C., Joudaki, S., & Heavens, A. F. 2015, MNRAS, 454, 1958

Metchnik, M. V. L. 2009, PhD thesis, The University of Arizona

Mohammed, I., Martizzi, D., Teyssier, R., & Amara, A. 2014, ArXiv e-prints, arXiv:1410.6826

Monaco, P. 2016, Galaxies, 4, 53

Morales, M. F., & Wyithe, J. S. B. 2010, Annual Review of Astronomy and Astrophysics, 48, 127

More, S., Kravtsov, A. V., Dalal, N., & Gottlöber, S. 2011, The Astrophysical Journal Supplement Series, 195, 4

Nelder, J. A., & Mead, R. 1965, The Computer Journal, 7, 308

Oort, J. H. 1983, Annual Review of Astronomy and Astrophysics, 21, 373

Padmanabhan, N., & White, M. 2009, Phys. Rev. D, 80, 063508

Padmanabhan, N., Xu, X., Eisenstein, D. J., et al. 2012, MNRAS, 427, 2132

Percival, W. J., & White, M. 2009, MNRAS, 393, 297

Perlmutter, S., Aldering, G., Goldhaber, G., et al. 1999, ApJ, 517, 565

Perryman, M. A. C., Brown, A. G. A., Lebreton, Y., et al. 1998, A&A, 331, 81

Pines, D. 1964, Elementary excitations in solids: lectures on phonons, electrons, and plasmons, Vol. 5 (WA Benjamin)

Planck Collaboration. 2016, A&A, doi:10.1051/0004-6361/201525830

Planck Collaboration, Ade, P. A. R., Aghanim, N., et al. 2014, A&A, 571, A16

Planck Collaboration, Aghanim, N., Akrami, Y., et al. 2018, arXiv e-prints, arXiv:1807.06209

Plummer, H. C. 1911, MNRAS, 71, 460

Potter, D., Stadel, J., & Teyssier, R. 2016, ArXiv e-prints, arXiv:1609.08621

Potter, D., Stadel, J., & Teyssier, R. 2017, Computational Astrophysics and Cosmology, 4, 2

Powell, M. J. D. 1964, The Computer Journal, 7, 155

Power, C., Robotham, A. S. G., Obreschkow, D., Hobbs, A., & Lewis, G. F. 2016, MNRAS, 462, 474

Preskill, J., Wise, M. B., & Wilczek, F. 1983, Physics Letters B, 120, 127

Quinn, T., Katz, N., Stadel, J., & Lake, G. 1997, arXiv e-prints, astro

Reddick, R. M., Wechsler, R. H., Tinker, J. L., & Behroozi, P. S. 2013, ApJ, 771, 30

Reed, D. S., Smith, R. E., Potter, D., et al. 2013, MNRAS, 431, 1866

Rein, H., & Spiegel, D. S. 2015, MNRAS, 446, 1424

Rein, H., & Tamayo, D. 2015, MNRAS, 452, 376

Riebe, K., Partl, A. M., Enke, H., et al. 2013, Astronomische Nachrichten, 334, 691

Riess, A. G., Casertano, S., Yuan, W., Macri, L. M., & Scolnic, D. 2019, arXiv e-prints, arXiv:1903.07603

Riess, A. G., Filippenko, A. V., Challis, P., et al. 1998, AJ, 116, 1009

Rubin, V. C., Ford, W. K., J., & Thonnard, N. 1980, ApJ, 238, 471

Schaller, M. 2015, PhD thesis, Durham University, UK, doi:10.5281/zenodo.49878

Schmittfull, M., Baldauf, T., & Zaldarriaga, M. 2017, Phys. Rev. D, 96, 023505

Schmittfull, M., Feng, Y., Beutler, F., Sherwin, B., & Chu, M. Y. 2015, Phys. Rev. D, 92, 123522

Schneider, A., & Teyssier, R. 2015, JCAP, 12, 049

Schneider, A., Teyssier, R., Potter, D., et al. 2016a, JCAP, 4, 047

—. 2016b, JCAP, 4, 047

Scoccimarro, R. 1998, MNRAS, 299, 1097

Seo, H.-J., Beutler, F., Ross, A. J., & Saito, S. 2016, MNRAS, 460, 2453

Seo, H.-J., Eckel, J., Eisenstein, D. J., et al. 2010, ApJ, 720, 1650

Shapley, H. 1933, Proceedings of the National Academy of Science, 19, 591

Sheth, R. K., & Tormen, G. 1999, MNRAS, 308, 119

Sinha, M., & Garrison, L. 2017a, Corrfunc: Blazing fast correlation functions on the CPU, Astrophysics Source Code Library, ascl:1703.003

—. 2017b, Corrfunc: Blazing fast correlation functions on the CPU, Astrophysics Source Code Library, ascl:1703.003

Smoot, G. F., Bennett, C. L., Kogut, A., et al. 1992, ApJ, 396, L1

Spergel, D., Gehrels, N., Baltay, C., et al. 2015, ArXiv e-prints, arXiv:1503.03757

Spergel, D. N., Verde, L., Peiris, H. V., et al. 2003, The Astrophysical Journal Supplement Series, 148, 175

Springel, V. 2005, Mon. Not. R. Astron. Soc., 364, doi:10.1111/j.1365-2966.2005.09655.x

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

Springel, V., Pakmor, R., Pillepich, A., et al. 2018, MNRAS, 475, 676

Stone, J. M., Gardiner, T. A., Teuben, P., Hawley, J. F., & Simon, J. B. 2008, The Astrophysical Journal Supplement Series, 178, 137

Takahashi, R., Sato, M., Nishimichi, T., Taruya, A., & Oguri, M. 2012, ApJ, 761, 152

Tanaka, S., Yoshikawa, K., Minoshima, T., & Yoshida, N. 2017, ApJ, 849, 76

Tassev, S., & Zaldarriaga, M. 2012, Journal of Cosmology and Astro-Particle Physics, 2012, 006

Tassev, S., Zaldarriaga, M., & Eisenstein, D. J. 2013, JCAP, 6, 036

Teyssier, R. 2001, Astron. Astrophys., 385, doi:10.1051/0004-6361:20011817

Teyssier, R. 2010, RAMSES: A new N-body and hydrodynamical code, ascl:1011.007

Tinker, J., Kravtsov, A. V., Klypin, A., et al. 2008, ApJ, 688, 709

Tram, T., Brandbyge, J., Dakin, J., & Hannestad, S. 2019, Journal of Cosmology and Astro-Particle Physics, 2019, 022

Troxel, M. A., & Ishak, M. 2015, Phys. Rep., 558, 1

van Daalen, M. P., Schaye, J., Booth, C. M., & Dalla Vecchia, C. 2011, MNRAS, 415, 3649

van der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, Computing in Science and Engineering, 13, 22

Wang, Y. 2008, Journal of Cosmology and Astro-Particle Physics, 2008, 021

Warren, M. S. 2013, ArXiv e-prints, arXiv:1310.4502

Wechsler, R. H., & Tinker, J. L. 2018, Annual Review of Astronomy and Astrophysics, 56, 435

Weinberg, D. H., Mortonson, M. J., Eisenstein, D. J., et al. 2013, Phys. Rep., 530, 87

White, M., Tinker, J. L., & McBride, C. K. 2014, MNRAS, 437, 2594

Wibking, B. D., Salcedo, A. N., Weinberg, D. H., et al. 2019, MNRAS, 484, 989

Yoshikawa, K., Yoshida, N., & Umemura, M. 2013, ApJ, 762, 116

Yuan, S., & Eisenstein, D. J. 2019, MNRAS, 486, 708

Yuan, S., Eisenstein, D. J., & Garrison, L. H. 2018, MNRAS, 478, 2019

Zehavi, I., Zheng, Z., Weinberg, D. H., et al. 2011, ApJ, 736, 59

Zel'dovich, Y. B. 1970, A&A, 5, 84

Zhai, Z., Tinker, J. L., Becker, M. R., et al. 2018, ArXiv e-prints, arXiv:1804.05867

Zhang, H., Eisenstein, D. J., Garrison, L. H., & Ferrer, D. W. 2017, arXiv e-prints, arXiv:1712.05787

Zheng, Z., Coil, A. L., & Zehavi, I. 2007, ApJ, 667, 760

Zheng, Z., Berlind, A. A., Weinberg, D. H., et al. 2005, ApJ, 633, 791

Zhu, H.-M., Yu, Y., Pen, U.-L., Chen, X., & Yu, H.-R. 2017, Phys. Rev. D, 96, 123502