Department of Physics and Astronomy
Aarhus University

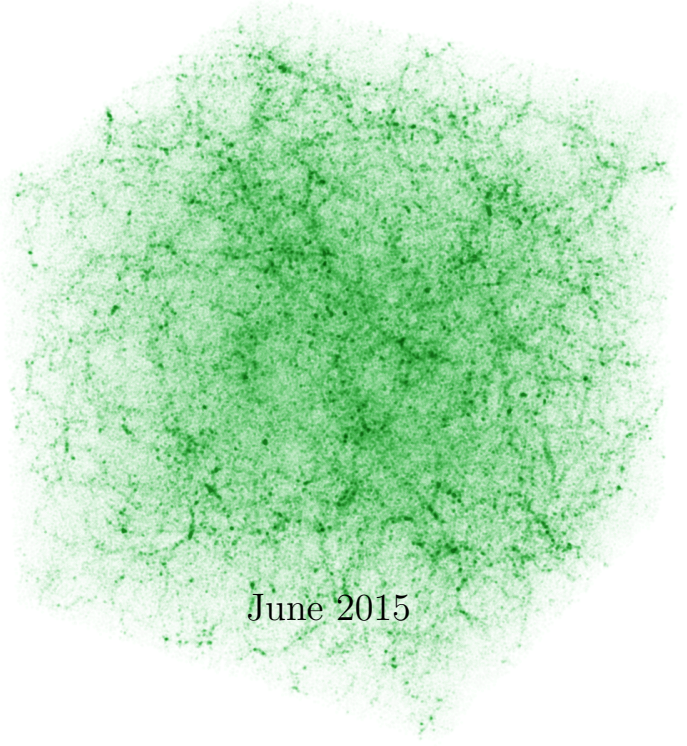
Master's Thesis

Computing Dark Universes

Cosmological *N*-body Simulations of Dark Matter

Author:
Jeppe Mosgaard Dakin

Supervisor:
Steen Hannestad



June 2015

Contents

Summary

The wish to understand the Universe in which we find ourselves has been with us for thousands of years, leading to many imaginative cosmologies — ideas about what the Universe is and how it functions. It is however only within the last century that modern, physical cosmology has emerged, answering some of the big questions. This has been achieved partly by theorization, but to a large degree by observations of the Universe at large scales. Time and time again, scientists have been baffled by the real nature of the Universe. Two of the most outstanding problems in physics today is "what is dark matter?" and "what is dark energy?". As we cannot sense or even instrumentally detect either of these directly, they are beyond our experimental reach, at least for now.

Given a specific cosmological hypothesis, like the exact form of the equation of state for dark energy, say, it is not at all obvious how this hypothesis affects the history of the Universe and results in observable effects. With the advent of powerful modern computers, a new way of testing out such physical hypotheses has emerged. Quite detailed simulations of the evolution of the Universe is now doable, and so we now have a tool for converting hypotheses into simulated observables, which can then be compared to real world observations. One such observable is the large scale structure of the Universe. In so-called N-body simulations, the numerical universe consists of N particles which start out being distributed almost homogeneously. The particles are then evolved through time until we arrive at the present age, at which point the particle distribution (the amount of structure at different scales) can be compared directly with observations. Such cosmological N-body simulations are the main subject of this work.

As part of this work, the full-fledged N-body code CONCEPT has been developed, capable of tracking the evolution of dark matter particles in any cosmology. The methods typically used in N-body codes are discussed, with special attention paid to the methods which actually made it into CONCEPT. In particular, the problem of computing the gravitational forces efficiently along with the problem of accurate time integration, are addressed.

This Master's thesis could be viewed as a self-contained introduction to the field of cosmological N-body simulations. It is my hope that some future version of CONCEPT may one day be used in research.

Introduction

Cosmology, in the broadest sense of the term, refers to our understanding of the *Cosmos* (or Universe); the entirety of all things. In modern times we have arrived at a scientific understanding of the Universe, which constitutes *physical* cosmology. The birth of physical cosmology came with the advent of general relativity a full century ago as of the time of writing. General relativity provides a scene for the cosmic story to unfold, but it does not dictate which story or even which actors should take part in it. Observations tell us that the Universe is homogeneous and isotropic on the very largest of scales, that it is expanding — even accelerating — and that it always has been, since its dawn in the Big Bang 13.8 billion years ago. Observations also tell us that the aforementioned actors — the stuff of the Universe — are stranger than anything previously imagined. In contemporary cosmology, the standard model for the Universe is that of ΛCDM, where the Universe consists of $\sim 70\,\%$ dark energy (Λ^*), $\sim 25\,\%$ cold dark matter (CDM) and just $\sim 5\,\%$ "ordinary" matter. Here, ordinary matter refers to atoms and everything else within the standard model of particle physic, the nature of which we understand intimately. In the context of cosmology, this matter component is often — somewhat imprecise — dubbed *baryonic* matter. The two dark components dominate the Universe, yet we have little to no clue as to the true nature of either. Dark matter appears in observations as invisible matter, intervening gravitationally in the otherwise lonely dance of the luminous baryonic matter, making up the galaxies. Conversely, clustering of dark energy has never been detected. Its effect is seen only on the evolution of the Universe as a whole, where it is responsible for the acceleration of the expansion. For scales sufficiently large, the only important features of the Universe are the universal expansion of space itself as well as the gravitational attraction between matter within it.

Much of what we know today about our Universe come from observations of the cosmic microwave background, together with distant supernovae observations and detailed mappings of the visible large-scale structure. To test cosmological hypotheses, we need a common middle ground between these and the real world. We are thus in need not only of accurate observations, but also accurate predictions of the very same observables as those measured. This

*As in Einstein's cosmological constant.

thesis is concerned with one such tool for generating the closest thing of what might be considered a "universe in the lab", namely cosmological N-body simulations. These simulations emulate the evolution of a given universe, including the stuff within it, rather directly. Given a set of initial conditions consisting of N particles, the future* particle configuration can then be reached by evolving the system according to a given set of physical laws. The field of cosmology (itself borrowing from many branches of physics) is mature enough for these laws to be quite† uniquely established. Creating general purpose N-body codes, where the only freedom in the simulations are given by‡ the values of a set of parameters, is then both meaningful and doable. A given hypothesis about the Universe (the precise amounts of dark matter and energy, say) can then be tested by running the corresponding simulation and comparing the results to the observed large-scale structure of the Universe.

This thesis revolves around the§ CONCEPT (COsmological N-body CodE in PyThon) code, developed by the author as part of this work. Although the physics is well established, the best way of implementing it into N-body codes is not universally agreed upon. The CONCEPT code is heavily inspired by the publicly available GADGET-2 [1] code. Throughout the thesis, different N-body methods are first established generally, after which a more specific implementation is considered. This specific implementation always coincide with that used in the CONCEPT code. At the time of writing, this code is fully functional but lacks the superb numerical efficiency (due to the use of simpler algorithms) of well-established modern codes, as well as "additional" physics (e.g. baryons). When appropriate, these more advanced methods used by modern codes are outlined. This thesis might then be thought of as a self-contained beginners guide to N-body codes, introducing everything needed to construct a working N-body code, capable of discerning different cosmologies.

The technical software perspective, involving how to actually write or even just run N-body codes, is left out of the main body of this thesis. A user guide for the CONCEPT code has however been written and is included with the source code. It is the authors strong¶ opinion that the CONCEPT code is superior to more established codes when it comes to ease of usability and further development. The code is well-structured and documented, but most importantly written in the modern Cython programming language, as opposed to one of the classical "scientific" languages (e.g. C, Fortran). Cython code is essentially Python code — famous for its expressiveness and dynamic abilities, resulting in fast development — compiled down to lower level C code, which

*Or the past for that matter.

†Alternatives to even the most established physics (e.g. modified gravity) do exist but are considered exotic.

‡And of course by different initial conditions.

§The source code is made publicly available at https://github.com/jmd-dk/concept/.

¶And, naturally, quite biased.

in turn is famous for its speed. A measure of the expressiveness of the Python language is the ratio of number of lines of code in the Python source code of CONCEPT and the equivalent C source code, generated by Cython. As of the time of writing, the CONCEPT code consists of 5×10^3 lines of Python code, which are then translated* into 150×10^3 lines of equivalent C code.

Before diving into any work revolving around N-body codes, a brief review of some jargon is advised. The particles of the simulation lives within a cubic, periodic *box* which constitutes the simulated universe. A file containing the total state of the system (the positions and velocities of all N particles) at any given time is referred to as a *snapshot*. The initial snapshot constituting the initial conditions is then called the *initial condition file*. Besides the initial conditions, a particular simulation is defined by the values of a set of parameters, controlling the cosmology, physics and numerical schemes and accuracies of the simulation. A file containing the collective set of parameter values is called a *parameter file*. The user-defined input to each simulation is then an initial condition file and a parameter file. A particular instance of a simulation is referred to as a *run*.

The thesis is structured as follows. In chapter 1, the basic equations of cosmology, governing a homogeneous and isotropic universe, are established. These general relativistic equations are then distilled down to the Newtonian approximation in comoving coordinates. Fluctuations in the otherwise homogeneous matter distribution are then introduced using linear perturbation theory, from which we learn quantitatively about the behavior of structure formation in the Universe. It is made clear that to solve for the evolution of the matter fluctuations beyond just the early Universe, N-body techniques are needed. With the N-body approach motivated, the basic constituents of any N-body code, the gravity solver and the time evolution, are developed in chapters 2 and 3, respectively. Without much consideration, chapter 2 sets out to construct the most naïve gravity solver imaginable. As flaws and inabilities become apparent, they are corrected for in a very much exploratory manner. After the completion of this first gravity solver, more elaborate and efficient gravitational solvers are developed. Along the way, the different gravity solvers implemented in the CONCEPT code are compared to one another, and to the gravitational solver of GADGET-2. Chapter 3 deals with formal as well as practical time evolution of N-body systems, which requires greater consideration than what one might think. Finally, the work is reviewed and concluded in chapter 4.

*This is the result of the Cython translation. If done by hand, the Python to C line count ratio would still be impressive, though not quite as much. Both line counts exclude comments.

1 Cosmology

In this chapter we review modern cosmology, deriving the basic equations governing the homogeneous and isotropic universe. These are then distilled down to Newtonian mechanics in comoving coordinates. We then perturb this universe with density fluctuations and attempt to solve for the evolution of the density field, resulting in structure formation. We will discover that analytical methods are unable to trace this structure formation except at early times, where linearization is possible. This failure of analytical methods then pave the way for numerical N-body simulations.

A geometrized system of units is used throughout this chapter, with Newton's constant and the speed of light both set equal to unity, $G = 1 = c$. In future chapters we shall give up this convention.

1.1 The Expanding Universe

In this section we shall derive the basic equations governing the evolution of the flat universe.

1.1.1 The Metric

The study of the Universe as a whole begins not with its contents, but rather with its geometry. The detailed geometry of spacetime is complicated, reflecting the distribution of all things embedded within it, as described by the Einstein field equations. On very large scales though, observations tell us that the contents of the Universe appears very isotropic. Disregarding the odd change of us being located at a very special location in the Universe, this observation implies universal large-scale homogeneity. On large scales then, all spatial locations are exactly equivalent. In particular, the local curvature is constant throughout space. The only* manifolds with this property are hyperbolic, Euclidean and elliptical manifolds. The line elements $\mathrm{d}\Sigma$ of these manifolds may all be written in polar coordinates as

$$\mathrm{d}\Sigma^2 = \frac{\mathrm{d}\boldsymbol{x}^2}{1 - k\boldsymbol{x}^2} + \boldsymbol{x}^2\,\mathrm{d}\Omega^2\,,$$

*Tori are disallowed as these break isotropy.

where $\boldsymbol{x}$ is the Cartesian coordinate, labeling the points on the manifold and so $\boldsymbol{x}^2$ is the radial distance squared. As usual, the solid angular element is $\mathrm{d}\Omega^2 = \mathrm{d}\theta^2 + \sin^2\theta\,\mathrm{d}\phi^2$, where θ and ϕ are the polar and azimuthal angle, respectively. The constant k is the universal curvature of the manifold. The three possible types of manifolds are then distinguished by the sign of k; negative (hyperbolic), zero (Euclidean) or positive (elliptical). Introducing time t as a coordinate perpendicular to $\boldsymbol{x}$, the spacetime line element $\mathrm{d}s^2$ becomes*

$$\mathrm{d}s^2 = -\,\mathrm{d}t^2 + a^2(t)\,\mathrm{d}\Sigma^2\,, \tag{1.1}$$

where the spatial part of the line element has been multiplied by a time dependent function $a(t)$ called the scale factor, included for generality. The isotropy requirement can only be fulfilled in one particular frame of reference, where all velocities (if any) appear radial. Homogeneity now implies that such observed radial velocities must be the same regardless of vantage point. These radial velocities can then only be ascribed to space itself expanding or contracting homogeneously, which is made possible by the scale factor a. The distance $|\boldsymbol{x}|$ does then not correspond to a physical distance, as the expansion of space need to be factored in. The coordinates $\boldsymbol{x}$ is called comoving coordinates, while $a\boldsymbol{x}$ are the physical coordinates. The particular frame containing only these radial velocities is called the comoving frame. As velocities superimposed on top of the universal expansion are disallowed by isotropy, the content of the Universe is stationary in the comoving frame.

The metric (1.1) is the Friedmann–Lemaître–Robertson–Walker (FLRW) metric, the most general *spatially* homogeneous and isotropic spacetime metric. The question of the sign of k is an experimental one. The best current estimates of cosmological parameters such as k come from combining observations of the anisotropies in the cosmic microwave background with baryon acoustic oscillations and the Hubble constant. As found in e.g. [2], our Universe seems exceptionally flat, corresponding to k^{-1} being much larger than the cosmic horizon. As our goal is to study our particular Universe, we allow ourselves to limit our study to flat models only:

$$\mathrm{d}s^2 = -\,\mathrm{d}t^2 + a^2(t)\big(\mathrm{d}\boldsymbol{x}^2 + \boldsymbol{x}^2\,\mathrm{d}\Omega^2\big)\,. \tag{1.2}$$

The metric tensor $g_{\mu\nu}$ for this line element, corresponding to† $\mathrm{d}s^2 = g_{\mu\nu}\,\mathrm{d}x^\mu\,\mathrm{d}x^\nu$ with $x^\mu = (t, |\boldsymbol{x}|, \theta, \phi)$, is then

$$g_{\mu\nu} = \mathrm{diag}\big(-1,\, a^2,\, a^2\boldsymbol{x}^2,\, a^2\boldsymbol{x}^2\sin^2\theta\big)\,. \tag{1.3}$$

*Here the metric signature is chosen to be spacelike; $(-+++)$. We shall stick with this convention throughout this chapter.

†As is customary in general relativity, repeated indices implies summation when they appear in both the co- and contravariant form. Also, Greek letter indices run over all four spacetime coordinates, while Latin letter indices run over spatial coordinates only.

Apart from the as of yet unspecified scale factor $a(t)$, the geometry of our Universe is now established. The remaining unspecified piece of information is the nature of the homogeneous and isotropic contents within the Universe, on which $a(t)$ is allowed to depend.

1.1.2 Basic Equations of Cosmology

The connection between the metric of the Universe — and therefore $a(t)$ — and the stuff within it is given by the Einstein field equations,

$$R_{\mu\nu} - \frac{1}{2} g_{\mu\nu} R = 8\pi T_{\mu\nu} \,. \tag{1.4}$$

where $T_{\mu\nu}$ is the stress-energy tensor while $R_{\mu\nu}$ and R are the Ricci curvature tensor and scalar, respectively. A cosmological constant is deliberately left out from the left-hand side, as we shall think of this as a component — dark energy — to the stress-energy tensor $T_{\mu\nu}$ on the right-hand side. The Ricci tensor, written in terms of the Christoffel symbols, is given by

$$R_{\mu\nu} = \Gamma^{\alpha}{}_{\mu\nu,\alpha} - \Gamma^{\alpha}{}_{\mu\alpha,\nu} + \Gamma^{\alpha}{}_{\mu\nu}\Gamma^{\beta}{}_{\alpha\beta} - \Gamma^{\alpha}{}_{\mu\beta}\Gamma^{\beta}{}_{\nu\alpha} \,, \tag{1.5}$$

where indices following a comma imply differentiation. The Christoffel symbols themselves are given as derivatives of the metric tensor,

$$\Gamma^{\mu}{}_{\nu\rho} = \frac{1}{2} g^{\alpha\mu} \big(g_{\alpha\nu,\rho} + g_{\alpha\rho,\nu} - g_{\nu\rho,\alpha} \big) \,, \tag{1.6}$$

where $g^{\mu\nu}$ is the inverse of $g_{\mu\nu}$, which for the diagonal FLRW metric (1.2) can be calculated element-wise; $g^{\mu\mu} = 1/g_{\mu\mu}$. Through (1.5) and (1.6) the Ricci tensor can now be calculated. These are straight forward but tedious calculations, and so here we simply state the (non-zero) results:

$$\begin{cases} \Gamma^{0}{}_{11} = a\dot{a} \,, & \Gamma^{0}{}_{22} = a\dot{a}\boldsymbol{x}^2 \,, \qquad \Gamma^{0}{}_{33} = a\dot{a}\boldsymbol{x}^2 \sin^2\theta \,, \\ \Gamma^{i}{}_{0i} = \dfrac{\dot{a}}{a} \,, & \\ \Gamma^{1}{}_{22} = -|\boldsymbol{x}| \,, & \Gamma^{1}{}_{33} = -|\boldsymbol{x}| \sin^2\theta \,, \\ \Gamma^{2}{}_{12} = \Gamma^{3}{}_{13} = |\boldsymbol{x}|^{-1} \,, & \\ \Gamma^{2}{}_{33} = -\sin\theta\cos\theta \,, & \Gamma^{3}{}_{23} = \cot\theta \,, \end{cases}$$

$$\Rightarrow \begin{cases} R_{00} = -3\dfrac{\ddot{a}}{a} \,, \\ R_{11} = \big(a\ddot{a} + 2\dot{a}^2\big) \,, \\ R_{22} = \big(a\ddot{a} + 2\dot{a}^2\big)\boldsymbol{x}^2 \,, \\ R_{33} = \big(a\ddot{a} + 2\dot{a}^2\big)\boldsymbol{x}^2 \sin^2\theta \,. \end{cases}$$

It should be remembered that the Christoffel symbol is symmetric in the lower indices, as is evident from (1.6) together with the symmetric metric. The metric (1.3) then produces a total of 16 Christoffel symbols. Raising an index on the Ricci tensor, $R^{\mu}{}_{\nu} = g^{\alpha\mu}R_{\nu\alpha}$, we get the cleaner form

$$\begin{cases} R^{0}{}_{0} = 3\dfrac{\ddot{a}}{a}\,, \\ R^{i}{}_{j} = \left(\dfrac{\ddot{a}}{a} + 2\dfrac{\dot{a}^2}{a^2}\right)\delta^{i}{}_{j}\,, \end{cases} \tag{1.7}$$

where $\delta^{i}{}_{j}$ is the Kronecker delta tensor. From (1.7), the Ricci scalar, the trace of the Ricci tensor, is then

$$\begin{aligned} R &\equiv R^{\alpha}{}_{\alpha} \\ &= 3\frac{\ddot{a}}{a} + 3\left(\frac{\ddot{a}}{a} + 2\frac{\dot{a}^2}{a^2}\right) \\ &= 6\left(\frac{\ddot{a}}{a} + \frac{\dot{a}^2}{a^2}\right), \end{aligned} \tag{1.8}$$

which completes the left-hand-side of the Einstein field equations (1.4).

We now define the stress-energy tensor $T^{\mu\nu}$ appearing on the right-hand-side of the Einstein field equations (1.4). This tensor describes the density and flux of energy and momentum in spacetime. As described in the previous subsection, the contents of the Universe is stationary in comoving coordinates, and so no momentum or shear stress is present; $T^{0i} = 0$, $T^{ij} \propto \delta^{ij}$. We are left with the energy density T^{00} and the pressure T^{ii} components. Requiring that the ii-components be the isotropic pressure $\bar{P}/a^2$ in Cartesian coordinates, we have for our choice of spherical coordinates

$$T^{\mu\nu} = \mathrm{diag}\left(\rho_{\mathrm{c}}, \frac{\bar{P}}{a^2}, \frac{\bar{P}}{a^2\boldsymbol{x}^2}, \frac{\bar{P}}{a^2\boldsymbol{x}^2\sin^2\theta}\right), \tag{1.9}$$

where $T^{00} = \rho_{\mathrm{c}}$ amounts to nothing more than a relabeling of the energy density. The bars indicate spatial averaging, which of course does not make a difference in the case of exact homogeneity. As we intend to perturb the Universe later, the bars are necessary to distinguish fields from their mean values. The subscript on the density indicates that this is not to be understood as a field either, but as the *critical* density, exactly the value of the homogeneous density which leads to a flat universe. We shall refer to the content of the Universe, fully described by $T^{\mu\nu}$, as a fluid.

As it stands, (1.9) is not manifestly covariant and so it is not clear that $T^{\mu\nu}$ is indeed a tensor. We can however compose the result (1.9) out of the metric tensor and the tensor $u^{\mu}u^{\nu}$, where u^{μ} is the four-velocity of the fluid:

$$T^{\mu\nu} = (\rho_{\mathrm{c}} + \bar{P})u^{\mu}u^{\nu} + \bar{P}g^{\mu\nu}\,, \tag{1.10}$$

where $u^\mu = (1, 0, 0, 0)$ for the fluid, stationary in the comoving frame. With a proper tensor expression (1.10) for $T^{\mu\nu}$, we can lower an index to find the simplified expression

$$\begin{aligned} T^\mu{}_\nu &= (\rho_c + \bar{P})u^\mu u_\nu + \bar{P} g^\mu{}_\nu \\ &= (\rho_c + \bar{P})u^\mu u_\nu + \bar{P}\delta^\mu{}_\nu \\ &= \mathrm{diag}(-\rho_c, \bar{P}, \bar{P}, \bar{P})\,, \end{aligned} \tag{1.11}$$

where the minus sign is due to $u_0 = g_{\alpha 0}u^\alpha = g_{00}u^0 = -1 \times 1$.

With both sides of the Einstein field equations at hand, let us put them together to find the equations relating the expansion of the Universe to its contents. Raising an index in the field equations (1.4) and inserting (1.7), (1.8) and (1.11), the 00-component becomes

$$\frac{\dot{a}^2}{a^2} = \frac{8\pi}{3}\rho_c\,, \tag{1.12}$$

which is the first Friedmann equation, relating the expansion of the Universe to the mean density. In a similar manner, the ii-components of the Einstein field equations all produce

$$\begin{aligned} 2\frac{\ddot{a}}{a} + \frac{\dot{a}^2}{a} &= -8\pi\bar{P} \\ \Rightarrow \frac{\ddot{a}}{a} &= -\frac{4\pi}{3}(\rho_c + 3\bar{P})\,, \end{aligned} \tag{1.13}$$

where (1.12) have been used to eliminate $\dot{a}$. Equation (1.13) is then the second Friedmann equation. Differentiation (1.12) with respect to time and inserting (1.13), we now get

$$\begin{aligned} \dot{\rho_c} &= \frac{3}{4\pi}\frac{\dot{a}}{a}\left(\frac{\ddot{a}}{a} - \frac{\dot{a}^2}{a^2}\right) \\ &= -3\frac{\dot{a}}{a}(\rho_c + \bar{P})\,, \end{aligned} \tag{1.14}$$

which is really the continuity equation, expressing conservation of energy. The same result may be found by requiring that the covariant derivative of the stress-energy tensor vanishes, $T^{\mu\nu}{}_{;\nu} = 0$.

It is fortunate that both the density ρ and the pressure P are additive entities when dealing with multi-component universes, containing several, different fluids. All fluids with interest to us have an equation of state of the form

$$P = w\rho\,, \tag{1.15}$$

with some constant w inherent to each type of fluid. For such a multi-component universe, we then have

$$\rho_c = \sum_s \bar{\rho}_s \qquad \bar{P} = \sum_s \bar{P}_s \tag{1.16}$$

$$= \sum_s w_s \bar{\rho}_s \,,$$

where the sums are over all fluids present within the Universe. The left equality of (1.16), with the critical density defined in (1.12), amounts to the criterion of flatness. Inserting (1.16) in the continuity equation (1.14), we see that it must hold separately for each fluid. With the equation of state (1.15), the continuity equation may be integrated to

$$\bar{\rho} = \frac{\bar{\rho}_0}{a^{3(1+w)}} \,, \tag{1.17}$$

where $\bar{\rho} = \bar{\rho}(a)$ is the mean density of some fluid, its present value being $\bar{\rho}_0 = \bar{\rho}(a = 1)$. Using (1.16) and (1.17), the first Friedmann equation (1.12) can now be written

$$\begin{aligned} \frac{H}{H_0} &= \sqrt{\sum_s \Omega_s} \\ &= \sqrt{\sum_s \frac{\Omega_{s,0}}{a^{3(1+w_s)}}} \,, \end{aligned} \tag{1.18}$$

where $H = \dot{a}/a$ is the Hubble parameter and $\Omega \equiv \bar{\rho}/\rho_\mathrm{c}$ the dimensionless density parameter, where again, subscripts s and 0 refer to the fluid type and the present, respectively.

Let us introduce the specific fluid types of our Universe. Nonrelativistic *matter* is used as a collective name for all pressureless fluids, (cold) dark and baryonic matter being the primary examples. From the equation of state (1.15), pressureless means $w = 0$, and so the matter density parameter is $\Omega_\mathrm{m} \propto a^{-3}$. In the Friedmann equation (1.18), no distinction is made between dark and baryonic matter, and so these have identical effects on the Universe as a whole. In the other end of the spectrum, we have *radiation*, corresponding to relativistic matter, of which photons and neutrinos are prime examples. Radiation has vanishing rest mass, and so all of its energy density is constituted by its momentum density. As the momentum is distributed isotropically, the pressure to energy density ratio is $w = 1/3$; $\Omega_\mathrm{r} \propto a^{-4}$. The most important component of our present Universe is dark energy, the nature of which is largely unknown. The simplest candidate to dark energy is a cosmological constant, corresponding to a constant energy density with an associated negative pressure. To achieve constant energy density $\Omega_\Lambda \propto 1$, we must have $w = -1$. From the second Friedmann equation (1.13), we see that dark energy is the only of these three components which have a positive contribution to the acceleration $\ddot{a}$ of the expansion.

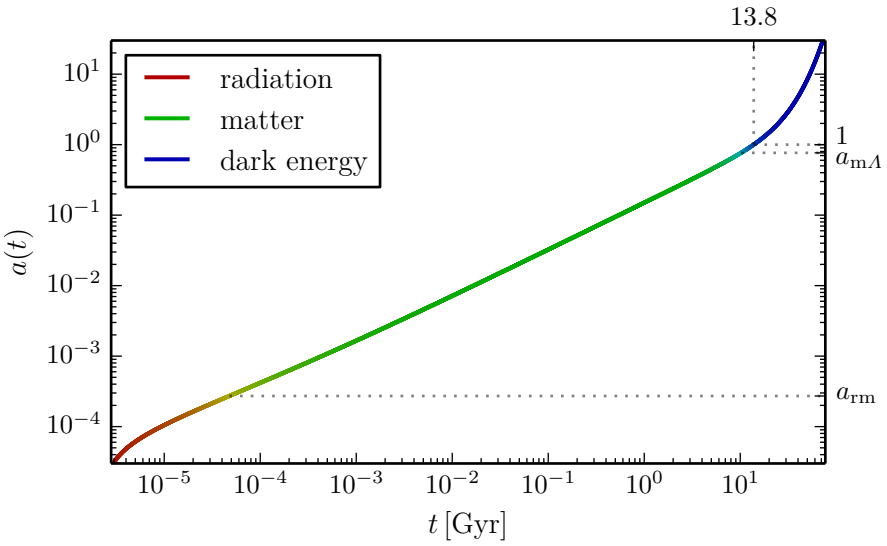


Figure 1.1 – The evolution of the scale factor $a(t)$ in our three-component Universe (1.19), from the early Universe to the present $a(t_0) = 1 \Rightarrow t_0 = 13.8\,\mathrm{Gyr}$ and beyond. The different equations of state for the components lead to density domination at different epochs. The color of the line reflects the relative densities.

The best determined values [2] of the present cosmological parameters* are

$$\begin{cases} \Omega_{\Lambda,0} = 0.691 \pm 0.006\,, \\ \Omega_{\mathrm{m},0} = 0.309 \pm 0.006\,, \\ \Omega_{\mathrm{r},0} \approx 8.4 \times 10^{-5}\,, \\ H_0 = (67.7 \pm 0.5)\,\mathrm{km\ s^{-1}Mpc^{-1}}\,. \end{cases} \tag{1.19}$$

Our Universe with these three components then has the following Friedmann equation:

$$\frac{H}{H_0} = \sqrt{\Omega_{\Lambda,0} + \frac{\Omega_{\mathrm{m},0}}{a^3} + \frac{\Omega_{\mathrm{r},0}}{a^4}}\,. \tag{1.20}$$

It is clear that at early times, radiation was the dominating component. At intermediate times, the Universe then becomes matter dominated, after which it becomes dominated by dark energy. At the present epoch (1.19), dark energy is well under way to total domination. At each domination epoch, two components are all we need to accurately model the Universe. As we are concerned with the present Universe, we thus wish to neglect radiation. To find out when this is possible, we calculate the value of a at which radiation and matter are equally important:

$$\Omega_{\mathrm{r}} = \Omega_{\mathrm{m}}$$

*The radiation density is not included in [2]. The one listed is from [3].

$$\begin{aligned} \Rightarrow a_{\mathrm{rm}} &= \frac{\Omega_{\mathrm{r},0}}{\Omega_{\mathrm{m},0}} \\ &= 2.7 \times 10^{-4}\,, \end{aligned}$$

where the observational data (1.19) have been used. At times a somewhat larger than a_{rm}, radiation can safely be ignored. Similarly, matter/dark energy equivalence occur at $a_{\mathrm{m}\Lambda}$,

$$\begin{aligned} \Omega_{\mathrm{m}} &= \Omega_{\Lambda} \\ \Rightarrow a_{\mathrm{m}\Lambda} &= \left(\frac{\Omega_{\mathrm{m},0}}{\Omega_{\Lambda,0}}\right)^{1/3} \\ &= 0.76\,. \end{aligned}$$

Figure 1.1 shows the evolution of the scale factor as given by the Friedmann equation (1.20) with the values of the cosmological parameters corresponding to (1.19). The separate domination epochs are clearly visible as the color of the line, but more importantly as the expansion rate. For complete domination, these rates can easily be calculated using the single-component Friedmann equation, $\dot{a} = H_0 a^{-\frac{1}{2}(1+3w)}$. For domination by each of the three fluid types, we get

$$\dot{a} = \begin{cases} H_0 a^{-1} & \text{(radiation)} \\ H_0 a^{-1/2} & \text{(matter)} \\ H_0 a & \text{(dark energy)} \end{cases} \tag{1.21}$$

$$\Rightarrow a = \begin{cases} (2H_0 t)^{1/2} & \text{(radiation)} \\ \left(\frac{3}{2} H_0 t\right)^{2/3} & \text{(matter)} \\ C\mathrm{e}^{H_0 t} & \text{(dark energy),} \end{cases} \tag{1.22}$$

with C being some integration constant. These growth rates match what we see in Figure 1.1.

1.2 Newtonian Cosmology

We wish to find the Newtonian limit of the cosmology presented above, reducing the complexity of further analysis greatly. For an almost homogeneous universe, the gravitational field is very weak, which justifies this approximation, at least on sub-horizon scales.

1.2.1 The Newtonian Limit of General Relativity

The Newtonian limit of general relativity is defined by small velocities together with weak and slowly changing gravitational fields. In the limit of no velocities

at all, corresponding to a matter dominated universe ($\bar{P} = 0$), only the rest mass contribute to the stress-energy tensor, and so from (1.11),

$$T_{00} = \rho\,, \qquad T_{\mu i} = 0\,, \tag{1.23}$$

where ρ is the almost homogeneous matter density field. This reduces the 10 independent equations of (1.4) to just a single one, as is the case in Newtonian gravity. In the end we wish to approximate gravitation as being Newtonian, but maintain the overall effect of general relativity, namely the background expansion. Locally*, the FLRW metric is just the Minkowski metric of special relativity, which in Cartesian coordinates are just:

$$\eta_{\mu\nu} = \mathrm{diag}(-1, 1, 1, 1)\,.$$

Our approach is then to adopt the Minkowski metric as the "Newtonian metric", and then later add in the expansion through a change to comoving variables. Using a completely flat Minkowski metric $\eta_{\mu\nu}$ amounts to no gravity at all. To allow for a weak gravitational field, we therefore use a perturbed Minkowski metric for the Newtonian universe;

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu}\,, \quad |h_{\mu\nu}| \ll 1\,,$$

where the dynamics of spacetime, corresponding to gravitation, is encoded purely in the perturbation $h_{\mu\nu}$.

We wish to exploit the very simple stress-energy tensor (1.23) of the Newtonian universe. Instead of having the trace R of the Ricci tensor appearing in the Einstein field equations (1.4), we can "trace-reverse" these equations to produce a form where the trace $T = -\rho$ of the stress-energy tensor appears instead of R. To do this, take the trace of (1.4), leaving the scalar equation $-R = 8\pi T$, where $g^{\mu\nu}g_{\mu\nu} = \delta^{\mu}{}_{\mu} = 4$ is used. Now add this times $-g_{\mu\nu}/2$ to (1.4) to obtain

$$R_{\mu\nu} = 8\pi\left(T_{\mu\nu} - \frac{1}{2}g_{\mu\nu}T\right).$$

Since the 00 component of the stress-energy tensor is the only one which does not vanish, this reduces to

$$R_{00} \approx 4\pi\rho\,, \tag{1.24}$$

where only leading orders has been kept; $g_{00} \approx \eta_{00}$.

From (1.5), the 00 component of the Ricci tensor is given by

$$\begin{aligned} R_{00} &= \Gamma^{\alpha}{}_{00,\alpha} - \Gamma^{\alpha}{}_{\alpha 0,0} + \Gamma^{\alpha}{}_{\alpha\beta}\Gamma^{\beta}{}_{00} - \Gamma^{\alpha}{}_{0\beta}\Gamma^{\beta}{}_{\alpha 0}\,, \\ &\approx \Gamma^{\alpha}{}_{00,\alpha}\,, \end{aligned}$$

*Local is here used in the temporal sense, removing explicit time dependence of the metric.

where all temporal derivatives are neglected. The two products of Christoffel symbols then vanish due to direct temporal differentiation (1.6) of the metric. Also from (1.6), the one remaining term is given by

$$\begin{aligned}
\Gamma^{\alpha}{}_{00,\alpha} &= \frac{1}{2}\partial_\alpha\left[g^{\beta\alpha}\left(g_{\beta 0,0} + g_{\beta 0,0} - g_{00,\beta}\right)\right], \\
&\approx -\frac{1}{2}\partial_\alpha\left[\eta^{\beta\alpha}h_{00,\beta}\right] \\
&= -\frac{1}{2}h_{00}{}^{,\alpha}{}_{,\alpha} \\
&= \frac{1}{2}\nabla^2 h_{00}\,,
\end{aligned} \tag{1.25}$$

where temporal derivatives again are neglected, along with second order terms in $h_{\mu\nu}$. If we now choose $h_{00} = 2\phi$ where ϕ is the Newtonian gravitational potential and substitute R_{00} in (1.24) for (1.25), we get

$$\nabla^2\phi = 4\pi\rho\,, \tag{1.26}$$

which is of course Poisson's equation for Newtonian gravity. The manual association of h_{00} to 2ϕ cannot be avoided, as a conversion between the metric and the Newtonian gravitational field does not emerge naturally from general relativity, where such a field is not defined. To check whether this association also gives consistent equations of motion, we consider the geodesic equation. Because we are working in the Newtonian limit, we are free to use coordinate time t as the affine parameter:

$$\begin{aligned}
\frac{\mathrm{d}^2x^\mu}{\mathrm{d}t^2} &= -\Gamma^{\mu}{}_{\alpha\beta}\frac{\mathrm{d}x^\alpha}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t} \\
&\approx -\Gamma^{\mu}{}_{00} \\
&\approx \begin{cases} 0 & \mu = 0\,, \\ -\dfrac{\mathrm{d}\phi}{\mathrm{d}x^\mu} & \mu \in \{1,2,3\}\,, \end{cases}
\end{aligned} \tag{1.27}$$

where low velocities $\mathrm{d}x^\mu/\,\mathrm{d}t \approx (1,0,0,0)$ is used. The temporal equation is just the trivial statement $\mathrm{d}^2t/\,\mathrm{d}t^2 = 0$. More interestingly, the spatial equations state that the acceleration is the negative gradient of the potential. We have therefore successfully regained Newtonian gravitation from the equations of general relativity.

1.2.2 Expanding Space

The above analysis used the (perturbed) Minkowski metric to describe the geometry of a universe. To model the real Universe, the universal expansion must be taken into account, at least at large scales. We can use Newtonian gravitation as derived above, and then add in the expansion by hand. To do

this we simply change from proper $x^i = \boldsymbol{r}$ to comoving $\boldsymbol{x}$ coordinates. As seen from the FLRW metric, these coordinate systems are related simply by

$$\boldsymbol{r} = a\boldsymbol{x}\,. \tag{1.28}$$

In comoving coordinates the Hubble flow is factored out, meaning that the separation between points comoving with the Hubble flow remains fixed despite of the expansion. To see this, differentiate (1.28) with respect to cosmic time t:

$$\begin{aligned}\dot{\boldsymbol{r}} &= \dot{a}\boldsymbol{x} + a\dot{\boldsymbol{x}} \\ &\equiv H\boldsymbol{r} + \boldsymbol{u}\,,\end{aligned}$$

where the peculiar velocity $\boldsymbol{u} \equiv a\dot{\boldsymbol{x}}$ has been introduced. Thus the proper velocity has contributions both from the Hubble flow and the peculiar velocity. The peculiar velocity $\boldsymbol{u}$ is hence the velocity of a fluid element when viewed in the comoving frame.

The Newtonian equation of motion (1.27) may now be written in the conventional way,

$$\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = -\nabla_{\boldsymbol{r}}\phi\,,$$

where the subscript on the del operator is used to signify that the differentiation is with respect to proper coordinates. This additional labeling is necessary when dealing with multiple coordinate systems. From (1.28), it follows that $\nabla_{\boldsymbol{x}} = a\nabla_{\boldsymbol{r}}$. With this additional level of care, let us rewrite the Poisson equation (1.26) as

$$\nabla^2_{\boldsymbol{r}}\phi = 4\pi\rho\,. \tag{1.29}$$

1.3 Newtonian Perturbation Theory

With gravity established as a Newtonian force, we can perturb the otherwise perfectly homogeneous density field ρ. The evolution of this field is then governed by Newtonian gravity, as well as of the expanding background, the only general relativistic effect kept from the previous analysis. As neither radiation nor dark energy* participate in clustering, we shall from now on think of ρ as the *matter* density field, specifically.

1.3.1 Comoving Fluid Equations

As matter is modelled as a fluid, continuum Newtonian mechanics should be used to evolve ρ in time. Besides the Poisson equation, we are thus in need of

*This is not a settled question, though very little clustering is allowed by observations.

the two Euler equations, representing the conservation of mass and momentum, respectively:

$$\begin{cases} \left.\dfrac{\partial\rho}{\partial t}\right|_{\boldsymbol{r}} + \nabla_{\boldsymbol{r}}\cdot(\rho\dot{\boldsymbol{r}}) = 0\,, \\ \left.\dfrac{\partial\dot{\boldsymbol{r}}}{\partial t}\right|_{\boldsymbol{r}} + \dot{\boldsymbol{r}}\cdot\nabla_{\boldsymbol{r}}\dot{\boldsymbol{r}} = -\nabla_{\boldsymbol{r}}\phi\,, \end{cases} \tag{1.30}$$

where the only force on the fluid is the gravitational force $-\nabla_{\boldsymbol{r}}\phi$, as the fluid is without pressure (and therefore a pressure gradient). The temporal derivatives are also given the subscript $\boldsymbol{r}$, to remind us that it is the proper coordinates which are kept fixed under the differentiation. As the scalefactor a in $\boldsymbol{r} = a\boldsymbol{x}$ is time dependent, ∂_t cannot constrain $\boldsymbol{r}$ and $\boldsymbol{x}$ simultaneously, and so the specification $\partial_t|_{\boldsymbol{r}}$ is needed.

To factor in the expansion of the Universe, the Poisson equation (1.29) as well as the Euler equations (1.30) need to be written in comoving coordinates. We can take care of this transformation for the Poisson equation right away. To do this, define* the peculiar potential

$$\varphi(\boldsymbol{x}) \equiv a\phi(a\boldsymbol{x}) + \frac{1}{2}a^2\ddot{a}\boldsymbol{x}^2\,, \tag{1.31}$$

the job of which is simply to produce a nice looking Poisson equation in the comoving frame. Now using the proper Poisson equation (1.29), we have

$$\begin{aligned} \nabla^2_{\boldsymbol{x}}\varphi(\boldsymbol{x}) &= a\nabla^2_{\boldsymbol{x}}\phi(a\boldsymbol{x}) + \frac{1}{2}a^2\ddot{a}\nabla^2_{\boldsymbol{x}}\boldsymbol{x}^2 \\ &= a^3\nabla^2_{\boldsymbol{r}}\phi(\boldsymbol{r}) + 3a^2\ddot{a} \\ &= a^3\left(4\pi\rho(\boldsymbol{x}) + 3\frac{\ddot{a}}{a}\right) \\ &= 4\pi a^3(\rho(\boldsymbol{x}) - \bar{\rho}) \\ &= 4\pi\big(\varrho(\boldsymbol{x}) - \bar{\varrho}\big)\,, \end{aligned} \tag{1.32}$$

where the the second Friedmann equation (1.13) has been used to replace $\ddot{a}$. In the last equality, $\varrho = a^3\rho$ is the comoving density. The resultant Poisson equation (1.32) is that of the (comoving) density fluctuations, not the density itself. This can only amount to a shift in the potential, and so has no physical effects. It turns out that it is actually necessary to consider the fluctuations only, as the ordinary Poisson equation has no solution† in an infinite universe.

*In the literature, the peculiar potential is most often defined to be $\phi(a\boldsymbol{x}) + a\ddot{a}\boldsymbol{x}^2$, corresponding to φ/a. This difference in convention is of no importance.

†We shall tackle the mathematics of this degeneracy of the Newtonian potential in chapter 2. The physical intuition behind it is that normally in classical mechanics, the potential is always implicitly assumed to vanish at infinity, corresponding to having a finite system. In general relativity, the problem never arises due to the speed of gravity being finite.

Before converting the Euler equations (1.30) to comoving coordinates, we need to take a closer look at the temporal derivatives used in these equations. Consider a function $f(\boldsymbol{r}, t)$, its differential being

$$\begin{aligned} \mathrm{d}f &= \nabla_{\boldsymbol{r}} f(\boldsymbol{r},t)\cdot \mathrm{d}\boldsymbol{r} + \left.\frac{\partial f(\boldsymbol{r},t)}{\partial t}\right|_{\boldsymbol{r}} \mathrm{d}t \\ &= \nabla_{\boldsymbol{x}} f(\boldsymbol{r},t)\cdot \mathrm{d}\boldsymbol{x} + \left[\frac{\dot{a}}{a}\boldsymbol{x}\cdot\nabla_{\boldsymbol{x}} f(\boldsymbol{r},t) + \left.\frac{\partial f(\boldsymbol{r},t)}{\partial t}\right|_{\boldsymbol{r}}\right]\mathrm{d}t\,, \end{aligned} \tag{1.33}$$

where $\nabla_{\boldsymbol{x}} = a\nabla_{\boldsymbol{r}}$ and $\mathrm{d}\boldsymbol{r} = \boldsymbol{x}\,\mathrm{d}a + a\,\mathrm{d}\boldsymbol{x} = \dot{a}\boldsymbol{x}\,\mathrm{d}t + a\,\mathrm{d}\boldsymbol{x}$ as been used. Allowing the function to alternatively take in comoving coordinates as arguments, $f(\boldsymbol{r}, t) = f(a\boldsymbol{x}, t) \equiv f(\boldsymbol{x}, t)$, its differential in comoving coordinates become

$$\mathrm{d}f = \nabla_{\boldsymbol{x}} f(\boldsymbol{x},t)\cdot \mathrm{d}\boldsymbol{x} + \left.\frac{\partial f(\boldsymbol{x},t)}{\partial t}\right|_{\boldsymbol{x}} \mathrm{d}t\,. \tag{1.34}$$

By setting he right-hand-sides of (1.33) and (1.34) equal to each other, dividing by $\mathrm{d}t$ and removing the function, we obtain the operator equation

$$\left.\frac{\partial}{\partial t}\right|_{\boldsymbol{r}} = \left.\frac{\partial}{\partial t}\right|_{\boldsymbol{x}} - \frac{\dot{a}}{a}\boldsymbol{x}\cdot\nabla_{\boldsymbol{x}}\,, \tag{1.35}$$

giving us the relation between temporal derivatives in proper and comoving coordinates. With (1.35), we can now write the first Euler equation of (1.30) in comoving coordinates:

$$\begin{aligned} 0 &= \left[\left.\frac{\partial\rho}{\partial t}\right|_{\boldsymbol{x}} - \frac{\dot{a}}{a}\boldsymbol{x}\cdot\nabla_{\boldsymbol{x}}\rho\right] + \frac{1}{a}\big(\rho\nabla_{\boldsymbol{x}}\cdot\dot{\boldsymbol{r}} + \dot{\boldsymbol{r}}\cdot\nabla_{\boldsymbol{x}}\rho\big) \\ &= \left.\frac{\partial\rho}{\partial t}\right|_{\boldsymbol{x}} - \overbrace{\frac{\dot{a}}{a}\boldsymbol{x}\cdot\nabla_{\boldsymbol{x}}\rho + \frac{\dot{a}}{a}\boldsymbol{x}\nabla_{\boldsymbol{x}}\rho}^{0} + \frac{\dot{a}}{a}\rho\overbrace{\nabla_{\boldsymbol{x}}\cdot\boldsymbol{x}}^{3} + \frac{1}{a}\overbrace{\big(\rho\nabla_{\boldsymbol{x}}\cdot\boldsymbol{u} + \boldsymbol{u}\cdot\nabla_{\boldsymbol{x}}\rho\big)}^{\nabla_{\boldsymbol{x}}\cdot(\rho\boldsymbol{u})} \\ &= \left.\frac{\partial\rho}{\partial t}\right|_{\boldsymbol{x}} + 3\frac{\dot{a}}{a}\rho + \frac{1}{a}\nabla_{\boldsymbol{x}}\cdot(\rho\boldsymbol{u})\,, \end{aligned} \tag{1.36}$$

the first two terms of which we recognize as precisely the continuity equation (1.14) for a homogeneous, pressureless fluid in an expanding universe, while the third is just the usual velocity divergence term, now in comoving coordinates.

The second Euler equation in (1.30) in comoving coordinates, again using (1.35), becomes

$$\begin{aligned} 0 = \left(\left.\frac{\partial}{\partial t}\right|_{\boldsymbol{x}} - \frac{\dot{a}}{a}\boldsymbol{x}\cdot\nabla_{\boldsymbol{x}}\right)(\boldsymbol{u}+\dot{a}\boldsymbol{x}) + \frac{1}{a}(\boldsymbol{u}+\dot{a}\boldsymbol{x})\cdot\nabla_{\boldsymbol{x}}(\boldsymbol{u}+\dot{a}\boldsymbol{x}) \\ + \frac{1}{a^2}\left(\nabla_{\boldsymbol{x}}\varphi - \frac{1}{2}a^2\ddot{a}\nabla_{\boldsymbol{x}}\boldsymbol{x}^2\right) \end{aligned}$$

$$
\begin{aligned}
&= \frac{\partial}{\partial t}\bigg|_{\boldsymbol{x}} (\boldsymbol{u} + \dot{a}\boldsymbol{x}) - \overbrace{\frac{\dot{a}}{a}\boldsymbol{x}\cdot\nabla_{\boldsymbol{x}}(\boldsymbol{u}+\dot{a}\boldsymbol{x}) + \frac{\dot{a}}{a}\boldsymbol{x}\cdot\nabla_{\boldsymbol{x}}(\boldsymbol{u}+\dot{a}\boldsymbol{x})}^{0} \\
&\qquad + \frac{1}{a}(\boldsymbol{u}\cdot\nabla_{\boldsymbol{x}}\boldsymbol{u} + \dot{a}\underbrace{\boldsymbol{u}\cdot\nabla_{\boldsymbol{x}}\boldsymbol{x}}_{\boldsymbol{u}}) + \frac{1}{a^2}\nabla_{\boldsymbol{x}}\varphi - \ddot{a}\boldsymbol{x} \\
&= \dot{\boldsymbol{u}} + \ddot{a}\boldsymbol{x} + \dot{a}\overbrace{\frac{\partial \boldsymbol{x}}{\partial t}\bigg|_{\boldsymbol{x}}}^{0} + \frac{1}{a}\boldsymbol{u}\cdot\nabla_{\boldsymbol{x}}\boldsymbol{u} + \frac{\dot{a}}{a}\boldsymbol{u} + \frac{1}{a^2}\nabla_{\boldsymbol{x}}\varphi - \ddot{a}\boldsymbol{x} \\
&= \dot{\boldsymbol{u}} + \frac{1}{a}\boldsymbol{u}\cdot\nabla_{\boldsymbol{x}}\boldsymbol{u} + \frac{\dot{a}}{a}\boldsymbol{u} + \frac{1}{a^2}\nabla_{\boldsymbol{x}}\varphi\,, \qquad (1.37)
\end{aligned}
$$

witch completes out set of equations governing the evolution of ρ in comoving coordinates. As we are interested to small perturbations of ρ around $\bar{\rho}$, let us define the density contrast δ as

$$\delta = \frac{\rho - \bar{\rho}}{\bar{\rho}}\,.$$

In terms of δ, the Poisson equation (1.32) is

$$\nabla^2_{\boldsymbol{x}}\varphi = 4\pi a^3 \delta\bar{\rho}\,, \qquad (1.38)$$

while the first Euler equation (1.36) is

$$
\begin{aligned}
0 &= \frac{\partial\bar{\rho}(\delta+1)}{\partial t}\bigg|_{\boldsymbol{x}} + 3\frac{\dot{a}}{a}\bar{\rho}(\delta+1) + \frac{1}{a}\bar{\rho}\nabla_{\boldsymbol{x}}\cdot([\delta+1]\boldsymbol{u}) \\
&= \dot{\bar{\rho}}\delta + \bar{\rho}\dot{\delta} + \dot{\bar{\rho}} + \frac{1}{a}\bar{\rho}\nabla_{\boldsymbol{x}}([\delta+1]\boldsymbol{u}) \\
&= \bar{\rho}\dot{\delta} + \frac{1}{a}\bar{\rho}\nabla_{\boldsymbol{x}}([\delta+1]\boldsymbol{u})\,, \qquad (1.39)
\end{aligned}
$$

where $\bar{\rho} = \bar{\rho}_0/a^3 \Rightarrow \dot{\bar{\rho}} = -3(\dot{a}/a)\bar{\rho}$ has been used. The density, and therefore δ, does not appear in the momentum conservation equation (the second Euler equation (1.30)). Equations (1.38), (1.39) and (1.37) then constitutes the full set of equations governing the evolution of the density perturbations δ. These coupled differential equations are generally not easily solved. Let us state the perturbation criteria as $\delta \ll 1$, $\boldsymbol{u}\cdot\nabla_{\boldsymbol{x}}\boldsymbol{u} \approx 0$, in which case the equations reduces to the more manageable set

$$
\begin{cases}
\nabla^2_{\boldsymbol{x}}\varphi = 4\pi a^3\delta\bar{\rho} \\
\dot{\delta} + \dfrac{1}{a}\nabla_{\boldsymbol{x}}\cdot\boldsymbol{u} = 0 \\
\dot{\boldsymbol{u}} + \dfrac{\dot{a}}{a}\boldsymbol{u} + \dfrac{1}{a^2}\nabla_{\boldsymbol{x}}\varphi = \boldsymbol{0}\,.
\end{cases}
$$

We can get rid of the velocity field $\boldsymbol{u}$ by combining the temporal derivative of the first Euler equation with the spatial $(a^{-1}\nabla_{\boldsymbol{x}}\cdot)$ derivative of the second

Euler equation

$$\begin{aligned} 0 &= \frac{\partial}{\partial t}\left(\dot{\delta} + \frac{1}{a}\nabla_{\boldsymbol{x}}\boldsymbol{u}\right) - \frac{1}{a}\nabla_{\boldsymbol{x}}\left(\dot{\boldsymbol{u}} + \frac{\dot{a}}{a}\boldsymbol{u} + \frac{1}{a^2}\nabla_{\boldsymbol{x}}\varphi\right) \\ &= \left(\ddot{\delta} - \frac{\dot{a}}{a^2}\nabla_{\boldsymbol{x}}\boldsymbol{u} + \frac{1}{a}\nabla_{\boldsymbol{x}}\dot{\boldsymbol{u}}\right) - \frac{1}{a}\left(\nabla_{\boldsymbol{x}}\dot{\boldsymbol{u}} + \frac{\dot{a}}{a}\nabla_{\boldsymbol{x}}\boldsymbol{u} + \frac{1}{a^2}\nabla_{\boldsymbol{x}}^2\phi\right) \\ &= \ddot{\delta} + 2\frac{\dot{a}}{a}\dot{\delta} - 4\pi\delta\bar{\rho}\,, \end{aligned} \tag{1.40}$$

where the Poisson equation has been used to replace the Laplacian of the potential with the density fluctuations. Equation (1.40) is the final result of the linearization of the fluid equations. The only assumption used has been that of almost homogeneity — almost homogeneity of ρ so that it is perturbative, and almost (or complete) homogeneity of everything else, allowing the use of Newtonian mechanics embedded in an expanding background, as laid out in section 1.1. Equation (1.40) is therefore valid for the matter component within any cosmology that complies with this assumption.

1.3.2 Matter Fluctuations

We can solve (1.40) analytically for each domination epoch, resulting in the qualitative behavior of the growth of the perturbations. Let us first consider the case of matter domination. From (1.21) and (1.22), we have $\dot{a}/a = H_0 a^{-3/2} = 2/(3t)$, while since $\bar{\rho} = \rho_{\mathrm{c}}$, we have $\bar{\rho} = 1/(6\pi t^2)$ from (1.12). The linear density perturbation equation (1.40) then becomes

$$0 = \ddot{\delta} + \frac{4}{3t}\dot{\delta} - \frac{2}{3t^2}\delta\,,$$

which have solutions

$$\begin{aligned} \delta &= C_1 t^{2/3} + C_2 t^{-1} \\ &= C_3 a + C_4 a^{-3/2}\,. \end{aligned}$$

The decaying mode of the density contrast will eventually vanish, leaving only the growing mode. The primary result of the linear perturbation theory is then

$$\delta \propto a \qquad \text{(matter domination)}.$$

The density perturbations within a matter dominated universe then grows linearly with the scale factor, leading to clustering of matter.

Now we consider the radiation dominated era, preceding matter domination. Here we have $\dot{a}/a = H_0 a^{-2} = 1/(2t)$ from (1.21) and (1.22), and so (1.40) becomes

$$0 = \ddot{\delta} + \dot{\delta}t^{-1} - 4\pi\bar{\rho}\delta\,, \tag{1.41}$$

where the density perturbations cannot easily be written in terms of t, as it requires solving the first Friedmann equation with $\rho_{\mathrm{c}} = \bar{\rho}_{\mathrm{r}} + \bar{\rho}$, where the radiation density is much larger than the matter density, $\bar{\rho}_{\mathrm{r}} \gg \bar{\rho}$. Were we to insert the solution δ for the matter dominated Universe into (1.41), we would get $0 = Ct^{-4/3} - 4\pi\bar{\rho}t^{2/3}$, where at early times corresponding to radiation domination, the term containing $\bar{\rho}$ can be ignored. Because the expansion rate is larger for radiation domination than for matter domination, we expect less clustering at this early era. The approximation of neglecting the $\bar{\rho}$ term of (1.41) is then even more sound for the actual radiation dominated case. We are left with $0 = \ddot{\delta} + \dot{\delta}t^{-1}$, which have solutions

$$\begin{aligned} \delta &= C_1 \log(H_0 t) + C_2 \\ &= C_3 \log(a) + C_2 \end{aligned} \qquad \text{(radiation domination).}$$

Because the matter density perturbations grow only logarithmically in the radiation dominated phase, no significant structure is formed before matter begins to dominate. Linear perturbation theory is then fully capable of evolving the matter density field through the radiation era.

Remaining is the qualitative growth rate of the matter density in the era dominated by dark energy. From (1.21), we have $\dot{a}/a = H_0$. Comparing the square* of the front factor of the middle term in (1.40) to the front factor of the last term, we have $H_0^2 \sim \rho_{\mathrm{c}} \gg \bar{\rho}$, which means that we once again are allowed to neglect the last term. We then have $0 = \ddot{\delta} + 2H_0\dot{\delta}$, which have as the solution

$$\begin{aligned} \delta &= C_1 \mathrm{e}^{-2H_0 t} + C_2 \\ &= C_1 a^{-2} + C_2 \end{aligned} \qquad \text{(dark energy domination).}$$

We then see that matter density fluctuations stop growing when the Universe enters the dark energy dominated phase. Clustering of matter then only occurs between the radiation and the dark energy dominated epochs, while the Universe is matter dominated.

The above analysis was made using Newtonian perturbation theory, and so the results only apply for perturbations on scales much below the horizon. A fully general relativistic analysis would add to the story, though the overall result will still be that sub-horizon matter perturbations grow as $\delta \propto a$ in the matter dominated phase, while no growth occurs during the radiation and the dark energy epochs.

1.3.3 The *N*-body Approach

The linear perturbation theory have given us qualitative hints about the evolution of structure in the Universe. We even found linear perturbation

*A direct comparison — without squaring — cannot be done due to the dimensionality of the terms.

theory to be fully adequate for the radiation dominated epoch. Given some primordial* density fluctuations, numerical codes can then be used to solve the (relativistic) linear differential equations governing the matter fluctuations. An example of such a code is CAMB [4]. These computations are often done in Fourier space, and so the outputs are $\delta(\boldsymbol{k})$, the Fourier transform of $\delta = \delta(\boldsymbol{x})$. One often quantifies these in terms of the power spectrum $P(\boldsymbol{k})$, which is the Fourier transform of the two-point correlation function $\overline{\delta(\boldsymbol{x}_1)\delta(\boldsymbol{x}_2)}$, where the extended bar indicates spatial averaging:

$$P(\boldsymbol{k}) = \left| \int \overline{\delta(\boldsymbol{x}_1)\delta(\boldsymbol{x}_2)} \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\boldsymbol{x}} \, \mathrm{d}\boldsymbol{x} \right|^2 , \tag{1.42}$$

where for large-scale isotropy $P(\boldsymbol{k}) = P(|\boldsymbol{k}|)$. The value of $|P(|\boldsymbol{k}|)|$ then indicates the amount of structure at the scale $1/|\boldsymbol{k}|$.

During matter domination and beyond, the perturbations grow as $\delta \propto a$, making the density evolution highly non-linear during the matter domination epoch. Perturbation theory is then of no use, and direct numerical simulation of the system is the only† option. To approach these simulations, introduce the phase-space distribution function $f(\boldsymbol{x}, \boldsymbol{u}, t)$, where $f(\boldsymbol{x}, \boldsymbol{u}, t) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{u}$ is the probability density for finding a fluid element within $[\boldsymbol{x} \pm \mathrm{d}x, \boldsymbol{u} \pm \mathrm{d}\boldsymbol{u}]$. The matter density is then

$$\rho \propto \int f \, \mathrm{d}\boldsymbol{u} \, .$$

By sampling the phase space distribution function at N points in $(\boldsymbol{x}, \boldsymbol{u})$ space, the overall statistical behavior of the system is retained with an error proportional to $N^{-1/2}$. If we do this via the Lagrangian‡ approach, it corresponds to tracking the phase space trajectories of N particles through time, which is the basic idea of N-body simulations, the main subject of this work. At the heart of it, an N-body simulation is then nothing more than a Monte Carlo simulation of the density field.

The distribution function f must be constant along the trajectory $[\boldsymbol{x}(t), \boldsymbol{u}(t)]$ followed by a particle, according to the interpretation of $f(\boldsymbol{x}, \boldsymbol{u}, t) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{u}$. The evolution of the phase-space distribution function is then governed by

$$0 = \frac{\mathrm{d}f(\boldsymbol{x}, \boldsymbol{u}, t)}{\mathrm{d}t}$$

*The observed inhomogeneities in the cosmic microwave background are used to probe these early fluctuations. It is believed that the "original" fluctuations are the results of quantum effects during the inflation phase of the very early Universe.

†Formalisms like that of Press and Schechter [5] is another option, not involving direct simulation. In the end however, these formalisms are only justified by their agreement with the results of N-body approaches.

‡Here, Lagrangian refers to following the phase-space fluid elements through time, as opposed to fixing $\boldsymbol{x}$ and $\boldsymbol{u}$ and see how $f(\boldsymbol{x}, \boldsymbol{u}, t)|_{\boldsymbol{x},\boldsymbol{u}}$ changes through time, referred to as the Eulerian approach.

$$= \frac{\partial f(\boldsymbol{x}, \boldsymbol{u}, t)}{\partial t} + \dot{\boldsymbol{r}} \frac{\partial f(\boldsymbol{x}, \boldsymbol{u}, t)}{\partial \boldsymbol{r}} + \dot{\boldsymbol{u}} \frac{\partial f(\boldsymbol{x}, \boldsymbol{u}, t)}{\partial \boldsymbol{u}} \ , \tag{1.43}$$

which is the (collisionless) Boltzmann equation. Replacing the acceleration $\dot{\boldsymbol{u}}$ with the negative gradient of the gravitational potential, the Boltzmann equation coupled to the Poisson equation now amounts exactly to the Hamiltonian formulation of Newtonian dynamics. By simulating a large number N of particles, each evolved through time according to the Newtonian equations of motion*, we can successfully construct (a granular version of) the density field. Appropriate initial conditions, amounting to N particle positions and velocities, are needed for the N-body simulation. These are generated by perturbing a uniform grid of particles in such a manner that the statistics of the sampled density field equal that of the power spectrum, computed using linear perturbation theory. It is thus important to start the simulation at an early time, so that matter fluctuations are accurately described by linear perturbation theory.

With the N-body approach motivated, the rest of this work focus on the numerical techniques involved in such simulations. We shall focus only on the two most important aspects, namely gravitation and time evolution. This is indeed all there is to say for the system described by the collisionless Boltzmann equation (1.43), and it is also the only physics implemented in the CONCEPT code. Having no collisions is fine for dark matter, which is supposed to be collisionless. Baryonic matter however, act as a fluid with viscosity and a finite sound speed, which in the N-body approach leads to collisions. Although the Friedmann equations do not distinguish between dark and baryonic matter, structure formation does. This is e.g. why the luminous, baryonic part of a spiral galaxy arranges itself into such a complex structure, while the dark matter halo of the galaxy remains spherical, or at least ellipsoidal.

*We shall study the Hamiltonian equations of motion in comoving coordinates in full detail in chapter 3.

2 Newtonian Gravitation

In this chapter we develop the formalism for both describing and computing Newtonian gravitation, in the context of cosmological N-body simulations. The very heart of any cosmological N-body code is its gravitational solver. This is true in the sense that this part constitutes a very large part of the actual code body, but also in the sense that this is where most of the computing time will be spend. This is indeed the way it ought to be, as gravity dominates structure formation at cosmological scales. It is therefore worthwhile going to great lengths developing efficient and accurate methods for computing the gravitational force.

In this chapter we shall study three separate such methods in detail. They are — in order of complexity and usefulness — the particle-particle (PP), the particle-mesh (PM) and the particle-particle-particle-mesh (P^3M) method, respectively. A separate section is dedicated to each method. In the first section, the PP method is developed in an exploratory fashion. The simplest possible gravitational solver is constructed and then build upon, as we discover its flaws and limitations. With a bird's-eye view of what is required by a gravitational solver, we then take a step back and construct the PM method, which is far more applicable than the PP method due to its computational efficiency. While the PP method is virtually exact, this is not true for the PM method. Finally then, we combine what we have learned from the two previous methods and constructs the hybrid P^3M method, which is both accurate and reasonably efficient.

The use of Newtonian gravitation in cosmology differs from its typical use in classical mechanics in two respects. First, the flat Universe — our system of study — contains infinitely many particles, which causes some trouble. Second, and less importantly for now, we wish to describe our particle system in comoving coordinates. As the expansion of the Universe is homogeneous and isotropic (the scale factor depends solely on time), the transformation between the classical force in Euclidean space, and the force given in comoving coordinates in the now expanding, flat space, must be a simple multiplication by the scale factor to some power. We can therefore ignore this last difficulty and pretend that our particles live in static Euclidean space, for now.

2.1 The Particle-Particle Method

In this section we develop the PP method, the simplest fully fledged gravitational solver useful in cosmological N-body simulations. Along the way, many concepts are introduced and analytical results obtained.

The simplest formalism for Newtonian gravitation is the (modern form of the) equation originally put forward by Newton himself, known as Newton's law of universal gravitation. In this formalism, matter consists of particles which interacts pairwise. This is also the formalism used by the PP method, hence the name "particle-particle". We shall therefore begin our study of Newtonian gravitation by deriving and numerically implementing Newton's law of universal gravitation.

2.1.1 Newton's law of Universal Gravitation

We remind ourselves that we now work in Euclidean, static space. Let us denote such spatial coordinates by $\boldsymbol{r}$. In chapter 1 we derived the equivalent Poisson equation for Newtonian gravitation, (1.26),

$$\nabla^2\phi(\boldsymbol{r}) = 4\pi G\rho(\boldsymbol{r})\,, \tag{2.1}$$

where a factor G has been added on, as we no longer work in geometrized units. Newton's law of universal gravitation expresses gravitation in terms of forces, which in the field formulation corresponds to the (negative) gradient of the potential. We thus need to reduce the Laplacian of (2.1) to a gradient, in order to convert it into Newton's law of universal gravitation. We do this by applying the inverse divergence operation to both sides. This will be an integral operator with some vector kernel $\boldsymbol{\mathcal{G}}_\nabla$; the Green's function of the divergence operator:

$$\nabla\phi(\boldsymbol{r}) = 4\pi G\int \boldsymbol{\mathcal{G}}_\nabla(\boldsymbol{r},\boldsymbol{r}')\rho(\boldsymbol{r}')\,\mathrm{d}\boldsymbol{r}'\,, \tag{2.2}$$

where the integral is to be taken over the support of the mass density ρ, or simply all of space. Taking the divergence on both sides of (2.2) should convert it back to (2.1). We therefore have

$$\nabla\cdot\boldsymbol{\mathcal{G}}_\nabla(\boldsymbol{r},\boldsymbol{r}') = \delta(\boldsymbol{r}'-\boldsymbol{r})\,,$$

where $\delta(\boldsymbol{r})$ is the delta function in three dimensions. The solution to the above equation is

$$\boldsymbol{\mathcal{G}}_\nabla(\boldsymbol{r},\boldsymbol{r}') = -\frac{1}{4\pi}\frac{\boldsymbol{r}'-\boldsymbol{r}}{|\boldsymbol{r}'-\boldsymbol{r}|^3}\,. \tag{2.3}$$

Inserting (2.3) into (2.2), we get

$$\nabla\phi(\boldsymbol{r}) = -G\int\frac{\boldsymbol{r}'-\boldsymbol{r}}{|\boldsymbol{r}'-\boldsymbol{r}|^3}\rho(\boldsymbol{r}')\,\mathrm{d}\boldsymbol{r}'\,. \tag{2.4}$$

Equation (2.4) still treats the mass distribution as continuous. We can discretize the mass into point* particles by writing

$$\rho(\boldsymbol{r}) = \sum_{j=1}^{N} m_j \delta(\boldsymbol{r}_j - \boldsymbol{r}),$$

where the sum runs over all N particles j, with individual positions and masses $\boldsymbol{r}_j$ and m_j, respectively. With this discretized mass distribution, (2.4) becomes

$$\nabla\phi(\boldsymbol{r}) = -G \sum_{j=1}^{N} m_j \frac{\boldsymbol{r}_j - \boldsymbol{r}}{|\boldsymbol{r}_j - \boldsymbol{r}|^3}\,. \tag{2.5}$$

The left-hand-side of (2.5) is the negative gravitational/acceleration field at the point $\boldsymbol{r}$, due to all particles j. The acceleration of a specific particle i due to all the others, is then

$$\boldsymbol{F}_i = G m_i \sum_{\substack{j=1 \\ j\neq i}}^{N} m_j \frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^3}\,, \tag{2.6}$$

where the factor m_i comes from Newton's second law. We thus arrive at Newton's law of universal gravitation.

One could consider the removal of the $j = i$ term from the sum in (2.6) as a bit of a cheat, since we have to do it by hand. In the integral of (2.4), the analogues singularity at $\boldsymbol{r}' = \boldsymbol{r}$ is left as is. The problem do occur here as well, but since the integral is antisymmetric around the singularity, any spurious self-forces — though divergent — cancel out, rendering the problem less severe than including the $j = i$ term in the sum. Remembering that the integral in (2.4) is completely invariant under explicit removal of a small region around $\boldsymbol{r}$ in the integration volume, the legitimacy of the exclusion of the $j = i$ term becomes transparent.

Equation (2.6) lends itself extremely easy to numerical implementation. In fact, listing 2.1 shows a complete N-body time loop, utilizing (2.6) as the gravitational solver. A naïve time integration scheme is also shown, for completeness. Listing 2.1 is shown to convince the reader that N-body codes can be very simple indeed. One may think that "real" N-body codes utilizes much more sophisticated algorithms in order to model the real world more accurately. They *are* more sophisticated, not because of a need for better accuracy†, but rather to cut down on the computation time, often at the *expense* of accuracy. As the code in listing 2.1 is written in one-to-one correspondence

*The derivation is equally valid for finite, spherically symmetric particles, as per the shell theorem.

†Here we think exclusively of Newtonian gravity. A closer similarity with the real world can of course be achieved by including more components, e.g. hydrodynamics (baryons).

Listing 2.1 – Naïve but complete time loop of N-body code implementing (2.6), written in Python. Variable names should be self-explanatory.

```
⋮
# Time loop
while not done:
    # Force computation (equation 2.6)
    for i in range(N):
        for j in range(N):
            if j == i:
                continue
            F[i] += (G*m[i]*m[j]*(r[j] - r[i])
                     /norm(r[j] - r[i])**3)
    # Time integration
    p += F*Δt
    r += p/m*Δt
⋮
```

with (2.6), it in fact evolves the system arbitrarily accurately (determined by the size of `Δt`). To a large extent then, the challenge in building N-body codes is not merely to implement the correct physics, but to implement it in a clever way, so that results can be computed within sub-cosmological time scales.

2.1.2 Ewald Summation

We have seen that Newton's law of universal gravitation (2.6) can be easily implemented numerically, e.g. as in listing 2.1. In cosmology however, we seek to describe an infinite* system, which immediately cause problems. If we allow our system to contain infinitely many particles, $N \to \infty$, it would require a correspondingly infinite amount of logical operations (and therefore computing time) to complete even a single time step. In the real world, which is governed by general relativity, particles only influence each other when within each others cosmological horizon. That is, each particle has only ever access to a finite amount of information. One could try to mimic this behaviour in one way or another, in order to get rid of the problem of infinite computing time. This will however only amount to removing the infinity of the inner loop of listing 2.1; every particle will only be influenced by finitely many particles, but we still need to compute the forces on infinitely many particles.

The universal idea utilized in cosmological N-body codes to tackle this problem, is to place $N < \infty$ particles within a cubic box with periodic boundaries. The particles and their gravitational interactions then live on a three-torus. This has the effect of removing the infinity of the outer loop of listing 2.1; we need to compute the force on $N < \infty$ particles, but each

*At least for the cases of flat universes.

particle is still influenced by what is effectively infinitely many particles, as the gravitational interaction is allowed to loop around any of the dimensions of the three-torus an arbitrary amount of times. We are thus still left with one remaining infinity in our computation. However, since this infinity is now periodic, it contains a finite amount of information and can be dealt with in a finite amount of time.

A complementary (but to the three-torus equivalent) interpretation of the manifold on which the particles live, is as an infinite, three-dimensional flat space, the content of which (the N particles) is repeated periodically in all directions. It is clear that the periodicity enforces homogeneity on scales larger than the box size. This is of course not a problem* in cosmology, as long as the box is sufficiently large. Viewing the manifold in this infinite manner ($\boldsymbol{r}$ can take on any value in $\mathbb{R}^3$), the mass density becomes

$$\rho(\boldsymbol{r}) = \sum_{j=1}^{N} m_j \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \delta(\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r})\,, \tag{2.7}$$

where the sum over all integer triples $\boldsymbol{n}$ takes care of the periodic copies of the box with linear size L. We shall refer to the original†, $\boldsymbol{n} = \boldsymbol{0}$ particles simply as particles (or sometimes proper or actual particles), and to their copies as *replicas*. The particles and their replicas are then collectively referred to as *images*.

To compute gravitational forces in this periodic system, we make use of the Ewald summation technique [6], for reasons which will become clear shortly. This technique is easiest to developed on the potential, rather on the force directly. We therefore wish to solve the Poisson equation for the potential. This is achieved in an analogous fashion to what we did in the previous subsection, when calculating the gradient of the potential from the Poisson equation. The corresponding Green's function used as kernel for the integral operator, inverse to the Laplacian, must now solve $\nabla^2\mathcal{G}_{\nabla^2}(\boldsymbol{r},\boldsymbol{r}') = \delta(\boldsymbol{r}'-\boldsymbol{r})$. That is,

$$\mathcal{G}_{\nabla^2}(\boldsymbol{r}) = -\frac{1}{4\pi|\boldsymbol{r}|}\,, \tag{2.8}$$

where $\mathcal{G}_{\nabla^2}(\boldsymbol{r}'-\boldsymbol{r}) = \mathcal{G}_{\nabla^2}(\boldsymbol{r},\boldsymbol{r}')$. The solution to the general Poisson equation (2.1) is then

$$\phi(\boldsymbol{r}) = -G\int \frac{\rho(\boldsymbol{r}')}{|\boldsymbol{r}'-\boldsymbol{r}|}\,\mathrm{d}\boldsymbol{r}'\,. \tag{2.9}$$

*Structure at scales larger than the box size should of course not be given any significance.

†The question of which box to refer to as "the original" is analogous to the question of locating the center of the Universe; it does not matter which you choose, if any. For the purpose of numerical computation though, it is useful to regard the $\boldsymbol{n} = \boldsymbol{0}$ box as *the* box.

Inserting (2.7) into (2.9), we get the potential of the periodic system:

$$\phi(\boldsymbol{r}) = -G\sum_{j=1}^{N} m_j \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \frac{1}{|\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}|} \tag{2.10}$$

$$= 4\pi G\sum_{j=1}^{N} m_j \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \mathcal{G}_{\nabla^2}(\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r})\,. \tag{2.11}$$

The periodic force per unit mass is the negative gradient of the potential, which from (2.10) is given as

$$\nabla\phi(\boldsymbol{r}) = -G\sum_{j=1}^{N} m_j \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \frac{\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}}{|\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}|^3}\,. \tag{2.12}$$

We now need to find a way to handle these infinite sums over images. One way to approximate it would be to simply truncate it. This solution is indeed possible, but it has two drawbacks. The first drawback is the fact that the sum is only conditionally convergent, meaning that its value depends on the order of summation. That is to say, as it stands in (2.10), the sum is not uniquely defined. One thus first have to state the proper summation order, and then apply the truncation in a manner consistent with this order. The second drawback with the truncation proposal is that the sum (even when ordered properly) converges very slowly, rendering the computation numerically infeasible. To tackle both of these drawbacks, the Ewald summation technique may be used.

The Ewald summation technique [6] is a method for computing accurate approximations to infinite sums like that in (2.7). The idea is to split the gravitational interaction in two; a short-range and a long-range component, making two infinite triple-sums instead of one. The short-range sum is then heavily suppressed at large distances, making it converge both absolutely and fast, as opposed to conditionally and slow. The bad convergence behaviour is then solely inherited by the long-range sum, which is suppressed only at small scales. The periodicity of the particle configuration allow us to do the summation in Fourier space, where the small scale suppression means that Fourier components with large $\boldsymbol{k}$ values become negligible. At the same time, the periodicity of the real space means that there exist a smallest non-zero $\boldsymbol{k}$. We can thus obtain absolute and fast convergence for the long-range sum by performing it in Fourier space.

The force split is incorporated by writing the Green's function as a sum of a short-range term $\mathcal{G}_{\mathrm{s}}$ and a long-range term $\mathcal{G}_{\mathrm{l}}$, as $\mathcal{G}_{\mathrm{s}} + \mathcal{G}_{\mathrm{l}} = \mathcal{G}_{\nabla^2}$. We can then rewrite (2.11) using these two contributions. Exploiting the periodicity, we can immediately write the long-range term as a Fourier series. We then

have

$$\phi(\boldsymbol{r}) = 4\pi G \sum_{j=1}^{N} m_j \left[\sum_{\boldsymbol{n}\in\mathbb{Z}^3} \mathcal{G}_{\mathrm{s}}(\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}) + \frac{1}{L^3} \sum_{\boldsymbol{k}\in\left\{\frac{2\pi}{L}\boldsymbol{h}\,|\,\boldsymbol{h}\in\mathbb{Z}^3\right\}} \widetilde{\mathcal{G}}_{\mathrm{l}}(\boldsymbol{k}) \mathrm{e}^{\mathrm{i}\boldsymbol{k}(\boldsymbol{r}_j - \boldsymbol{r})} \right], \quad (2.13)$$

where $\widetilde{\mathcal{G}}_{\mathrm{l}}$ is the Fourier transform of $\mathcal{G}_{\mathrm{l}}$. Our job is now to find suitable expressions for $\mathcal{G}_{\mathrm{s}}$ and $\widetilde{\mathcal{G}}_{\mathrm{l}}$.

Writing $\mathcal{G}_{\mathrm{l}}(\boldsymbol{r}) = \mathcal{G}_{\nabla^2}(\boldsymbol{r})\chi(\boldsymbol{r})$, $\mathcal{G}_{\mathrm{s}}(\boldsymbol{r}) = \mathcal{G}_{\nabla^2}(\boldsymbol{r})\big[1-\chi(\boldsymbol{r})\big]$, wee see that $\chi(\boldsymbol{r})$ should vanish at $\boldsymbol{r}=\boldsymbol{0}$ and tend towards unity as $|\boldsymbol{r}|\to\infty$, to ensure that $\mathcal{G}_{\mathrm{s}}$ and $\mathcal{G}_{\mathrm{l}}$ represent their designated regimes. A particular choice for χ would be the error function, as first proposed by Ewald. Here we follow this choice:

$$\mathcal{G}_{\mathrm{s}}(\boldsymbol{r}) = -\frac{1}{4\pi|\boldsymbol{r}|}\operatorname{erfc}\left(\frac{|\boldsymbol{r}|}{2r_{\mathrm{s}}}\right), \quad (2.14)$$

$$\mathcal{G}_{\mathrm{l}}(\boldsymbol{r}) = -\frac{1}{4\pi|\boldsymbol{r}|}\operatorname{erf}\left(\frac{|\boldsymbol{r}|}{2r_{\mathrm{s}}}\right), \quad (2.15)$$

where the complementary error function $\operatorname{erfc}(x) \equiv 1 - \operatorname{erf}(x)$ and r_{s} is the length scale of the force split*. To calculate $\widetilde{\mathcal{G}}_{\mathrm{l}}$, we take the *regularized*† Fourier transform of $\mathcal{G}_{\mathrm{l}}$:

$$\begin{aligned}
\widetilde{\mathcal{G}}_{\mathrm{l}}(\boldsymbol{k}) &= \lim_{\epsilon\to0}\int \mathrm{d}\boldsymbol{r}\, \mathcal{G}_{\mathrm{l}}(\boldsymbol{r})\mathrm{e}^{-\epsilon|\boldsymbol{r}|}\mathrm{e}^{-\mathrm{i}\boldsymbol{k}\boldsymbol{r}} \\
&= -\frac{1}{2}\lim_{\epsilon\to0}\int_0^\infty \mathrm{d}|\boldsymbol{r}|\, \operatorname{erf}\left(\frac{|\boldsymbol{r}|}{2r_{\mathrm{s}}}\right)\mathrm{e}^{-\epsilon|\boldsymbol{r}|}|\boldsymbol{r}| \int_{-1}^{1} \mathrm{d}\cos\theta\, \mathrm{e}^{-\mathrm{i}|\boldsymbol{k}||\boldsymbol{r}|\cos\theta} \\
&= -\frac{1}{|\boldsymbol{k}|}\lim_{\epsilon\to0}\int_0^\infty \mathrm{d}|\boldsymbol{r}|\, \operatorname{erf}\left(\frac{|\boldsymbol{r}|}{2r_{\mathrm{s}}}\right)\mathrm{e}^{-\epsilon|\boldsymbol{r}|}\sin\big(|\boldsymbol{k}||\boldsymbol{r}|\big) \\
&= -\frac{1}{|\boldsymbol{k}|}\lim_{\epsilon\to0}\operatorname{Im}\int_0^\infty \mathrm{d}|\boldsymbol{r}|\, \operatorname{erf}\left(\frac{|\boldsymbol{r}|}{2r_{\mathrm{s}}}\right)\mathrm{e}^{(\mathrm{i}|\boldsymbol{k}|-\epsilon)|\boldsymbol{r}|} \\
&= -\frac{1}{|\boldsymbol{k}|}\lim_{\epsilon\to0}\operatorname{Im}\Bigg[\underbrace{\operatorname{erf}\left(\frac{|\boldsymbol{r}|}{2r_{\mathrm{s}}}\right)\frac{\mathrm{e}^{(\mathrm{i}|\boldsymbol{k}|-\epsilon)|\boldsymbol{r}|}}{\mathrm{i}|\boldsymbol{k}|-\epsilon}\Bigg|_{|\boldsymbol{r}|=0}^{|\boldsymbol{r}|=\infty}}_{0} - \int_0^\infty \overbrace{\frac{\mathrm{e}^{-\boldsymbol{r}^2/(4r_{\mathrm{s}}^2)}}{r_{\mathrm{s}}\sqrt{\pi}}}^{\frac{\mathrm{d}}{\mathrm{d}|\boldsymbol{r}|}\operatorname{erf}\left(\frac{|\boldsymbol{r}|}{2r_{\mathrm{s}}}\right)} \frac{\mathrm{e}^{(\mathrm{i}|\boldsymbol{k}|-\epsilon)|\boldsymbol{r}|}}{\mathrm{i}|\boldsymbol{k}|-\epsilon}\,\mathrm{d}|\boldsymbol{r}|\Bigg] \\
&= -\frac{1}{\sqrt{\pi}|\boldsymbol{k}|r_{\mathrm{s}}}\lim_{\epsilon\to0}\operatorname{Im}\left[\frac{\mathrm{i}|\boldsymbol{k}|+\epsilon}{\boldsymbol{k}^2+\epsilon^2}\int_0^\infty \exp\left(-\frac{\boldsymbol{r}^2}{4r_{\mathrm{s}}^2}\right)\mathrm{e}^{(\mathrm{i}|\boldsymbol{k}|-\epsilon)|\boldsymbol{r}|}\,\mathrm{d}|\boldsymbol{r}|\right] \\
&= -\frac{1}{\sqrt{\pi}\boldsymbol{k}^2 r_{\mathrm{s}}}\operatorname{Re}\int_0^\infty \exp\left(-\frac{\boldsymbol{r}^2}{4r_{\mathrm{s}}^2}\right)\mathrm{e}^{\mathrm{i}|\boldsymbol{k}||\boldsymbol{r}|}\,\mathrm{d}|\boldsymbol{r}|
\end{aligned}$$

*The factor $1/2$ in the error function and the complementary error function in (2.14) and (2.15) ensures that $\mathcal{G}_{\mathrm{s}}(r_{\mathrm{s}}) \approx \mathcal{G}_{\mathrm{l}}(r_{\mathrm{s}}) \approx \mathcal{G}(r_{\mathrm{s}})/2$.

†Were we to use a regular Fourier transformation, the radial integral would introduce an additional, divergent term into $\widetilde{\mathcal{G}}_{\mathrm{l}}$.

$$= -\frac{1}{2\sqrt{\pi}\boldsymbol{k}^2 r_\mathrm{s}} \int_{-\infty}^{\infty} \exp\left(-\frac{\boldsymbol{r}^2}{4r_\mathrm{s}^2}\right) \mathrm{e}^{\mathrm{i}|\boldsymbol{k}||\boldsymbol{r}|}\, \mathrm{d}|\boldsymbol{r}|$$
$$= -\frac{1}{\boldsymbol{k}^2} \mathrm{e}^{-\boldsymbol{k}^2 r_\mathrm{s}^2}\,, \tag{2.16}$$

where integration by parts has been used. We recognize the last integral as an (inverse) Fourier transformation of a Gaussian, which is again a Gaussian. This Gaussian came from the derivative of the error function, which follows directly from its definition;

$$\mathrm{erf}(x) \equiv \frac{2}{\sqrt{\pi}} \int_0^x \mathrm{e}^{-t^2}\, \mathrm{d}t\,,$$
$$\Rightarrow \frac{\mathrm{d}}{\mathrm{d}x} \mathrm{erf}(ax) = \frac{2a}{\sqrt{\pi}} \mathrm{e}^{-a^2x^2}\,. \tag{2.17}$$

Inserting (2.14) and (2.16) into (2.13), we arrive at the final expression for the potential:

$$\phi(\boldsymbol{r}) = -G \sum_{j=1}^{N} m_j \Bigg[\sum_{\boldsymbol{n}\in\mathbb{Z}^3} \frac{\mathrm{erfc}\left(\frac{|\boldsymbol{r}_j+\boldsymbol{n}L-\boldsymbol{r}|}{2r_\mathrm{s}}\right)}{|\boldsymbol{r}_j+\boldsymbol{n}L-\boldsymbol{r}|} + \frac{4\pi}{L^3} \sum_{\boldsymbol{k}\in\left\{\frac{2\pi}{L}\boldsymbol{h}\,|\,\boldsymbol{h}\in\mathbb{Z}^3\right\}} \frac{\exp\left(-\boldsymbol{k}^2 r_\mathrm{s}^2\right)}{\boldsymbol{k}^2} \cos(\boldsymbol{k}[\boldsymbol{r}_j-\boldsymbol{r}]) \Bigg]\,, \tag{2.18}$$

where the odd, imaginary part of the complex exponential vanishes because $\boldsymbol{k}$ takes on symmetrical values.

It should be noted that as it stands, the $\boldsymbol{k} = \boldsymbol{0}$ term in (2.18) is badly behaved due to the factor $1/\boldsymbol{k}^2$. This term is however independent of $\boldsymbol{r}$, and so though being infinite, it amounts to a physically unimportant, spatially uniform shift in the potential. It does raise a slight concern as regards to the validity of (2.18) though, as we might have expected a nicely behaved potential as the solution to the Poisson equation. The real reason for this behaviour is in fact that *no solution exists* to the Poisson equation with triply-periodic boundary conditions, unless the average density is zero. This can be understood by considering the Poisson equation (2.9), where triply-periodic boundaries amounts to integrating over all space. This integral is then guaranteed to diverge if the mean value of ρ is different from 0. A commonly deployed trick to avoid this difficulty is to focus on the *peculiar* potential, which is exactly the potential you get from the Poisson equation, when you substitute the density for the density contrast. It is however equivalent to postpone the problem until the end — as we have done — and simply throwing away the $\boldsymbol{k} = 0$ term. To be precise, one should then refer to ϕ as the peculiar potential. Since we now know that this is really the only sensible potential in our system, we will continue to refer to it simply as the potential.

The Ewald summation has produced the expression for the gravitational potential (2.18). The gravitational force on particle i due to all particle images j, is then the negative gradient of this potential, times the mass of particle i:

$$\begin{aligned}\boldsymbol{F}_i = Gm_i \sum_{\substack{j=1\\ j\neq i}} m_j \Bigg\{ \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \frac{\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}_i}{|\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}_i|^3} \Bigg[\operatorname{erfc}\Bigg(\frac{|\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}_i|}{2r_\mathrm{s}} \Bigg) \\ + \frac{|\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}_i|}{\sqrt{\pi}\, r_\mathrm{s}} \mathrm{e}^{-\frac{|\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}_i|^2}{4r_\mathrm{s}^2}} \Bigg] \\ + \frac{4\pi}{L^3} \sum_{\boldsymbol{k}\in\left\{\frac{2\pi}{L}\boldsymbol{h}\,|\,\boldsymbol{h}\in\mathbb{Z}^3\setminus\boldsymbol{0}\right\}} \frac{\boldsymbol{k}}{\boldsymbol{k}^2} \exp\!\left(-\boldsymbol{k}^2 r_\mathrm{s}^2\right) \sin(\boldsymbol{k}[\boldsymbol{r}_j - \boldsymbol{r}_i]) \Bigg\},\end{aligned} \tag{2.19}$$

where the (negative) derivative of the complimentary error function is given in (2.17). Here we have explicitly removed the $\boldsymbol{k} = \boldsymbol{0}$ term, in accordance with the above discussion. It is interesting to note that the differentiation does not automatically rid us from this constant term.

Both the complementary error function and the Gaussians in (2.19) fall off rapidly as their arguments are increased. We can thus safely approximate the two infinite sums by partial sums, where $\boldsymbol{n}$ and $\boldsymbol{k}$ only takes on values within some appropriate distances to their origin. These distances depend on the convergence rate, which is controlled by r_s. That is, the total expression (2.19) is independent on the value of r_s, but the individual infinite sums are not. Lowering r_s decreases the range of the short-range force, which increases the rate of convergence of the short-range sum. Fewer values of $\boldsymbol{n}$ is then needed to approximate the sum. The cost of this is that the now missing force components must be supplied by the long-range sum, which must now resolve smaller scales than before. The partial long-range sum thus need to include larger $\boldsymbol{k}$ values than before. By choosing suitable values for $\max|\boldsymbol{n}|$, $\max|\boldsymbol{k}|$ and r_s, (2.18) then allow us to control the accuracy and the convergence rate of the computation.

It can be illuminating to visualize the periodic force. In Figure 2.1, the periodic force is plotted together with the usual, non-periodic force. For small separations, the two forces are approximately equal, though the periodic force is slightly smaller. In fact, within the lower left quadrant of Figure 2.1, corresponding to the first octant of the box, the periodic force is consistently smaller than the non-periodic force. This makes sense, since the closest replicas pull in the opposite direction of the proper particle. It is also very clear from Figure 2.1 that the periodic force has reflective symmetries. This is simply a direct consequence of the periodicity of the particle images. It is a useful property, as we shall see when implementing the periodic force numerically.

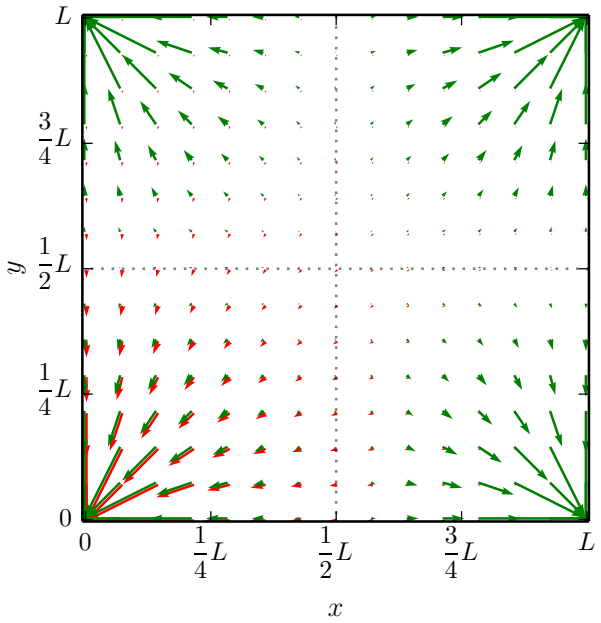


Figure 2.1 – Gravitational force fields in the $z = 0$ plane. A massive particle is placed at the origin, while test particles are placed regularly throughout the $z = 0$ plane of the box. The red arrows correspond to the non-periodic force on these test particles due to the particle at the origin. The green arrows correspond to the periodic force (2.19) on the test particles, due to the particle at the origin and all of its replicas. The length of arrows is proportional to the magnitude of the force. There is a slight shift in the relative locations of the red and the green arrows. This is so that one arrow do not cover the other.

2.1.3 Numerical Implementation

In the last subsection we derived an expression for the periodic gravitational force (2.19). Before we can upgrade the code in listing 2.1 with this expression, there are a few remaining hurdles. First of all, (2.19) contains the free parameter r_{s}, the value of which needs to be specified. We remember that its value determine the scale of the short-range/long-range force split, and therefore the convergence rate of the infinite sums in (2.19). In 1976, [7] suggested the values

$$r_{\mathrm{s}} = \frac{L}{4}\,, \qquad |\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}_i| < 2.6L\,, \qquad \boldsymbol{h}^2 < 8\,, \tag{2.20}$$

which gives very fast computations and good accuracy. Since then, computer power has increased hugely, and so it becomes reasonable to opt for better accuracy. In 1990, [8] therefore suggested

$$r_{\mathrm{s}} = \frac{L}{4}\,, \qquad |\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}_i| < 3.6L\,, \qquad \boldsymbol{h}^2 < 10\,, \tag{2.21}$$

which is what is adopted in the CONCEPT code*. Today's computer power do allow us to increase the accuracy still further, but (2.21) is already so precise that it is not at all the weak point in the combined computation scheme.

Note that the truncations of the $\boldsymbol{n}$ and the $\boldsymbol{h}$ sums as stated in (2.20) or (2.21), are determined by the length of $\boldsymbol{n}$ and $\boldsymbol{h}$. This corresponds to "summing spherically", as opposed to e.g. "summing cubically", which would be the case if the size of each component of $\boldsymbol{n}$ and $\boldsymbol{h}$ were constrained. This choice of spherical summation amounts to settle for a specific order in which to do the infinite sum over images. With this choice, the sum becomes absolutely convergent.

Even though (2.21) truncates the sums in (2.19) to contain a manageable number of terms, it is still vastly more expensive to compute than the non-periodic force. Since the force on particle i due to particle j and all of its replicas (the right-hand-side of (2.19) without the first summation) varies smoothly with $\boldsymbol{r}_j - \boldsymbol{r}_i$, it is possible to tabulate the periodic force prior to running the actual simulation. When running the simulation, the periodic force between a given pair of particles can then be found by a simple lookup in this table. A finite box and the smoothness of the periodic force allows for a finite table. Since Newton's constant and the masses are placed outside of the infinite sums in (2.19), we can leave them out when doing the tabulation, greatly improving its generality. The same is however not true for the box size L. However, since the entire curly bracket in (2.19) has units of inverse length squared, it is possible to write it in units of L^{-2}. If we tabulate the periodic force in this way, the same table can be used for any box size. This amounts to multiplying the curly brace of (2.19) with L^2. We end up with

$$\begin{aligned}\frac{L^2}{Gm_im_j}\boldsymbol{F}_{\text{cor}}(\boldsymbol{\varsigma}_{ij}) = &\sum_{\substack{\boldsymbol{n}\in\mathbb{Z}^3\\ |\boldsymbol{\varsigma}_{ij}+\boldsymbol{n}|<3.6}} \frac{\boldsymbol{\varsigma}_{ij}+\boldsymbol{n}}{|\boldsymbol{\varsigma}_{ij}+\boldsymbol{n}|^3}\left[\operatorname{erfc}(2|\boldsymbol{\varsigma}_{ij}+\boldsymbol{n}|) + \frac{4}{\sqrt{\pi}}|\boldsymbol{\varsigma}_{ij}+\boldsymbol{n}|\mathrm{e}^{-4|\boldsymbol{\varsigma}_{ij}+\boldsymbol{n}|}\right] \\ &+ 4\pi\sum_{\boldsymbol{k}\in\left\{2\pi\boldsymbol{h}\,|\,\boldsymbol{h}\in\mathbb{Z}^3,\boldsymbol{h}\neq\boldsymbol{0},|\boldsymbol{h}|<10\right\}} \frac{\boldsymbol{k}}{\boldsymbol{k}^2}\exp\left(-\frac{\boldsymbol{k}^2}{16}\right)\sin(\boldsymbol{k}\boldsymbol{\varsigma}_{ij}) \\ &- \frac{\boldsymbol{\varsigma}_{ij}}{|\boldsymbol{\varsigma}_{ij}|^3}\,,\end{aligned} \tag{2.22}$$

where

$$\boldsymbol{\varsigma}_{ij} = \frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{L}$$

and both $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ should be positions within the actual, $\boldsymbol{n} = \boldsymbol{0}$ box. The dimensionless force from the actual particle, $\boldsymbol{\varsigma}_{ij}/|\boldsymbol{\varsigma}_{ij}|^3$, has been subtracted from the right-hand-side of (2.22). This leaves only the periodic correction to the force, rather than the total periodic force. The right-hand-side of (2.22) is

*It is also what is used in the GADGET-2 code.

what is tabulated, while the correction force $\boldsymbol{F}_{\text{cor}}(\boldsymbol{\varsigma}_{ij})$ is the force on particle i due only to the replicas of particle j. It would be more efficient to tabulate the total periodic force $\boldsymbol{F}(\boldsymbol{\varsigma}_{ij})$ due to the replicas *and* the the actual particle, but it turns out that having to manually add in the direct force is valuable, as we shall see in the next subsection.

Before tabulating (2.22), it is worth looking for symmetries to save memory. This is easier using physical intuition than just looking at (2.22). Each component of $\boldsymbol{\varsigma}_{ij}$ can range from -1 to 1. However, the magnitude of the force can only depend on the distance between the two particles. We therefore only have to tabulate the periodic force corrections for $\boldsymbol{\varsigma}_{ij}$ with components between 0 and 1, reducing the needed memory by a factor of eight. The correct signs must then be added in programmatically. Another reduction in needed memory comes from noticing that the distance between the nearest image of particle j and particle i cannot have components greater than $L/2$. For an image with the distance $3/5L$ in the x-direction, say, there exist a neighbouring image with the distance $3/5L - L = -2/5L < L/2$ in the x-direction. We can thus save another factor of eight in memory by choosing to always do the lookup of the force on the nearest image. We can write down these symmetries as

$$\boldsymbol{F}(\boldsymbol{\varsigma}_{ij}) = \boldsymbol{s} \circ \boldsymbol{F}(\boldsymbol{s} \circ \boldsymbol{\varsigma}_{ij}) \,, \quad \boldsymbol{s} = (\pm 1, \pm 1, \pm 1) \,, \tag{2.23}$$

$$\boldsymbol{F}(\boldsymbol{\varsigma}_{ij}) = \boldsymbol{F}(\boldsymbol{\varsigma}_{ij} + \boldsymbol{n}) \,, \quad \boldsymbol{n} \in \mathbb{Z}^3 \,, \tag{2.24}$$

where $\circ$ is the elementwise product. The total periodic force is of course related to the correction force by

$$\boldsymbol{F}(\boldsymbol{\varsigma}_{ij}) = \boldsymbol{F}_{\text{cor}}(\boldsymbol{\varsigma}_{ij}) + \frac{G m_i m_j}{L^2} \frac{\boldsymbol{\varsigma}_{ij}}{|\boldsymbol{\varsigma}_{ij}|^3} \,. \tag{2.25}$$

All in all, when tabulating the periodic force corrections, we need only to let the components of $\boldsymbol{\varsigma}_{ij}$ run from 0 to $1/2$. The question of how to distribute the points in this octant remains. It is far simplest and computationally fastest to do the tabulation for a cubic array of $\boldsymbol{\varsigma}_{ij}$. Also, as is demonstrated by [8], the correction force varies smoothly for all separations. This is also evident from the lower left quadrant of Figure 2.1, where the differences between the red and green arrows correspond to correction forces. This renders an inhomogeneous density of tabulation points unnecessary. In the GADGET-2 code, a grid of 64^3 points is tabulated. This is also chosen as the standard for the CONCEPT code, but it can easily be changed in the parameter file.

What is left to define is the precise way in which to do the lookups. The simplest might be to lookup $\boldsymbol{F}_{\text{cor}}$ at the tabulated value of $\boldsymbol{\varsigma}_{ij}$ which is closest to the one actually desired. More accurate methods can be devised, where several tabulation points are interpolated. Once again, CONCEPT follows in the footsteps of GADGET-2 and implements a so-called cloud-in-cell (CIC) interpolation scheme. We will study this and similar interpolation schemes in subsection 2.2.2.

2.1.4 Softening

In this section we have derived Newton's law of universal gravitation with triply-periodic boundary conditions (2.19) and described how to implement it numerically, as (2.22). All along we have modelled the particles as infinitesimal points, which have served to keep their interactions as simple as possible. This simplicity comes at a cost, however. Modelling the single-particle densities as delta functions means that the gravitational interaction between a pair of particles is unbounded; it becomes arbitrarily strong at small separations, which is undesirable. Since the time evolution of the system is discrete, it is possible for two particles to suddenly be very close together, which then causes unphysical large accelerations. This is a problem of all direct summation codes*.

Even if we imagine that we could use arbitrarily small time steps, the fact that all the mass of a particle is concentrated at a single point is not desirable; it increases two-body relaxation and leads to the formation of many binary systems. Two-body relaxation simply refers to violent collisions between two particles, as described above. If our simulation particles actually represented physical, solitary particles (e.g. stars), this behaviour would indeed be desired. But our particles are the result of a numerical discretization of the continuous density throughout the Universe; a Monte Carlo ensemble of phase space. We want our particles to exhibit the collective behaviour of a continues mass distribution, not the behaviour of N single particles. Increasing N helps to prevent this under-sampling of the mass distribution, by pushing it towards lower scales. It would be nice however to gain additional resolution for free — that is, without increasing N.

We thus wish to "soften" the interactions to prevent these unphysical close encounters. The problem can be traced back to the choice of modelling the single-particle densities as delta functions. These singularities then propagate to the potential and to the force. We need to invent a new single-particle density profile, for which our calculations still hold true. From the shell theorem, we know that any finite, spherically symmetric mass distribution has the same external gravitational field as that of a point mass. We thus have a large freedom in choosing the exact form of the single-particle density. A popular and simple choice is that of a Plummer sphere [9], originally invented for representing the observed density profiles of star clusters:

$$\rho_{\mathrm{P}}(|\boldsymbol{r}|) \equiv \frac{3m}{4\pi\epsilon^3}\left(1+\frac{|\boldsymbol{r}|^2}{\epsilon^2}\right)^{-\frac{5}{2}}. \tag{2.26}$$

Here, m is the mass of the particle, while ϵ is a parameter with units of length, determining the size of the particle. We see from (2.26) that the density of a

*That is, codes as those in listing 2.1. Utilizing the Ewald method to accomplish periodic boundaries amounts simply to change the force expression; the structure of the code is untouched.

Plummer sphere falls off monotonically in the radial direction, and that the central density equals the mean density of a sphere with mass m and radius ϵ. The gravitational field produced by the Plummer density profile (2.26) is the solution to the Poisson equation where (2.26) is used as the density:

$$\phi_{\mathrm{P}}(|\boldsymbol{r}|) = -\frac{Gm}{\sqrt{\boldsymbol{r}^2 + \epsilon^2}}\,. \tag{2.27}$$

It is clear from (2.27) that the presence of ϵ removes the singularity at $\boldsymbol{r} = \boldsymbol{0}$. The gravitational force on a Plummer sphere i due to a point particle j, separated by the distance $\boldsymbol{r} = \boldsymbol{r}_j - \boldsymbol{r}_i$ is then given by the gradient of (2.27):

$$\boldsymbol{F}_{\mathrm{P},ij}(\boldsymbol{r}) = Gm_i m_j \frac{\boldsymbol{r}}{\left(\boldsymbol{r}^2 + \epsilon^2\right)^{3/2}}\,, \tag{2.28}$$

where m_i and m_j are the particle masses. Of course, (2.28) also accurately describes the force between two Plummer spheres, as long as they do not overlap significantly. In CONCEPT and other N-body codes utilizing Plummer softening, the gravitational force between particles is modelled precisely by (2.28). Only one of the particles in each interaction can then really be ascribed the Plummer shape; the other remains a point particle*. As the original goal was to soften the two-body interaction and not to model the particles as Plummer spheres per se, this is not a problem.

The Plummer density (2.26), potential (2.27) and force (2.28) are plotted together in Figure 2.2. We see that the density and the potential have a flat plateau around the origin, which translates into a very small force. Instead of increasing towards the origin, the force is strongest at $|\boldsymbol{r}| = \epsilon/\sqrt{2}$. The unsoftened force between two point particles $\boldsymbol{F}_{ij}$ is also shown in Figure 2.2. We see that the softened force is always less than the unsoftened force, but they agree very well for $|\boldsymbol{r}|$ larger than a couple of ϵ.

In writing down the final expression for the periodic force (2.22), we explicitly subtracted the force from the nearest image, resulting in the correction force $\boldsymbol{F}_{\mathrm{cor}}$. We can now incorporate the Plummer softening into the total periodic force, replacing (2.25) with

$$\begin{aligned}\boldsymbol{F}(\boldsymbol{\varsigma}_{ij}) &= \boldsymbol{F}_{\mathrm{cor}}(\boldsymbol{\varsigma}_{ij}) + \frac{Gm_i m_j}{L^2}\frac{\boldsymbol{\varsigma}_{ij}}{\left(\boldsymbol{\varsigma}_{ij}^2 + \frac{\epsilon^2}{L^2}\right)^{3/2}} \\ &= \boldsymbol{F}_{\mathrm{cor}}(\boldsymbol{\varsigma}_{ij}) - Gm_i m_j \frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{\left(|\boldsymbol{r}_j + \boldsymbol{r}_i|^2 + \epsilon^2\right)^{3/2}}\,.\end{aligned} \tag{2.29}$$

The softening is then only applied for the nearest image, which for sensible (small) ϵ/L is guaranteed to be good enough. As is clear from Figure 2.2,

*Which particle we refer to as the Plummer sphere and which we refer to as the point particle is of course ambiguous, in accordance with Newton's third law.

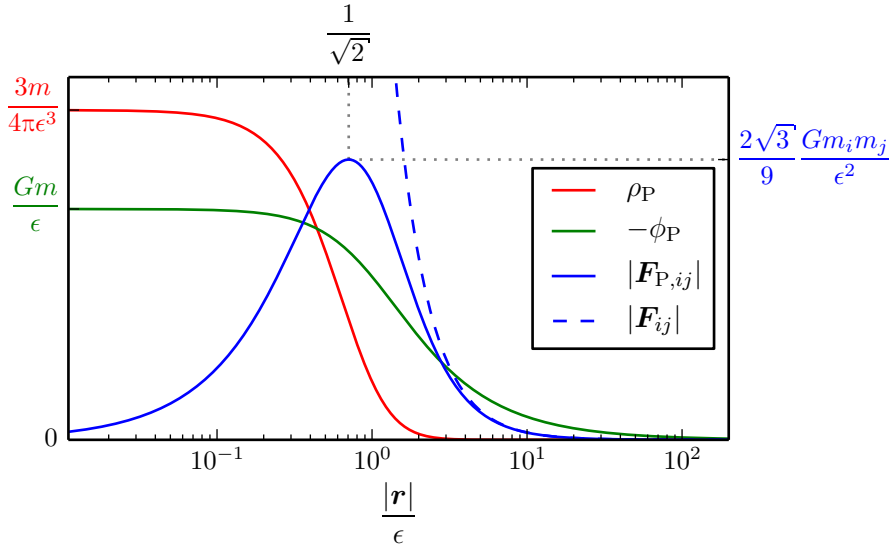


Figure 2.2 – The density ρ_P (2.26) and potential ϕ_P (2.27) of a Plummer sphere together with the magnitudes of the Plummer force $\boldsymbol{F}_{\mathrm{P},ij}$ (2.28) and the unsoftened force between two point particles $\boldsymbol{F}_{ij}$. The potential has been negated to better fit the figure. Since density, potential and force have different units, they cannot share the same axis. They have therefore been given separate (linear) ordinate axes, defined by the common 0 and the expression written next to their respective maximum values. The relative scale between these three should therefore not be given any significance. The two force magnitudes are however shown to the same scale.

the gravitational force at scales below a couple of ϵ is heavily altered by the softening. All structure at these scales are therefore to be disregarded, as they are not the result of proper gravitation. Introducing softening into the simulation thus necessarily introduces a smallest distance scale at which the simulation should be trusted*. We are therefore interested in a small ϵ, but it should not be so small that all softening effects vanish. A sensible criterion is that the fractional volume of the Universe occupied with particles should remain the same, regardless of N and L. That is, $N\epsilon^3/L^3$ should be constant. As stated in [10] and [11], optimal values for this constant is

$$\epsilon \approx (2\text{–}4\,\%)\frac{L}{N^{1/3}}\,.$$

That is, ϵ is a few percent of the mean interparticle distance. A value of $\epsilon = 0.03L/N^{1/3}$ is chosen as the standard in the CONCEPT code, but it can easily be changed in the parameter file.

*Having finitely many particles and a finite time step size alone is enough for such a scale to exist, and so it predates our introduction of softening. However, the softening parameter ϵ allow us to quantify this smallest scale explicitly.

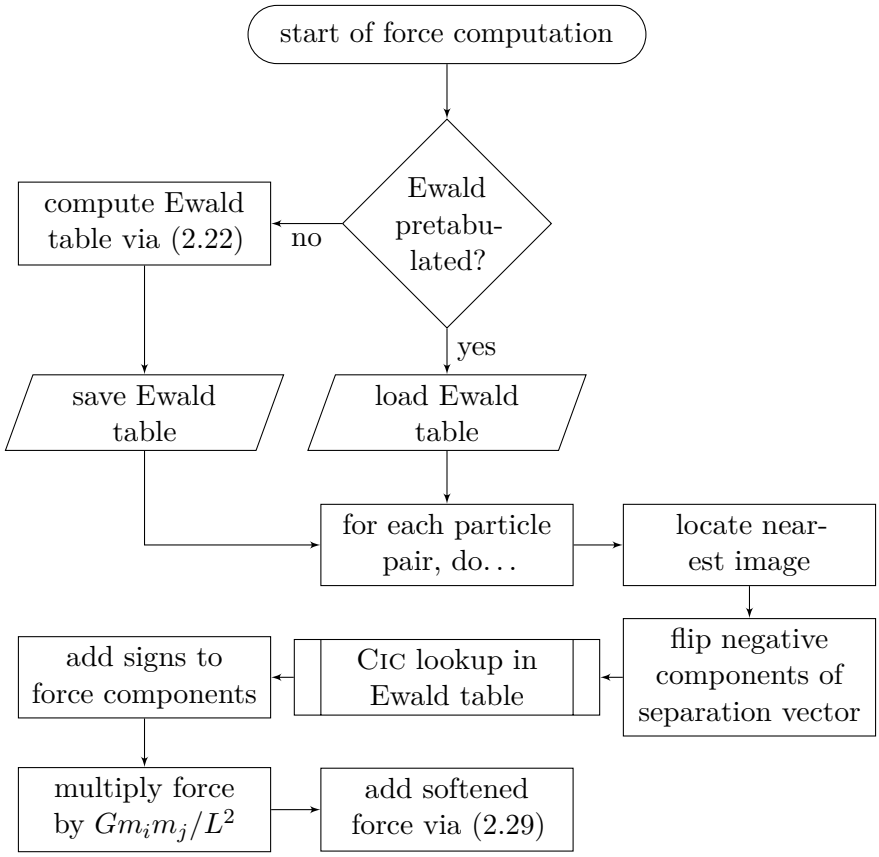


Figure 2.3 – Flowchart of the PP algorithm.

2.1.5 Recap of the Method

In this section we have build the components needed for the simplest gravitational algorithm used in cosmological N-body simulations. This is the basic direct summation algorithm of listing 2.1, extended with periodic force corrections via the Ewald method, together with Plummer softening. This is commonly known as the particle-particle algorithm, or simply the PP algorithm. This name is not completely unambiguous however, as similar algorithms utilizing e.g. a different softening technique still fall under this name. The different components of the PP algorithm were introduced one after another, as we discovered the need for them. This last subsection serves to put the results together to form a coherent picture of the PP method.

Figure 2.3 is a flowchart of the PP algorithm, briefly summarising the major steps, their order and interrelationship. These steps are described below in greater detail.

- Prior to the loop over all particle pairs, the Ewald table is needed. If

the Ewald corrections are already tabulated*, they are loaded from disk, saving time not to recompute them at the beginning of each run. If no previously constructed Ewald table is found, it is recomputed and saved to disk (though kept in memory) via (2.22).

The loop over all particle pairs begin. In the following we consider the iteration involving particle i and j. Both $\boldsymbol{F}_{ij}$ and $\boldsymbol{F}_{ji} = -\boldsymbol{F}_{ij}$ are computed in this single iteration.

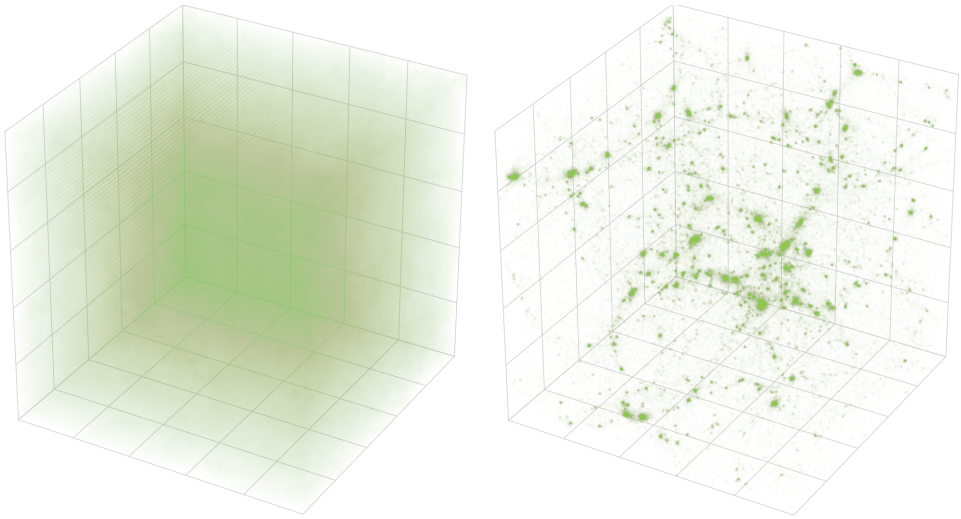
- The image of particle j nearest to particle i is located. This image is used rather than particle j itself (it is of course possible that the actual particle is itself the nearest image) in the following. This amounts to exploiting the symmetry (2.24).
- If the separation vector from particle i to the image of j has negative components, the symmetry (2.23) is now used to flip these components. The original signs of the components are saved for later. The separation vector now lies within the tabulated region.
- The Ewald correction force between the particle pair is now looked up in the table and CIC interpolated to exactly match the separation. As mentioned before, CIC and interpolation in general will be discussed in detail in subsection 2.2.2.
- The signs previously removed from the components of the separation vector now has to be placed on the components of the force, as the negation carries over linearly for each dimension.
- The factor Gm_im_j/L^2 is multiplied on the as of yet dimensionless force. The Gm_im_j of course comes from Newton's law of universal gravitation, while L^{-2} is needed due to the way the Ewald corrections are stored.
- Finally, the softened force from the nearest image of particle j is computed as in (2.29) and added to the periodic correction force. The total periodic force $\boldsymbol{F}_{ij}$ between particles i and j has now been computed.

This concludes our discussion of the PP method. We have left out any discussion about parallelism, which we will describe collectively for all gravitational methods near the end of this chapter.

If we add time evolution† to the PP method, we end up with a complete N-body code. Given some initial conditions, we can then evolve the particle system in time. Figure 2.4 contain two plots of the particles; one at an early time, where the Universe is very homogeneous, and one at the present time. The CONCEPT code, utilizing the PP method, has been used to evolve the

*In the resolution desired for the particular run.

†Time evolution will be established in chapter 3.



$a = 0.02$ $a = 1$

Figure 2.4 – Raw plots of particle configuration at different times, showing structure formation in action. The configuration on the left with $a = 0.02$ corresponds to initial conditions for the simulation. The particles are very homogeneously spread out and no structure are seen. The same particles except ~ 14 billion years later are shown on the right. Here we see many different sized structures. The simulation contains $N = 64^3$ particles and was run with the CONCEPT code utilizing the PP method. The box size is $L = 64\,\mathrm{Mpc}/h$, where $h = H_0/(100\,\mathrm{km\ s^{-1}\ Mpc^{-1}})$. The cosmology of the simulation is $\Omega_\Lambda = 0.7$, $\Omega_\mathrm{m} = 0.3$, $H_0 = 70\,\mathrm{km\ s^{-1}\ Mpc^{-1}}$.

particles. The initial conditions themselves were produced using a modified GADGET-2 code (see [12]), able to take in transfer functions from the CAMB [4] code and dump out a GADGET-2 snapshot containing a corresponding particle distribution.

The transformation depicted* in Figure 2.4 is the direct result of an N-body code run. Additional analysis can decompose the raw particle configuration into other, more manageable data. One such decomposition is the matter power spectrum $P(|\boldsymbol{k}|)$, as given in (1.42). In the next section, we shall see how to numerically compute the power spectrum from the particle distribution. The power spectrum corresponding to the right plot of Figure 2.4 is shown in Figure 2.5. The finite number of particles make for a jagged power spectrum. The density of data points increases drastically with $|\boldsymbol{k}|$, simply because a given small separation (large $\boldsymbol{k}$) fit into the box in many more ways than a larger separation. As we would expect, we see structure become more and more abundant as we lower the scale. At the very high $|\boldsymbol{k}|$ end though, P seems to increase. This defect is not caused by the PP method, as the Plummer radius

*Since Figure 2.4 do not show velocities, half of the information generated by the simulation is missing.

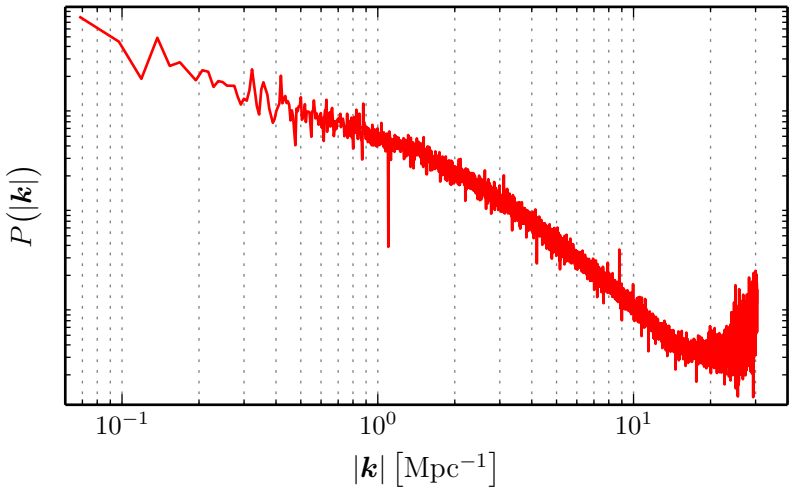


Figure 2.5 – Complete and unpolished power spectrum from the PP simulation of Figure 2.4 at $a = 1$.

$\epsilon = 0.03L/N^{1/3}$ has a corresponding $|\boldsymbol{k}|$ value of $|\boldsymbol{k}|_{\mathrm{P}} = 2\pi/\epsilon = 146\,\mathrm{Mpc}^{-1}$, where the values of N and L are those given in Figure 2.4. Instead it is a result of the way the power spectrum itself has been computed.

2.2 The Particle-Mesh Method

In the last section, our concern was with the creation of our first gravitational solver; the PP method. The two key extensions to the naïve listing 2.1 was periodicity and softening of the force. Computing gravitational forces as if all of space were filled with periodic replicas of the particle configuration within the box, we were able to emulate an infinite universe containing infinitely many particles. By softening the interaction, we got rid of unphysical behaviour having to do with two-body relaxation. The numerical results obtained via the PP method are superb, more so than the results of the PM method — the subject of the current section.

The shortcomings of the PP method becomes apparent when we consider a box with $N \gg 2$ particles. Computing the gravitational force on all particles for a single time step via the PP method requires $\mathcal{O}(N^2)$ operations, since each of the N particles have $N - 1 \approx N$ partners. This is an inevitable consequence of the long-range nature of gravity together with the formalism of Newton's law of universal gravitation, where gravity is described as a particle-particle interaction. What we need is a numerical method for computing gravitation, which has a better scaling than the $\mathcal{O}(N^2)$ scaling of the PP method. Such a method may involve lots of additional overhead per particle, and so might be slower than the PP method for low N. This is not a concern of ours, as we are interested in the very large N regime. The scaling [13] of the PM method

is only $\mathcal{O}(N \log N)$, a tremendous improvement: For the largest simulations currently feasible ($N \sim 10^{12}$, [14]), this is tens of billions of times faster than the PP method*!

The key to the speed of the PM algorithm is this. Instead of direct particle-particle interactions, the particles interact with a global gravitational field. We thus substitute Newton's law of universal gravitation as our main equation for gravity, for the Poisson equation. Numerically, N particles takes part in the creation of the field, after which each particle receives a force by a single particle-field interaction. We have then moved from a global to a local description of gravitation. It seems reasonable that this description has a better scaling than $\mathcal{O}(N^2)$, but the exact form $\mathcal{O}(N \ln N)$ is not obvious. Numerically, the continuous gravitational field is represented by a discrete but dense mesh of values. It is then this particle-mesh interaction that gives the PM method its name. Since the information stored on the mesh is computed from the particle distribution, it really carries no additional information than the particles themselves. This means that the PM method requires more computer memory than what is strictly needed to compute the gravitational forces, while a method like PP requires almost no memory in addition to the particles themselves. This feature of sacrificing memory (space) in order to gain efficiency (lower the computation time) is a common pattern in computing.

2.2.1 Overview

In the last section, we solved the general Poisson equation (2.1). The solution (2.9) can be written in terms of the Green's function of the Laplacian (2.8);

$$\begin{aligned}\phi(\boldsymbol{r}) &= 4\pi G \int \mathcal{G}_{\nabla^2}(\boldsymbol{r}' - \boldsymbol{r})\rho(\boldsymbol{r}')\,\mathrm{d}\boldsymbol{r}' \\ &= 4\pi G \mathcal{G}_{\nabla^2}(\boldsymbol{r}) * \rho(\boldsymbol{r})\,, \end{aligned} \tag{2.30}$$

where the integral has been written as a convolution. As with the PP method, several version of the PM method exists. The computationally fastest class of versions computes the gravitational potential in Fourier space. Remembering that the Fourier transform of a convolution is simply the product of individual Fourier transforms, (2.30) can immediately be cast into Fourier space:

$$\widetilde{\phi}(\boldsymbol{k}) = 4\pi G \widetilde{\mathcal{G}}_{\nabla^2}(\boldsymbol{k})\widetilde{\rho}(\boldsymbol{k})\,. \tag{2.31}$$

From (2.8), the Green's function is simply $\mathcal{G}_{\nabla^2}(\boldsymbol{r}) = -(4\pi|\boldsymbol{r}|)^{-1}$. To calculate $\widetilde{\mathcal{G}}_{\nabla^2}(\boldsymbol{k})$, we use a regularized Fourier transform, as otherwise the radial integral

*Indeed with this many particles, simulations using the PP method running on today's modern hardware have a runtime exceeding the timespan simulated! Generally in science, this is not necessarily a problematic feature. In cosmology however, it is a spectacular catastrophe, as neither the future nor the present can ever be reached by such simulations.

diverges:

$$\begin{aligned}
\widetilde{\mathcal{G}}_{\nabla^2}(\boldsymbol{k}) &= \lim_{\epsilon\to 0}\int \mathrm{d}\boldsymbol{r}\, \mathcal{G}_{\nabla^2}(\boldsymbol{r})\mathrm{e}^{-\epsilon|\boldsymbol{r}|}\mathrm{e}^{-\mathrm{i}\boldsymbol{k}\boldsymbol{r}} \\
&= -\frac{1}{2}\lim_{\epsilon\to 0}\int_0^\infty \mathrm{d}|\boldsymbol{r}|\,\mathrm{e}^{-\epsilon|\boldsymbol{r}|}|\boldsymbol{r}|\int_{-1}^{1}\mathrm{d}\cos\theta\,\mathrm{e}^{-\mathrm{i}|\boldsymbol{k}||\boldsymbol{r}|\cos\theta} \\
&= -\frac{1}{|\boldsymbol{k}|}\lim_{\epsilon\to 0}\int_0^\infty \mathrm{d}|\boldsymbol{r}|\,\mathrm{e}^{-\epsilon|\boldsymbol{r}|}\sin\big(|\boldsymbol{k}||\boldsymbol{r}|\big) \\
&= -\frac{1}{|\boldsymbol{k}|}\lim_{\epsilon\to 0}\mathrm{Im}\int_0^\infty \mathrm{d}|\boldsymbol{r}|\,\mathrm{e}^{(\mathrm{i}|\boldsymbol{k}|-\epsilon)|\boldsymbol{r}|} \\
&= -\frac{1}{|\boldsymbol{k}|}\lim_{\epsilon\to 0}\mathrm{Im}\left(\frac{1}{\epsilon-\mathrm{i}|\boldsymbol{k}|}\right) \\
&= -\frac{1}{|\boldsymbol{k}|}\lim_{\epsilon\to 0}\frac{|\boldsymbol{k}|}{\epsilon^2+\boldsymbol{k}^2} \\
&= -\frac{1}{\boldsymbol{k}^2}\,. \qquad (2.32)
\end{aligned}$$

This result does not apply for $\boldsymbol{k}=0$. As discussed in the previous section, we simply define $\widetilde{\mathcal{G}}_{\nabla^2}(\boldsymbol{0})\equiv\boldsymbol{0}$, removing structure of infinite wavelength (the mean density). To be precise, the potential from (2.31) is then really the *peculiar* potential. Combining (2.32) and (2.31) we obtain the solution for the potential in Fourier space:

$$\widetilde{\phi}(\boldsymbol{k}) = -\frac{4\pi G}{\boldsymbol{k}^2}\widetilde{\rho}(\boldsymbol{k})\,. \qquad (2.33)$$

We see that the differential Poisson equation reduces to a simple algebraic equation in Fourier space. This is such a nice feature that one might consider performing the entire simulation in Fourier space. This is possible, however impractical, as many computations (e.g. time evolution) is much more naturally expressed in real space. As the numerical algorithm for computing Fourier transforms — the fast Fourier transform, FFT — is indeed extremely fast, it is not a problem having to repeatedly convert back and forth between real and Fourier space. As the total PM method, the (fastest) FFT's have a complexity of $\mathcal{O}(N\ln N)\,\forall\, N$.

In the PM method, equation (2.33) shall serve as the main equation for gravity. It involves the fields $\widetilde{\phi}(\boldsymbol{k})$ and $\widetilde{\rho}(\boldsymbol{k})$. As the simulation is generally performed in real space, the fields $\phi(\boldsymbol{r})$ and $\rho(\boldsymbol{r})$ plays an equally important role. These four fields are represented numerically by regular, cubic meshes of values, spanning the box. A high-level view of the PM method can be described by the following steps.

1. Assign particle masses to the ρ mesh.
2. Transform $\rho \xrightarrow{\text{FFT}} \widetilde{\rho}$.

3. Compute $\widetilde{\phi}$ from $\widetilde{\rho}$ via (2.33).
4. Transform $\widetilde{\phi} \xrightarrow{\text{FFT}^{-1}} \phi$.
5. Compute forces from ϕ via finite differences.
6. Interpolate the forces on the mesh back to the particle positions.

In the first step, the density field is build from the particle distribution. Physically we do not make any distinction between these two quantities; the particles are precisely what make up the density field. The distinction arises numerically from the fact that such a field can be implemented in two distinct ways. One may discretize mass, resulting in a finite number of freely floating particles with constant mas, which samples the field values. This is precisely what is generally done in N-body simulations. Alternatively, one may choose to discretize space into finitely many points, but allow the mass at each point to vary continuously. These two descriptions are known as the Lagrangian and the Eulerian description, respectively. The first step in the list above amounts to such a change of representation, from Lagrangian to Eulerian.

Step two to four is the transformations $\rho \to \widetilde{\rho} \to \widetilde{\phi} \to \phi$, solving the Poisson equation. These four fields are all scalars, and are only ever used to produce the next field in line. This means that we can implement a single mesh, the values of which correspond to field values of each of the four fields, in turn. We thus refer to *the* mesh of the PM method, even though several fields are in play. A lot of our effort in the last section were spent implementing periodicity and softening, both of which we get for free when using the PM method. The FFT operation implicitly assumes periodicity (as opposed to e.g. vacuum boundary conditions), which means that $\widetilde{\rho}$ is the Fourier transform of the density field constructed by infinitely replicating the box in all directions. The problem of two-body relaxation presented by the PP method never comes up in the PM method, as here we do not have singular bodies. No softening is therefore needed. A smallest distance scale occur naturally by the fact that the meshes contain finitely many points; the smallest meaningful distance is the separation between neighbouring mesh points. Structure in the particle distribution below this scale are naturally to be disregarded. This scale is typically much larger than the corresponding softening length ϵ from the PP method, making it superior to the PM method with regards to resolution. The PM method is not exact for scales larger than this smallest scale either, as the geometry of the mesh introduces anisotropic errors at somewhat larger scales.

In the fifth and final step, forces are extracted from the potential, which can then be applied to the N particles. The forces are given by the negative gradient of the potential, which for the discrete mesh translates into finite differencing. Just as the Laplacian turned into multiplication by $-\boldsymbol{k}^2$ in Fourier space, the gradient turns into multiplication by $\mathrm{i}\boldsymbol{k}$. By multiplying $\widetilde{\phi}$ with $\mathrm{i}\boldsymbol{k}$ before transforming back to real space, the forces are obtained directly, without ever computing ϕ. As no finite differencing is needed, this is a slightly more

accurate method. We choose to use the finite difference approach however, as the multiplication by i$\boldsymbol{k}$ turns the field into a vector field, and so three separate inverse FFTs are needed instead of one.

In actuality, the numerical implementation of the PM algorithm is more complex than the above outline perpetrates. Before delving fully into these complexities, the next subsection develops formalism and techniques related to numerical meshes, so heavily used in the PM method. The use of meshes is also found elsewhere in N-body codes, e.g. the Ewald table used in the PP method.

2.2.2 Mesh Operations

A mesh is often used to numerically represent a continues field. In this work, whenever we use the word *mesh* (or sometimes *grid*), we refer to a regular, cubic array of numbers. A smooth field can be accurately represented by a coarse mesh, as long as appropriate interpolation is utilized when doing lookup. It is the goal of this subsection to develop the operations performed on the mesh, used in the PM method. This include the aforementioned interpolation together with finite differencing.

We denote the linear size of a mesh by N_m. That is, a mesh contains N_m^3 grid points. A specific point $\boldsymbol{p}$ is labelled by $(x_\mathrm{m}, y_\mathrm{m}, z_\mathrm{m}) \in \mathbb{N}^3$, where $0 \leq x_\mathrm{m}, y_\mathrm{m}, z_\mathrm{m} \leq N_\mathrm{m} - 1$. Often the field in question is a physical field in real space. We should then imagine the mesh as embedded within the box, each grid point having a specific, constant position in space. The mesh point $\boldsymbol{p}$ then have physical coordinates $(x_p, y_p, z_p) \equiv \boldsymbol{r_p} = H\boldsymbol{p}$, where H denote the separation between neighbouring points. The smallest cubic volumes spanned by the points are referred to as cells, each with volume H^3.

Meshes that span the entire box — like those used in the PM method — need to be toroidally constructed, to match the periodicity of the box. Smaller meshes, like the Ewald table used in the PP method which only spans one octant of the box, need not be periodic. To achieve periodicity, we add pseudopoints with $x_\mathrm{m} = N_\mathrm{m}$, $y_\mathrm{m} = N_\mathrm{m}$ or $z_\mathrm{m} = N_\mathrm{m}$ to the mesh, growing the mesh by one layer of pseudopoints ($3N_\mathrm{m}^2 + 3N_\mathrm{m} + 1$ pseudopoints added) at the three high-end faces. We then identify opposing mesh faces; $\boldsymbol{r_p} = H(\boldsymbol{p} \bmod N_\mathrm{m})$. Though the inclusion of the pseudopoints did not alter the number of distinct mesh points N_m^3, it did change the physical size of the cells, as one additional layer of these now has to fit inside the box. That is,

$$H = \frac{L}{N_\mathrm{m}}\,, \qquad \text{(periodic)}$$

$$H = \frac{L}{N_\mathrm{m} - 1}\,. \qquad \text{(non-periodic)}$$

This concludes the setup of the mesh geometry.

Interpolation

With the geometry of mesh fully established, we can begin our study of interpolation. Normally we think of interpolation as the process of assigning a value to a field at $\boldsymbol{r}$, based on known values at nearby grid points $\boldsymbol{r_p}$. However, the concepts needed for this interpolation presents themselves more clearly in the reverse process; distributing the value at $\boldsymbol{r}$ between neighbouring grid points at $\boldsymbol{r_p}$. An example of this reverse interpolation is the construction of the ρ mesh from the particle configuration. We shall stick with this example throughout this subsection. The interpolation schemes developed are however completely general.

To do the interpolation, we first establish "shapes" $S_i(\boldsymbol{r})$ for the particles, which should be thought of as unit mass single particle density fields. Taking all particles i to have the same shape S, but centered around their individual origins $\boldsymbol{r}_i$, we can then write the individual shapes S_i as

$$S_i(\boldsymbol{r}) = S(\boldsymbol{r} - \boldsymbol{r}_i)\,, \tag{2.34}$$

where the shape $S(\boldsymbol{r})$ is some function, localized around the origin. A possible choice for S could be the Plummer density (2.26) with $m = 1$. We then define the mass fraction of particle i assigned to mesh point $\boldsymbol{p}$ to be the overlap of S_i and a cell volume, centered at $\boldsymbol{r_p}$. We denote this fraction by $W_{\boldsymbol{p},i}$, which we may write as a continuous function for each mesh point $\boldsymbol{p}$, evaluated at the particle position:

$$W_{\boldsymbol{p},i} = W_{\boldsymbol{p}}(\boldsymbol{r})|_{\boldsymbol{r}=\boldsymbol{r}_i}\,. \tag{2.35}$$

As no mesh points are special, the assignment functions $W_{\boldsymbol{p}}(\boldsymbol{r})$ may be written as a translation of a global assignment function, in a way analogous to (2.34),

$$W_{\boldsymbol{p}}(\boldsymbol{r}) = W(\boldsymbol{r} - \boldsymbol{r_p})\,. \tag{2.36}$$

We can then write the overlap between S_i and a cell volume centered at $\boldsymbol{r_p}$ as

$$\begin{aligned}
W_{\boldsymbol{p},i} &= \int_{z'_p-H/2}^{z'_p+H/2}\int_{y'_p-H/2}^{y'_p+H/2}\int_{x'_p-H/2}^{x'_p+H/2} S_i\big(\boldsymbol{r}'\big)\,\mathrm{d}x'\,\mathrm{d}y'\,\mathrm{d}z' \\
&= \int_{z'_p-H/2}^{z'_p+H/2}\int_{y'_p-H/2}^{y'_p+H/2}\int_{x'_p-H/2}^{x'_p+H/2} S\big(\boldsymbol{r}'-\boldsymbol{r}_i\big)\,\mathrm{d}x'\,\mathrm{d}y'\,\mathrm{d}z' \\
&= \int \Pi\left(\frac{\boldsymbol{r}'-\boldsymbol{r_p}}{H}\right) S(\boldsymbol{r}'-\boldsymbol{r}_i)\,\mathrm{d}\boldsymbol{r}' \\
\Rightarrow W_{\boldsymbol{p}}(\boldsymbol{r}) &= \int \Pi\left(\frac{\boldsymbol{r}'-\boldsymbol{r_p}}{H}\right) S\big(\boldsymbol{r}'-\boldsymbol{r}\big)\,\mathrm{d}\boldsymbol{r}' \\
\Rightarrow W(\boldsymbol{r}) &= \int \Pi\left(\frac{\boldsymbol{r}'-\boldsymbol{r_p}}{H}\right) S\big(\boldsymbol{r}'-\boldsymbol{r}+\boldsymbol{r_p}\big)\,\mathrm{d}\boldsymbol{r}'\,,
\end{aligned} \tag{2.37}$$

where the definitions (2.34), (2.35) and (2.36) has been used and where Π is the top-hat function, defined by

$$\Pi(\boldsymbol{r}) \equiv \begin{cases} 1 & \text{if all } |x|, |y|, |z| < \frac{1}{2}\,, \\ 0 & \text{otherwize.} \end{cases}$$

The global assignment function $W(\boldsymbol{r})$ cannot depend on the mesh point positions $\boldsymbol{r_p}$, as (2.37) would have us believe. To get rid of this false dependency, we make use the evenness of the top-hat function and perform the translation $\boldsymbol{r'} \to \boldsymbol{r'} - \boldsymbol{r_p}$:

$$\begin{aligned} W(\boldsymbol{r}) &= \int \Pi\left(\frac{\boldsymbol{r'}}{H}\right) S\big(\boldsymbol{r'} - \boldsymbol{r}\big)\, \mathrm{d}\boldsymbol{r'} \\ &= \Pi\left(\frac{\boldsymbol{r}}{H}\right) * S(\boldsymbol{r})\,. \end{aligned} \tag{2.38}$$

Here, the last equality is true for even S only. From here on out, we keep this (rather sensible) restriction on S. Given some S, the mass fraction of particle i to assign the mesh point $\boldsymbol{p}$ can then be found by computing $W_{\boldsymbol{p},i} = W(\boldsymbol{r}_i - \boldsymbol{r_p})$ via (2.38).

With the mass assignment formalized, we still need to choose some S. If we require that W be smooth and that for interparticle separations large compared to H, the errors arising from this discretization of space should become negligible, it can be shown* [13] that S must be of the form

$$S_A(\boldsymbol{r}) = H^{-3A} \overset{A}{\underset{a=1}{\ast}} \Pi\Big(\frac{\boldsymbol{r}}{H}\Big)\,, \tag{2.39}$$

where the big $\ast$ stands for A repeated convolutions of the function to the right. The empty convolution, $A = 0$, is defined to be the delta function† $\delta(\boldsymbol{r})$. By adjusting the order parameter A, a compromise between quality and computational cost can be achieved. This hierarchy of shapes goes on indefinitely, but only the lowest three are commonly seen in numerical codes. Ignoring the normalization factor H^{-3A}, convolving a shape with a top-hat increases its order by a single step (2.39). Convolving with yet another top-hat transforms the shape into the assignment function (2.38). The shape of order $A + 1$ is then proportional to the assignment function of order A, giving us the relations

$$S_{A+1}(\boldsymbol{r}) = H^{-3} W_A(\boldsymbol{r})$$

*The notation used here differ from that in [13], as it is my own.

†The empty convolution convolved with the function should equal the single-function ($A = 1$) convolution. It then follows that the single-function convolution is just the function back again, with no convolution performed.

$$\Rightarrow W_{A+1}(\boldsymbol{r}) = H^{-3}\Pi\Big(\frac{\boldsymbol{r}}{H}\Big) * W_A(\boldsymbol{r})\,. \tag{2.40}$$

The zero-order version of S is simply a delta function without normalization, so that $W_{\boldsymbol{p},i}$ equals 1 for the mesh point $\boldsymbol{p}$ nearest to particle i, and 0 for all others. The mass assignment scheme associated with this choice of A is referred to as the NGP scheme, as the particles are simply snapped to their nearest grid point:

$$\begin{aligned} W_{\text{NGP}}(\boldsymbol{r}) &\equiv W_0(\boldsymbol{r}) \\ &= \Pi\Big(\frac{\boldsymbol{r}}{H}\Big) * \delta(\boldsymbol{r}) \\ &= \Pi\Big(\frac{\boldsymbol{r}}{H}\Big) \\ &= \begin{cases} 1 & \text{if all } |x|, |y|, |z| < \dfrac{H}{2}\,, \\ 0 & \text{otherwize}\,. \end{cases} \end{aligned} \tag{2.41}$$

The NGP scheme is not consistent with the requirement of a smooth W. Regardless of this flaw, it remains a popular assignment scheme.

In the first-order ($A = 1$) scheme, referred to as the cloud in cell (CIC) scheme, the shape is a normalized top-hat. That is, a particle is represented as homogeneous cube with the size of a cell. From (2.40) and (2.41), we have

$$\begin{aligned} W_{\text{CIC}}(\boldsymbol{r}) &\equiv W_1(\boldsymbol{r}) \\ &= H^{-3}\Pi\Big(\frac{\boldsymbol{r}}{H}\Big) * \Pi\Big(\frac{\boldsymbol{r}}{H}\Big) \\ &= \begin{cases} \displaystyle\prod_{w\in\{x,y,z\}} \Big(1 - \frac{|w|}{H}\Big) & \text{if all } |x|, |y|, |z| < H\,, \\ 0 & \text{otherwize,} \end{cases} \end{aligned} \tag{2.42}$$

where the last equality can easily be guessed once you recognize that $W_{\text{CIC}}(\boldsymbol{r})$ must be a linear, even function of all coordinates individually, subject to the conditions $W_{\text{CIC}}(\boldsymbol{0}) = 1$ and $W_{\text{CIC}}(x, y, z) = 0$ if any of $x, y, z \geq H$. The CIC scheme (in just two dimensions, for less clutter) is visualized in Figure 2.6. The GADGET-2 code implements the CIC scheme as its only interpolation method. The same is true for the CONCEPT code, although alternative schemes could easily be implemented.

The last popular interpolation scheme is the case $A = 2$, known as the triangular shaped cloud (TSC) method. From (2.40) and (2.42), we have

$$\begin{aligned} W_{\text{TSC}}(\boldsymbol{r}) &\equiv W_2(\boldsymbol{r}) \\ &= H^{-3}\Pi\Big(\frac{\boldsymbol{r}}{H}\Big) * W_{\text{CIC}}(\boldsymbol{r}) \end{aligned}$$

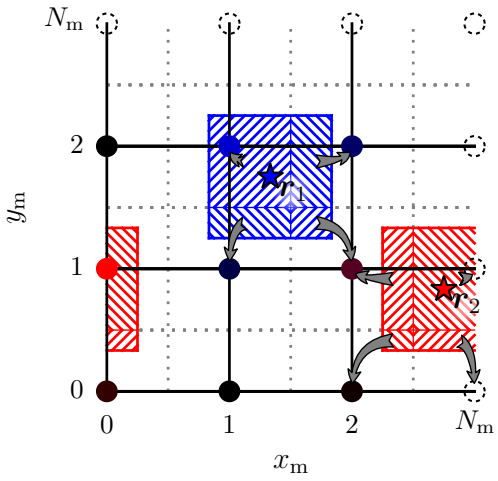


Figure 2.6 – Visualization of the CIC mass assignment scheme on a two-dimensional, periodic mesh with $N_\mathrm{m} = 3$. Two particles are located at $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, respectively, the precise position of which are represented by $\star$ symbols. The particles are given a top-hat shape, corresponding to homogeneous squares with $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ as centers. The particles are colored for clearer distinction. Their areas are composed into rectangular segments, corresponding to the mass fractions assigned to the grid points, the color of which also reflect this assignment. Pseudopoints are unfilled. Part of the shape of the particle centered at $\boldsymbol{r}_2$ sticks through the toroidal boundary, and so its mass is distributed among mesh points lying far away from each other in $\boldsymbol{p}$-space.

$$= \begin{cases} \displaystyle\prod_{w\in\{x,y,z\}} \left(\frac{3}{4} - \frac{w^2}{H^2}\right) & \text{if all } |x|,|y|,|z| \leq \frac{H}{2}\,, \\ \displaystyle\frac{1}{8}\prod_{w\in\{x,y,z\}} \left(\frac{3}{2} - \frac{|w|}{H}\right)^2 & \text{if all } \frac{H}{2} < |x|,|y|,|z| < \frac{3}{2}H\,, \\ 0 & \text{otherwize.} \end{cases} \tag{2.43}$$

The final form of (2.43) is not obvious. It is however easy to check that it fulfills all the criteria; it is a continues, quadratic function in x, y and z, with $W_\mathrm{TSC}(x,y,z) = 0$ if any $x,y,z \geq 3H/2$. Crucially, $W_\mathrm{TSC}(\boldsymbol{r})$ integrates to 1. Note that the linear reach of the assignment function increases by $H/2$ for every step up in order, corresponding to smearing the particles over a larger volume, which makes for smoother interpolation. The number of mesh points taking part in a given interpolation of order A, is $(A+1)^3$.

For the numerical implementation of the PM method, we will need the Fourier transforms of the assignment functions. These are simply convolutions of top-hat functions, with Fourier transforms

$$\begin{aligned} \widetilde{\Pi}(\boldsymbol{k}) &= \int \Pi(\boldsymbol{r}) \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\boldsymbol{r}}\,\mathrm{d}\boldsymbol{r} \\ &= \prod_{w\in\{x,y,z\}} \int_{-1/2}^{1/2} \mathrm{e}^{-\mathrm{i}k_w w}\,\mathrm{d}w \\ &= \prod_{w\in\{x,y,z\}} \int_{-1/2}^{1/2} \cos(k_w w)\,\mathrm{d}w \end{aligned}$$

$$= \prod_{w\in\{x,y,z\}} \operatorname{sinc}\left(\frac{k_w}{2}\right)$$

$$\Rightarrow \mathscr{F}\left[\Pi\left(\frac{\boldsymbol{r}}{H}\right)\right](\boldsymbol{k}) = H^3 \prod_{w\in\{x,y,z\}} \operatorname{sinc}\left(\frac{Hk_w}{2}\right),$$

where $\mathscr{F}$ represents the Fourier transform and sinc is the unnormalized cardinal sine function; $\operatorname{sinc}(x) = \sin(x)/x$. The last equality follows from the similarity theorem for Fourier transforms. From (2.38) and (2.39), we now have

$$W_A(\boldsymbol{r}) = H^{-3A} \overset{A+1}{\underset{a=1}{\ast}} \Pi\left(\frac{\boldsymbol{r}}{H}\right)$$

$$\Rightarrow \widetilde{W}_A(\boldsymbol{k}) = H^{-3A} \prod_{a=1}^{A+1} \mathscr{F}\left[\Pi\left(\frac{\boldsymbol{r}}{H}\right)\right](\boldsymbol{k})$$

$$= H^3 \prod_{w\in\{x,y,z\}} \operatorname{sinc}^{A+1}\left(\frac{Hk_w}{2}\right).$$

Written out in full, our three assignment functions in Fourier space is then

$$\widetilde{W}_{\text{NGP}}(\boldsymbol{k}) = H^3 \operatorname{sinc}\left(\frac{Hk_x}{2}\right) \operatorname{sinc}\left(\frac{Hk_y}{2}\right) \operatorname{sinc}\left(\frac{Hk_z}{2}\right),$$

$$\widetilde{W}_{\text{CIC}}(\boldsymbol{k}) = H^3 \operatorname{sinc}^2\left(\frac{Hk_x}{2}\right) \operatorname{sinc}^2\left(\frac{Hk_y}{2}\right) \operatorname{sinc}^2\left(\frac{Hk_z}{2}\right),$$

$$\widetilde{W}_{\text{TSC}}(\boldsymbol{k}) = H^3 \operatorname{sinc}^3\left(\frac{Hk_x}{2}\right) \operatorname{sinc}^3\left(\frac{Hk_y}{2}\right) \operatorname{sinc}^3\left(\frac{Hk_z}{2}\right).$$

This concludes the development of mesh interpolation, which is used in the PM method for mass assignment and force interpolation. We shall deal with the exact numerical implementation of these interpolations later.

Finite Differencing

Viewing the mesh as a continuous field sampled at the mesh points, the notion of differentiation remain meaningful. From the definition of differentiation, we have

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \lim_{H\to 0} \frac{f(x+H) - f(x)}{H} \tag{2.44}$$

$$\Rightarrow \hat{D}_{\mathrm{f}} f_{\mathrm{m}}(x_p) = \frac{f_{\mathrm{m}}(x_p + H) - f_{\mathrm{m}}(x_p)}{H}, \tag{2.45}$$

where f is some continuous function of x, while f_{m} is a discrete function only defined on the (one-dimensional) mesh points x_p, with $f_{\mathrm{m}}(x_p) = f(x_p)$. In the last equation, H is the mesh point separation, and the normal differentiation

operator has been substitutes for the finite difference operator $\hat{D}_\mathrm{f}$. Equation (2.45) is the most direct translation from continuous differentiation to discrete differencing. It does not correspond to the exact derivative of the continuous function, but only approximates it; $\hat{D}_\mathrm{f} f_\mathrm{m}(x_p) \approx \mathrm{d}f(x)/\,\mathrm{d}x|_{x=x_p}$. The differencing scheme in equation (2.45) is referred to as *forward* differencing (hence the subscript on the difference operator), because it requires the evaluation of f_m at $x_p + H$ but not $x_p - H$. The corresponding backwards difference is of course equally valid, though not equal to the forward difference. It is straight forward to construct a central (symmetrical) difference, which is what is most often used in practice:

$$\hat{D}_2 f_\mathrm{m}(x_p) = \frac{f_\mathrm{m}(x_p + H) - f_\mathrm{m}(x_p - H)}{2H}\,, \tag{2.46}$$

where the subscript on the difference operator now refers to its order of accuracy. Equation (2.46) does not have the same form as the limit in (2.44), but it still corresponds to a discrete differentiation, as the corresponding central differentiation, the forward differentiation (2.44) and the backward differentiation are all equal.

If the sampled field is smooth, values closer to the actual differentiated field may be achieved by gathering information not just from the two neighbouring points, but from a larger region. How to generalize the above procedure of central differences to incorporate more than two points is not immediately clear. We can construct higher order methods by hand by utilizing the Taylor expansion of f. To construct the four-point method, where the points $x_p \pm H = x_{p\pm1}$ and $x_p \pm 2H = x_{p\pm2}$ are used to approximate the derivative at x_p, we write down the fourth-order Taylor expansion of f around x_p, evaluated in these four points:

$$\begin{aligned}
f(x_{p\pm1}) \approx f(x_p) &\pm H \left.\frac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x=x_p} + \frac{H^2}{2}\left.\frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2}\right|_{x=x_p} \\
&\pm \frac{H^3}{6}\left.\frac{\mathrm{d}^3 f(x)}{\mathrm{d}x^3}\right|_{x=x_p} + \frac{H^4}{24}\left.\frac{\mathrm{d}^4 f(x)}{\mathrm{d}x^4}\right|_{x=x_p}\,, \\
f(x_{p\pm2}) \approx f(x_p) &\pm 2H \left.\frac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x=x_p} + 2H^2\left.\frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2}\right|_{x=x_p} \\
&\pm \frac{4H^3}{3}\left.\frac{\mathrm{d}^3 f(x)}{\mathrm{d}x^3}\right|_{x=x_p} + \frac{2H^4}{3}\left.\frac{\mathrm{d}^4 f(x)}{\mathrm{d}x^4}\right|_{x=x_p}\,.
\end{aligned}$$

We see that the even terms all have a positive sign, regardless of which point — forward or backward — f is evaluated in. We can thus get rid of the even terms by the following subtractions:

$$f(x_{p+1}) - f(x_{p-1}) \approx 2H \left.\frac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x=x_p} + \frac{H^3}{3}\left.\frac{\mathrm{d}^3 f(x)}{\mathrm{d}x^3}\right|_{x=x_p}$$

$$f(x_{p+2}) - f(x_{p-2}) \approx 4H \left.\frac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x=x_p} + \frac{8H^3}{3} \left.\frac{\mathrm{d}^3 f(x)}{\mathrm{d}x^3}\right|_{x=x_p} .$$

We can now eliminate the terms with the third derivative of f by calculating $8[f(x_{p+1}) - f(x_{p-1}] - [f(x_{p+2}) - f(x_{p-2})]$. Isolating the first derivative of f from this calculation, we end up with

$$\begin{aligned}
\left.\frac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x=x_p} &\approx \frac{f(x_{p-2}) - 8f(x_{p-1}) + 8f(x_{p+1}) - f(x_{p+2})}{12H} \\
&= \frac{f_{\mathrm{m}}(x_{p-2}) - 8f_{\mathrm{m}}(x_{p-1}) + 8f_{\mathrm{m}}(x_{p+1}) - f_{\mathrm{m}}(x_{p+2})}{12H} \\
&\equiv \hat{D}_4 f_{\mathrm{m}}(x_p) \,. \qquad (2.47)
\end{aligned}$$

In the CONCEPT code as well as the GADGET-2 code, fourth-order differencing (2.47) is used for approximating the derivative of the mesh-sampled potential in the PM method. The mesh used in the codes are three-dimensional, and so we need to construct a vector difference operator $\hat{\boldsymbol{D}}$. As we can simply differentiate each dimension separately, we have

$$\hat{\boldsymbol{D}} f_{\mathrm{m}}(\boldsymbol{r}) = \left(\hat{D}^x f_{\mathrm{m}}(\boldsymbol{r}), \hat{D}^y f_{\mathrm{m}}(\boldsymbol{r}), \hat{D}^z f_{\mathrm{m}}(\boldsymbol{r})\right) , \qquad (2.48)$$

where superscripts explicitly states the variable with which to do the differencing with respect to.

Constructing higher order central differencing methods in the same way as used above is tedious. The following formula [15] can be used to produce central difference methods to any order A directly:

$$\hat{D}_A f_{\mathrm{m}}(x_p) = \frac{1}{H} \sum_{\substack{j=-A/2 \\ j\neq 0}}^{A/2} \frac{\mathrm{d}}{\mathrm{d}\xi} \prod_{\substack{i=-A/2 \\ i\neq j}}^{A/2} \left.\frac{\xi - i}{j - i}\right|_{\xi=0} f_{\mathrm{m}}(x_{p+j}) \,. \qquad (2.49)$$

Note that A must be even, corresponding to evaluating f_{m} in the same number of forward and backward points.

2.2.3 The Force Computation

With the mesh operations formalized, we can construct the algorithm for the PM method, computing forces $\boldsymbol{F}_i$ based on particle masses m_i and positions $\boldsymbol{r}_i$. The standard representation of the particles is the Lagrangian description, where each particle is located at a specific point. We can write this set of masses and positions $\{m_i, \boldsymbol{r}_i\}$ as a proper density field, as

$$\rho(\boldsymbol{r}) = \sum_{i=1}^{N} m_i \delta(\boldsymbol{r} - \boldsymbol{r}_i) \,.$$

Since each particle occupy an infinitesimal amount of space, the density goes to infinity at $\boldsymbol{r} = \boldsymbol{r}_i$. In the Eulerian description, space is discretized, which means that the density value at each mesh point must be finite. In fact, the density at the mesh points must be the mass enclosed within a mesh cell centered at the point, divided by the cell volume. The enclosed mass depends on the shape of the particle, as described in the last subsection. The fraction of the mass of particle i enclosed in the cell volume is given by $W_{\boldsymbol{p},i}$. The mesh defined densities $\rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}})$ are then given by

$$\begin{aligned}\rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}}) &= \sum_{i=1}^{N} W_{\boldsymbol{p},i} \frac{m_i}{H^3} \\ &= H^{-3} \sum_{i=1}^{N} W(\boldsymbol{r}_i - \boldsymbol{r}_{\boldsymbol{p}}) m_i \\ &= H^{-3} \int W(\boldsymbol{r} - \boldsymbol{r}_{\boldsymbol{p}}) \sum_{i=1}^{N} m_i \delta(\boldsymbol{r} - \boldsymbol{r}_i) \, \mathrm{d}\boldsymbol{r} \\ &= \frac{W(\boldsymbol{r}_{\boldsymbol{p}}) * \rho(\boldsymbol{r}_{\boldsymbol{p}})}{H^3} \,, \end{aligned} \tag{2.50}$$

where the evenness of the assignment function W has been used to write the convolution. In the numerical implementation, the mesh are assigned the values $\rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}})$, in a manner similar to the first equality in (2.50). It is then from the $\rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}})$ values that the Poisson equation should be solved.

Equation (2.50) is easily cast into Fourier space;

$$\widetilde{\rho}_{\mathrm{m}}(\boldsymbol{k}) = \frac{\widetilde{W}(\boldsymbol{k}) \widetilde{\rho}(\boldsymbol{k})}{H^3} \,. \tag{2.51}$$

Some caution should be taken when constructing this Fourier transform, as the mesh defined density ρ_{m} should be considered discrete. It is only defined at the mesh points $\boldsymbol{r}_{\boldsymbol{p}}$. The transform should then be done as a series, not as an integral. To do an integral transform, we would need a continuously defined density. We can construct such a density by allowing ρ_{m} to take any position as argument*, $\rho_{\mathrm{m}}(\boldsymbol{r})$, and multiply it by a Dirac comb Ш:

$$\rho_{\text{Ш}}(\boldsymbol{r}) = \rho_{\mathrm{m}}(\boldsymbol{r}) \text{Ш}\Big(\frac{\boldsymbol{r}}{H}\Big) \,,$$

where the Dirac comb is defined by

$$\begin{aligned} \text{Ш}(\boldsymbol{\varsigma}) &\equiv \sum_{n \in \mathbb{Z}^3} \delta(\boldsymbol{\varsigma} - \boldsymbol{n}) \\ \Rightarrow \text{Ш}\Big(\frac{\boldsymbol{r}}{H}\Big) &= H^3 \sum_{n \in \mathbb{Z}^3} \delta(\boldsymbol{r} - H\boldsymbol{n}) \end{aligned}$$

*Clearly, (2.50) is mathematically well defined for all $\boldsymbol{r}$.

$$= H^3 \sum_{\boldsymbol{p}} \delta(\boldsymbol{r} - \boldsymbol{r}_{\boldsymbol{p}})$$

$$\Rightarrow H^3 \rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}}) = \int_{z_p - H/2}^{z_p + H/2} \int_{y_p - H/2}^{y_p + H/2} \int_{x_p - H/2}^{x_p + H/2} \rho_{\mathrm{III}}(\boldsymbol{r}) \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \,,$$

where the left-hand side of the last equality can be interpreted as the mass assigned to mesh point $\boldsymbol{p}$. The continuously defined $\rho_{\mathrm{III}}(\boldsymbol{r})$ is thus a sum of delta functions, just like $\rho(\boldsymbol{r})$. One may argue that it would be better to use $\rho_{\mathrm{III}}(\boldsymbol{r})$ — which is really the original, Lagrangian $\rho(\boldsymbol{r})$, interpolated to the mesh points — rather than $\rho_{\mathrm{m}}(\boldsymbol{r})$, to do the mesh computations, solving the Poisson equation. It turns out however, that their Fourier transforms are identical:

$$\begin{aligned}
\widetilde{\rho}_{\mathrm{III}}(\boldsymbol{k}) &= \int \rho_{\mathrm{III}}(\boldsymbol{r}) \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\boldsymbol{r}} \, \mathrm{d}\boldsymbol{r} \\
&= H^3 \int \sum_{\boldsymbol{p}} \rho_{\mathrm{m}}(\boldsymbol{r}) \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\boldsymbol{r}} \delta(\boldsymbol{r} - \boldsymbol{r}_{\boldsymbol{p}}) \, \mathrm{d}\boldsymbol{r} \\
&= H^3 \sum_{\boldsymbol{p}} \rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}}) \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\boldsymbol{r}_{\boldsymbol{p}}} \\
&= \widetilde{\rho}_{\mathrm{m}}(\boldsymbol{k}) \,. \qquad (2.52)
\end{aligned}$$

Reassuringly, we do not have to worry about errors due to the transformation from the Lagrangian to the Eulerian description, or vice versa.

We now Fourier transform the mesh defined densities via an FFT:

$$\widetilde{\rho}_{\mathrm{m}}(\boldsymbol{k}_{\boldsymbol{p}}) = \mathrm{FFT}\big(\rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}})\big) \,. \qquad (2.53)$$

where $\boldsymbol{k}_{\boldsymbol{p}}$ are the discrete mesh points in Fourier space, $\boldsymbol{k}_{\boldsymbol{p}} = (2\pi/L)\boldsymbol{p}$. It should be noted that $\widetilde{\rho}_{\mathrm{m}}(\boldsymbol{k})$ as constructed in (2.53) corresponds to the Fourier transform of the *periodic* mesh defined density. Without ever worrying about it, periodicity is achieved for free. The Fourier transformed density values $\widetilde{\rho}_{\mathrm{m}}(\boldsymbol{k}_{\boldsymbol{p}})$ are those resulting from the interpolated density. We can use (2.51) to achieve the density values resulting from the *original*, non-interpolated density:

$$\widetilde{\rho}(\boldsymbol{k}_{\boldsymbol{p}}) = \frac{H^3}{\widetilde{W}(\boldsymbol{k}_{\boldsymbol{p}})} \mathrm{FFT}\big(\rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}})\big) \,. \qquad (2.54)$$

Even though $\widetilde{\rho}(\boldsymbol{k}_{\boldsymbol{p}})$ is the Fourier transform of the non-interpolated density, it is still only defined for values of $\boldsymbol{k}$ on the (Fourier transformed) mesh, $\boldsymbol{k} = \boldsymbol{k}_{\boldsymbol{p}}$.

The next step is to solve the Poisson equation in Fourier space, as in (2.33). That is,

$$\widetilde{\phi}(\boldsymbol{k}_{\boldsymbol{p}}) = -\frac{4\pi G}{\boldsymbol{k}^2} \widetilde{\rho}(\boldsymbol{k}_{\boldsymbol{p}}) \,, \qquad (2.55)$$

where again, the potential values $\widetilde{\phi}(\boldsymbol{k_p})$ are only defined on the mesh. We can transform these back to coordinate space by an inverse FFT,

$$\phi(\boldsymbol{r_p}) = \text{FFT}^{-1}\big(\widetilde{\phi}(\boldsymbol{k_p})\big)\,.$$

With the potential values at the mesh points, finite differencing is used to compute the forces at these points. Since the mesh point masses do not correspond to the masses of the particles, we will find the force per unit mass; the gravitational field $\boldsymbol{g}$. The differencing is done via the difference operator $\hat{\boldsymbol{D}}$ as described in the previous subsection:

$$\boldsymbol{g}(\boldsymbol{r_p}) = -\hat{\boldsymbol{D}}\phi(\boldsymbol{r_p})\,. \tag{2.56}$$

In the actual CONCEPT code, the fourth-order accuracy differencing operator $\hat{\boldsymbol{D}}_4$ is used.

To interpolate the accelerations from the mesh points and back to the particle positions, the same interpolation scheme (the same order of W) as that used in the mass assignment, must be used. If not, momentum conservation will be perturbed [13]. The accelerations become

$$\begin{aligned}
\boldsymbol{g}(\boldsymbol{r}_i) &= \sum_{\boldsymbol{p}} W_{\boldsymbol{p},i}\boldsymbol{g}(\boldsymbol{r_p}) \\
&= \sum_{\boldsymbol{p}} W(\boldsymbol{r}_i - \boldsymbol{r_p})\boldsymbol{g}(\boldsymbol{r_p}) \\
&= \int W(\boldsymbol{r}_i - \boldsymbol{r}) \sum_{\boldsymbol{p}} \boldsymbol{g}(\boldsymbol{r})\delta(\boldsymbol{r} - \boldsymbol{r_p})\,\mathrm{d}\boldsymbol{r} \\
&= H^{-3}\int W(\boldsymbol{r}_i - \boldsymbol{r})\boldsymbol{g}_{\text{III}}(\boldsymbol{r})\,\mathrm{d}\boldsymbol{r} \\
&= \frac{W(\boldsymbol{r}_i) * \boldsymbol{g}_{\text{III}}(\boldsymbol{r}_i)}{H^3}\,,
\end{aligned} \tag{2.57}$$

where

$$\boldsymbol{g}_{\text{III}}(\boldsymbol{r}) = \boldsymbol{g}(\boldsymbol{r})\text{III}\Big(\frac{\boldsymbol{r}}{H}\Big)\,. \tag{2.58}$$

Needless to say, the forces are then finally given by

$$\boldsymbol{F}_i = m_i\boldsymbol{g}(\boldsymbol{r}_i)\,.$$

Were we to combine all the above steps — from the mass assignment to the force interpolation — into a single expression, we would see that it is a single string of multiplications, convolutions and Fourier transforms, together with the difference operator. These multiplications and convolutions can safely travel inside the Fourier transforms, where they change role with one another (and become inversely transformed, relative to the given transformation). The

assignment function W from (2.58) can thus be replaced with a $\widetilde{W}$ in Fourier space. Additionally, the Dirac comb also from (2.58) can merge together with ρ_{m} to form $\rho_{\text{Ш}}$, which we know from (2.52) has equal Fourier transforms.

2.2.4 Numerical Implementation

All steps needed for the force calculation in the PM method were developed in the last subsection. We now need to put it all together, to form the numerical implementation. Here follows a list like the one we encountered in subsection 2.2.1, describing each step of the PM method. With our detailed knowledge from the previous subsection, we can now write a fully detailed list of the steps.

1. Assign particle masses to the mesh points, resulting in $\rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}})$. The total mass assigned to mesh point $\boldsymbol{p}$ is described by (2.50). It is however much more feasible to assign the mass of each particle in turn to its nearby mesh points. We can write this assignment as

$$\{m_i, \boldsymbol{r}_i\} \to \Delta_i \rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}}) = W_{\boldsymbol{p},i} \frac{m_i}{H^3}\,,$$

where $\Delta_i \rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}})$ is the added density value to $\rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}})$ from particle i:

$$\rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}}) = \sum_i \Delta_i \rho_{\mathrm{m}}(\boldsymbol{r}_{\boldsymbol{p}})\,.$$

The mesh points $\boldsymbol{p}$ considered in the assignment is the $(A+1)^3$ points near* $\boldsymbol{r}_i$. For the CIC interpolation, the eight mesh points sharing the mass of particle i is then

$$\boldsymbol{p}_{\mathrm{CIC}} = \left(\left\lfloor \frac{x_i}{H} \right\rceil_x, \left\lfloor \frac{y_i}{H} \right\rceil_y, \left\lfloor \frac{z_i}{H} \right\rceil_z \right), \tag{2.59}$$

where each occurrence of $\lfloor \bullet \rceil$ can be chosen to be either a floor or a ceiling function. The subscript on these are used only for distinction, so that later one can refer back to a particular instance. The values of the CIC assignment function can now be neatly written as

$$W_{\boldsymbol{p}_{\mathrm{CIC}},i} = \prod_{w \in \{x,y,z\}} \left| \frac{w_i}{H} - \left\lfloor \frac{w_i}{H} \right\rceil_w \right|, \tag{2.60}$$

which is the numerically most effective way of computing the weights in the CIC interpolation.

*These points are within a distance of $(A+1)H/2$ to $\boldsymbol{r}_i$ in at least one of the three dimensions. The points are then situated in a cube, not a sphere. Thus generally, it is not the nearest $(A+1)^3$ points which take part in the assignment.

The values assigned to the pseudopoints $\boldsymbol{p}_{\text{ps}}$ of (2.59) — corresponding to points where at least one of $w_{\text{m}} = \lceil w_i/H \rceil = N_{\text{m}}$, $w \in \{x, y, z\}$ — now need to be added to the existing values of the respective real points:

$$\rho_{\text{m}}(\boldsymbol{r}_{(\boldsymbol{p}_{\text{ps}} \bmod N_{\text{m}})}) \to \rho_{\text{m}}(\boldsymbol{r}_{(\boldsymbol{p}_{\text{ps}} \bmod N_{\text{m}})}) + \rho_{\text{m}}(\boldsymbol{r}_{\boldsymbol{p}_{\text{ps}}})\,,$$

which takes care of the periodicity in the mass assignment. For now, the pseudopoints has fulfilled their purpose and must be disregarded from this point on, as these now contain redundant information.

2. Fourier transform $\rho_{\text{m}}(\boldsymbol{r}_{\boldsymbol{p}}) \xrightarrow{\text{FFT}} \widetilde{\rho}_{\text{m}}(\boldsymbol{k}_{\boldsymbol{p}})$, where $\boldsymbol{k}_{\boldsymbol{p}} = (2\pi/L)\boldsymbol{p}$. This transform should be done in-place, replacing the density values on the mesh with the values from the Fourier transform.

3. While in Fourier space we need to do several things, which all come down to simple multiplications. To deconvolve for the mass assignment (transforming $\rho_{\text{m}}(\boldsymbol{k}_{\boldsymbol{p}})$ to $\rho(\boldsymbol{k}_{\boldsymbol{p}})$), we use (2.54). We know that another interpolation (2.57) awaits, so we might as well do its deconvolution while in Fourier space, as multiplication is easier to do than convolution. We also need to solve the Poisson equation, transforming the mesh values to potential values. Here we use (2.55). Doing all this at once, we have

$$\widetilde{\phi}'(\boldsymbol{k}_{\boldsymbol{p}}) = -\frac{4\pi G}{\boldsymbol{k}_{\boldsymbol{p}}^2} \frac{H^6}{\widetilde{W}^2(\boldsymbol{k}_{\boldsymbol{p}})} \widetilde{\rho}(\boldsymbol{k}_{\boldsymbol{p}})\,, \tag{2.61}$$

where the prime denotes that this is not the actual Fourier transformed potential, as it has been deconvolved with the assignment function one additional time.

The last thing we need to do before leaving Fourier space is to manually set $\widetilde{\phi}'(\boldsymbol{0}) = 0$. This is not just because (2.61) blows up at $\boldsymbol{k}_{\boldsymbol{p}} = \boldsymbol{0}$, but more importantly because this act corresponds to subtracting the mean density, as we have discussed before. This is crucial as the Poisson equation with periodic boundaries is not solvable unless the density *contrast* is used. If we do not explicitly subtract the mean density, an inverse FFT of $\widetilde{\phi}'(\boldsymbol{k}_{\boldsymbol{p}})$ will result in an error under many FFT implementations.

4. Fourier transform $\widetilde{\phi}'(\boldsymbol{k}_{\boldsymbol{p}}) \xrightarrow{\text{FFT}^{-1}} \phi'(\boldsymbol{r}_{\boldsymbol{p}})$. As with the forward Fourier transform, this transform should be done in-place. We now need to manually reassign values to the pseudopoints, as these are needed for the upcoming interpolation:

$$\phi'(\boldsymbol{r}_{\boldsymbol{p}_{\text{ps}}}) \to \phi'(\boldsymbol{r}_{(\boldsymbol{p}_{\text{ps}} \bmod N_{\text{m}})})\,.$$

That is, their values are simply copies of the values of their respective real points.

The next step involved the finite difference operator $\hat{\boldsymbol{D}}$, defined generally by (2.48) and (2.49). It is clear this operator can only approximate

the derivative at points which have $A/2$ neighbour points, where A is the order of accuracy of the finite difference operator. For $\hat{\boldsymbol{D}}$ to work, we thus have to construct additional pseudopoints around the entire mesh. These extra pseudopoints are referred to as ghost points. For $\hat{\boldsymbol{D}}$ to be applicable to all normal points and the original pseudopoints, we need to wrap the mesh with $A/2$ layers of ghost points, covering all six faces of the mesh. Ghost points $\boldsymbol{p}_\mathrm{g}$ then have one $-A/2 \leq w_\mathrm{m} < 0$ or $N_\mathrm{m} < w_\mathrm{m} \leq N_\mathrm{m} + A/2$, $w \in \{x, y, z\}$. The values of these ghost points $\boldsymbol{p}_\mathrm{g}$ should naturally be copies of the actual points as demanded by the periodicity of the mesh:

$$\phi'(\boldsymbol{r}_{\boldsymbol{p}_\mathrm{g}}) = \phi'(\boldsymbol{r}_{(\boldsymbol{p}_\mathrm{g} \bmod N_\mathrm{m})}) \,.$$

With the $6(N_\mathrm{m}+1)^2$ ghost points added to the mesh, the original points (including pseudopoints) can be finitely differenced.

5. Compute the gravitational field at the mesh points (excluding ghosts) as in (2.56),

$$\boldsymbol{g}'(\boldsymbol{r}_{\boldsymbol{p}}) = -\hat{\boldsymbol{D}}\phi'(\boldsymbol{r}_{\boldsymbol{p}}) \,,$$

where one possible choice for the finite difference operator $\hat{\boldsymbol{D}}$ would be the fourth-order accurate (2.47). This operator uses the ghost points to perform the differencing of the outermost points. When $\boldsymbol{g}'(\boldsymbol{r}_{\boldsymbol{p}})$ have been found, the ghost points are no longer needed.

We cannot convert the mesh itself into the gravitational field $\boldsymbol{g}'$ as it is a vector field. We could create $\boldsymbol{g}'$ as three meshes, equivalent to a $N_\mathrm{m} \times N_\mathrm{m} \times N_\mathrm{m} \times 3$ mesh, but it would be expensive in memory. Instead we create just a single additional mesh and compute and store one component of $\boldsymbol{g}'$ on it at a time. The sixth and last step should then really we done three times over, once for each component of $\boldsymbol{g}'$.

6. Interpolate the gravitational field at the mesh points (including pseudopoints) back to particle positions, in a manner similar to the mass assignment in step 1:

$$\{\boldsymbol{g}'(\boldsymbol{r}_{\boldsymbol{p}})\} \to \Delta_{\boldsymbol{p}}\boldsymbol{g}_i = W_{\boldsymbol{p},i}\boldsymbol{g}'(\boldsymbol{r}_{\boldsymbol{p}}) \,,$$
$$\boldsymbol{g}_i = \sum_{\boldsymbol{p}} \Delta_{\boldsymbol{p}}\boldsymbol{g}_i \,,$$

where the interpolation removes the prime, resulting in the actual acceleration at each particle position. For the choice of CIC interpolation, it is again only the eight neighbour points (2.59) which contribute to the acceleration of a specific particle, and the weights can be efficiently computed by (2.60). The forces are now trivially given by $\boldsymbol{F}_i = m_i\boldsymbol{g}_i$.

This concludes our development of the the PM method.

We wish to compare the results of a PM simulation against those of a PP simulation. To compare the raw particle configurations resulting from such simulation directly is difficult*. It is better to compute the power spectrum of each configuration and then compare those. We have deliberately postponed the description of how to go about computing these power spectra until now, as the computation is mesh based. As defined in (1.42), the power spectrum is essentially the absolute square of the Fourier transform of the density contrast. To compute $P(|\boldsymbol{k}|)$ then, we simply construct† the mesh $\rho_{\mathrm{m}}(\boldsymbol{r_p})$ as in step 1 above, Fourier transform it and deconvolve for the interpolation kernel W, by dividing with $\widetilde{W}(\boldsymbol{k})$:

$$P(\boldsymbol{k_p}) = \left| \frac{H^3}{\widetilde{W}(\boldsymbol{k_p})} \mathrm{FFT}\big(\rho_{\mathrm{m}}(\boldsymbol{r_p})\big) \right|^2 .$$

The cell spacing H then limits the resolution of the power spectrum, regardless of how finely resolved the particle configuration might be. This explains why the PP power spectrum Figure 2.5 did not resolve scales comparable to the Plummer radius ϵ.

With the power spectrum fully established, let us compare the PM method to the PP method. The two methods should give similar results, though we expect them to differ at small scales, where the PM method becomes inaccurate. Figure 2.7 shows such a comparison, where we see that the PM method produces too little structure at small scales, relative to the PP method. At large scales though, they match almost perfectly. For the particular simulations used for Figure 2.7, we see that switching to the PM method reduces the effective resolution with about one order of magnitude. Although the PM method provide a much needed speedup over the PP method, this costs is too large for it to be much useful as is.

2.3 Hybrid Methods

By now we have constructed two very different gravitational solvers; the PP method and the PM method. The first was a very direct numerical implementation of Newton's law of universal gravitation, which describe gravity as a pairwise force between particles. As this description lend itself directly to numerical implementation, the resulting PP method is virtually exact. Its complexity scaling $\mathcal{O}\big(N^2\big)$ however, itself the result of the pairwise description of gravity, limited the usefulness of the PP method drastically, as only simulations with small N are viable. The PM method resides in the other end

*Although this is how the CONCEPT code checks that it gives results consistent with those of GADGET-2.

†This process might be considered inverse to the creation of the initial particle configuration, which is done by perturbing a perfect grid of particles according to the power spectrum of the early Universe.

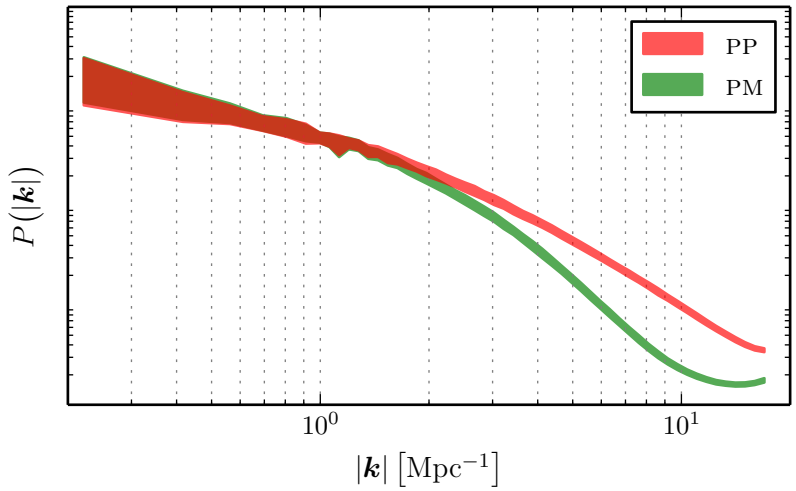


Figure 2.7 – Power spectra of PP (red) and PM (green) simulations with identical initial conditions. The specifics of the simulations are equal to those of Figure 2.4. The PP power spectrum is identical to that Figure 2.5, but logarithmically binned to smooth out the plot. The height of the filled area corresponds to the standard deviation of the data within each bin. The very large $|\boldsymbol{k}|$ regime, where in Figure 2.5 the power spectrum started to increase with $|\boldsymbol{k}|$, has been removed. Although the largest $|\boldsymbol{k}|$ has been left out, it is clear that the PM power spectrum include the same upwards bend as we have seen for the PP power spectrum. Both power spectra as well as the PM run itself was computed with a mesh with $N_\text{m} = 512$.

of the spectrum, with the scaling $\mathcal{O}(N \log N)$. Here, accuracy (and memory) is sacrificed to gain efficiency, as gravity is mediated to the particles via an intermediate gravitational field, itself constructed from the particle distribution. This field is numerically represented as a mesh, which explicitly imposes a resolution limit to gravity. Although the PM method is fast, the resolution penalty as shown in Figure 2.7 severely limits its usefulness. It is possible to combine the two methods into one, giving us the resolution power of the PP method and a speed comparable to the PM method. Such methods are collectively known as hybrid methods. In this work we discuss in detail only the most direct hybrid, the P^{3}M method, which is implemented in the CONCEPT code. Other, more effective hybrids do exist, the idea of which we will only briefly describe.

2.3.1 The Particle-Particle-Particle-Mesh Method

The particle-particle-particle-mesh (P^{3}M) method — as its name suggests — is a hybrid between the PP and the PM method. The P^{3}M method combines the two other methods in their entirety, pulling from all the tricks that we have established so far. It uses mesh based methods to compute gravity at

large scales, while the finer, small scale gravitation is supplied by particle-particle methods. To do this, gravity needs to be split into a long-range and a short-range component, just as when doing Ewald summation.

In the Ewald mehtod, the force split was introduces at the level of the Green's function for the Laplace operator in the Poisson equation. From (2.14) and (2.16), we had

$$\begin{aligned} \mathcal{G}_{\mathrm{s}}(\boldsymbol{r}) &= -\frac{1}{4\pi|\boldsymbol{r}|}\operatorname{erfc}\left(\frac{|\boldsymbol{r}|}{2r_{\mathrm{s}}}\right), \\ \widetilde{\mathcal{G}}_{\mathrm{l}}(\boldsymbol{k}) &= -\frac{1}{\boldsymbol{k}^2}\exp\bigl(-\boldsymbol{k}^2 r_{\mathrm{s}}^2\bigr), \end{aligned} \tag{2.62}$$

where r_{s} is the distance scale of the force split and $\mathcal{G}_{\mathrm{s}} + \mathcal{G}_{\mathrm{l}} = \mathcal{G}_{\nabla^2}$ solves the Poisson equation in triply periodic boundaries, as in (2.11),

$$\phi(\boldsymbol{r}) = 4\pi G \sum_{j=1}^{N} m_j \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \mathcal{G}_{\nabla^2}(\boldsymbol{r}_j + \boldsymbol{n}L - \boldsymbol{r}). \tag{2.63}$$

The long range part $\widetilde{\mathcal{G}}_{\mathrm{l}}$ was written in Fourier space, because the convergence rate of the sum in (2.63) was much higher if done in Fourier space, for this part. The reason for the slowness of the PP method was that each particle had to be paired with every other, due to the long-range nature of gravity. The central idea of the $\mathrm{P}^3\mathrm{M}$ method is to delegate the computation of the long-range component of gravity to the PM method, leaving only the short-range component to be done with the slower PP method. This is ideal, as the the fast PM method is accurate only at large scales. In itself though, this does not speed up the overall computation, as all particles must still be paired with one another in the short-range PP computation. However, since the short-range force falls off much faster than r^{-2}, particle pairs with large inter-particle distances (compared to r_{s}) contribute a negligible amount to the total short-range force. Thus we only need to compute the short-range force on a particle due to its neighbouring particles.

The long-range component of gravity is then computed through the same six steps which comprises the PM method, as described in subsection 2.2.4. The potential (2.61) should then be substituted for the short-range potential;

$$\begin{aligned} \widetilde{\phi}'(\boldsymbol{k_p}) &\to \widetilde{\phi}'_{\mathrm{s}}(\boldsymbol{k_p}) \\ &= -\frac{4\pi G}{\boldsymbol{k_p}^2}\exp\bigl(-\boldsymbol{k_p}^2 r_{\mathrm{s}}^2\bigr)\frac{H^6}{\widetilde{W}^2(\boldsymbol{k_p})}\widetilde{\rho}(\boldsymbol{k_p}), \end{aligned} \tag{2.64}$$

where $\widetilde{\mathcal{G}}_{\nabla^2}(\boldsymbol{k}) = -1/\boldsymbol{k}^2$ has been substituted for $\widetilde{\mathcal{G}}_{\mathrm{l}}(\boldsymbol{k})$ from (2.62). Renaming the remaining variables to reflect that they are concerned with the long-range component only, $\phi' \to \phi'_{\mathrm{l}}$, $\boldsymbol{g}' \to \boldsymbol{g}'_{\mathrm{l}}$, $\boldsymbol{g}_i \to \boldsymbol{g}_{\mathrm{l},i}$, $\boldsymbol{F}_i \to \boldsymbol{F}_{\mathrm{l},i}$, the six steps of the PM algorithm modified only by (2.64) will result in the long-range forces $\boldsymbol{F}_{\mathrm{l},i}$.

Since we only account for nearby particles with regards to the short-range force, we also disregard periodicity. This is not a problem as periodicity is accounted for in the long-range component, which is where it matters. The short-range P^{3}M force is then the short-range term of the Ewald-force (2.19), but without the sum over periodic images:

$$\boldsymbol{F}_{\mathrm{s},i} = Gm_i \sum_{\substack{j=1 \\ j\neq i}}^{N} m_j \frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^3} \left[\operatorname{erfc}\left(\frac{|\boldsymbol{r}_j - \boldsymbol{r}_i|}{2r_\mathrm{s}} \right) + \frac{|\boldsymbol{r}_j - \boldsymbol{r}_i|}{\sqrt{\pi}\, r_\mathrm{s}} \mathrm{e}^{-\frac{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2}{4r_\mathrm{s}^2}} \right] . \tag{2.65}$$

The full periodic force is then $\boldsymbol{F}_i = \boldsymbol{F}_{\mathrm{l},i} + \boldsymbol{F}_{\mathrm{s},i}$. When we discussed Ewald summation, we chose r_s as in (2.21), which led to fast convergence of both the short-range and the long-range sum. As these sums do not occur in the P^{3}M method, this choice is no longer advantageous. Preferably, r_s should be made extremely small, in order to delegate as much work as possible to the efficient long-range computation. We cannot make $r_\mathrm{s} \lesssim H$ though, as the mesh anisotropies will begin to show up as errors in the mesh-based long-range force. As noted in [1], r_s does not have to be much larger than H in order for these anisotropic force errors to be well below 1 %, in a root-mean-square sense. In fact in the GADGET-2 code,

$$r_\mathrm{s} = 1.25H\,,$$

which is also adopted as the standard in the CONCEPT code, although it can easily be changed in the parameter file.

Using (2.65) in the P^{3}M method would be a disaster, as it involves the pairwise sum responsible for the $\mathcal{O}\big(N^2\big)$ complexity. We wish to approximate this sum by simply neglecting all the terms for which $|\boldsymbol{r}_j - \boldsymbol{r}_i| > R_\mathrm{s}$, where R_s is some chosen cutoff range. In the GADGET-2 code, a value of $R_\mathrm{s} = 4.5r_\mathrm{s}$ is chosen. If we have as our criterion that the neglected short-range force most not contribute more than 1 % to the correct total force (for that particle), then we must have*

$$R_\mathrm{s} = 4.8r_\mathrm{s}\,,$$

which is the default value for the CONCEPT code. Like r_s, this can easily be changed in the parameter file. We can now write (2.65) as

$$\boldsymbol{F}_{\mathrm{s},i} \approx Gm_i \sum_{\substack{j=1 \\ j\neq i \\ |\boldsymbol{r}_j - \boldsymbol{r}_i| < R_\mathrm{s}}}^{N} m_j \frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^3} \left[\operatorname{erfc}\left(\frac{|\boldsymbol{r}_j - \boldsymbol{r}_i|}{2r_\mathrm{s}} \right) + \frac{|\boldsymbol{r}_j - \boldsymbol{r}_i|}{\sqrt{\pi}\, r_\mathrm{s}} \mathrm{e}^{-\frac{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2}{4r_\mathrm{s}^2}} \right] .$$

*This follows directly from (2.65):

$$0.01 = \operatorname{erfc}\left(\frac{R_\mathrm{s}}{2r_\mathrm{s}} \right) + \frac{R_\mathrm{s}}{\sqrt{\pi}\, r_\mathrm{s}} \mathrm{e}^{-\frac{R_\mathrm{s}^2}{4r_\mathrm{s}^2}} \Rightarrow \frac{R_\mathrm{s}}{r_\mathrm{s}} \approx 4.7634\,.$$

The question of how to actually perform the sum (2.65) without visiting all $N-1$ particles remain. This can only be solved by globally, spatially sorting the particles in advance. This is done in e.g. the treePM method, the default method used by GADGET-2. In the plain P^3M method implemented in the CONCEPT code, no additional sorting takes place. Instead we are saved from the $\mathcal{O}(N^2)$ problem by an implementation aspect which we have not yet discussed — parallelization. Before concluding our development of the P^3M method then, we need to take a detour around parallelism.

2.3.2 Parallelization

To do large simulations, accurate within a range of scales many orders of magnitude apart, we need many particles N. This in turn increases the number of computer operations, which constitutes the run. We need our runs to take a manageable amount of time, and so we have two options: Run the simulation on faster hardware or distribute the work of the run over several machines, which then run in parallel. In practice the latter option is preferable, as it is relatively easy and cheap to combine many ordinary computers into a single unit; a cluster. Most such clusters today are *distributed*, meaning that they consist of many CPUs grouped together into so-called nodes, where each node has its own dedicated memory. A serious N-body code then needs to be written as a parallel, distributed program.

The memory management of the CPUs inside a node should be handed carefully, so that one does not overwrite the data of another. The cleanest way of doing this is to completely separate the memory associated with each CPU, effectively isolating them from each other. From the point of view of the code, it now does not matter whether two CPUs reside inside the same node or not. A copy of the same program, possibly slightly modified, can now be run in parallel. The general N-body problem cannot be solved in this way, as sharing of data between the CPUs is a must. Problems which can be solved in this isolated fashion are referred to as *embarrassingly* parallel, an example or which is the tabulation of the Ewald force corrections (2.22).

For the more general problems requiring exchange of data between CPUs, two options are available. A mechanism can be deployed for safe sharing of memory within a node, of which OpenMP (Open Multi-Processing) is the de facto standard. We are still left with no cooperation between different nodes, and so OpenMP alone is not ideal for massively parallel programs. The other option is to communicate between CPUs via message passing: Rather than sharing memory, copies of data are send and received by the CPUs. The standardized system for this CPU-CPU communication is called MPI (Message Passing Interface), and can be used by CPUs within a singe node as well as CPUs belong to different nodes. For memory-bound problems, it can be advantageous to utilize both OpenMP and MPI, as message passing between CPUs of the same node uses more memory than simply sharing it. This hybrid parallelism

complicates the numerical code significantly, and so the CONCEPT code as well as the GADGET-2 code utilizes MPI only.

So far in our development of the numerical implementations of the gravitational solvers, we have not taken parallelism into account. Any valid program is automatically a valid MPI program, as it can trivially be run in embarrassingly parallel. To hook into the features of MPI, each CPU — in MPI terminology called a process — is given a unique rank between 0 and $N_\text{p} - 1$, where N_p is the number of processes. In the N-body code, each process governs its own set of particles, located within its dedicated volume of the box. At the start of the simulation, the box is then divided up into N_p equal volumes, each designated a specific process/rank. We shall refer to these rank specific volumes as domains. The domains are chosen to be of equal volume in an attempt to fairly* spread out the workload between the processes. The processes then receive the particles accompanying their designated domain, after which the actual simulation can begin. If a particle later exits its domain, the particle is immediately send to the process governing the newly entered domain. In order to keep this communication at a minimum, the domains should have a large volume to surface ratio. Ideally the domains are cubes, but this is only possible for cubic N_p. In the CONCEPT code, the best possible rectangular cuboid for the given N_p is computed† and chosen as the shape of the domains.

Each of our methods for solving gravity now need to be modified in such a way as to exploit message passing to speed up the computation. Below we discuss the needed changes of the established PP, PM and P^{3}M methods, in order to parallelize them to fit into the distributed scheme described above. While the PP and the PM methods do parallelize nicely, parallelization is a key feature of the P^{3}M method, without which the method is no good at all.

PP

The PP method as described in section 2.1 only need slight‡ modifications in order to be fully parallelized. The pairwise sum (e.g. (2.6)) cannot run over all particle pairs directly, as these reside on different processes. Instead, only particles within the same domain may be paired directly in this way. The rest of the pairs must be obtained by communicating particle positions between processes. For the particles i within the domain dom_ℓ of the process of rank ℓ,

*For large enough simulations, the universal homogeneity implies that equal volume leads to equal number of particles.

†This amounts to finding the three-number factorization of N_p with all three numbers as similar as possible.

‡This is from the standpoint of understanding the method. The number of additional lines of code needed to parallelize the PP method is significant.

the parallelized sum becomes

$$\sum_{\substack{j=1\\ j\neq i}}^{N} \to \sum_{\substack{j\neq i\\ \boldsymbol{r}_j\in\mathrm{dom}_\ell}} + \sum_{\substack{\mathrm{rk}=0\\ \mathrm{rk}\neq\ell}}^{N_\mathrm{p}-1} \sum_{\substack{j\\ \boldsymbol{r}_j\in\mathrm{dom}_\mathrm{rk}}},$$

where the rk sum runs over all ranks except ℓ. To minimize the computation, processes receiving particles should send the resulting forces back to the sender process, so that $\boldsymbol{F}_{ij}$ and $\boldsymbol{F}_{ji}$ are obtained in one go. It is then important that the process to send to and the process to receive from are different, so that the same computation does not take place on two processes. This is not completely possible if N_p is odd however, as in the end, each process lacks only to be paired with a single other process.

PM

In the parallel PM method, not just the particles but also the mesh is distributed. As we shall see, this means that a lot more work needs to be done in order to parallelize the PM method.

When doing the mass assignment, it is now possible for particle i located in domain ℓ to contribute to the mesh embedded in domain $\ell+1$. This is really the same problem as originally posed by periodicity, which we solved by introducing pseudopoints. We therefore equip each local mesh with pseudopoints just as we did for the entire (now called global) mesh in section 2.2.2. These pseudopoints now correspond to real points in neighbouring domains. These additional pseudopoints are then treated in an analogous fashion to that of the preexisting pseudopoints, both when it comes to mass assignment, potential differencing and force interpolation. Note that no additional ghost points need to be added.

The Fourier transforms of the PM method presents another problem, as the data we wish to transform is now distributed. It is worth mentioning the amount of care taken to speed up the process of Fourier transforming, as well as fixing the problem of distributed data. We are in need of a numerical implementation of a distributed-memory FFT running on MPI. In addition, we want the FFT to be in-place and in three dimensions. Luckily, numerical libraries with such capabilities do exist. Both CONCEPT and GADGET-2 rely on the* FFTW (the Fastest Fourier Transform in the West, [16]) library to perform these transforms. For reasons of speed, FFTW allows only a single floating point number to be stored at each mesh point. This introduces a problem, since the data to be transformed are real, and so the complex result of the transform consists of double the amount of floating point numbers. However, Fourier transforming real data results in negative-frequency terms that are just the complex conjugates of the corresponding positive-frequency terms. We can

*To be precise, CONCEPT rely on FFTW 3, with which GADGET-2 is incompatible. Instead, it uses FFTW 2.

exploit this redundancy by using a *real* FFT, also included in the FFTW library, which intelligently skips the calculation of the negative-frequency terms. The transformed data still does not quite fit the mesh, as the negative-frequency parts constitutes less than half of the data (due to the zero frequency parts). The mesh thus have to be padded with additional points, which receive values from the FFT but are now otherwise used. These inevitably coincide with the ghost and pseudopoints, but as these get their values reassigned after the mesh has been transformed back, it does not pose a problem. One last subtlety of efficient and distributed FFTs remain. The last step in the distributed-memory, in-place, real, 3D FFT consists of a global transposition of the data, without which the data comes out transposed. Such a transposition is costly when the data is distributed and so we choose to skip it, which is possible with FFTW. In Fourier space, the mesh is then transposed (along the first two dimensions). Once we transform back and again skip the transposition within the FFT, the mesh will no longer be transposed.

The above discussion about the numerical implementation of the FFT was really only concerned with the details. The major difference between a serial FFT and a distributed-memory FFT using FFTW, is that the data must be supplied in a given way. Specifically, it is expected that the mesh is "slab decomposed", meaning that the local mesh on each process is a yz-slab with little thickness in the x-direction. The slab decomposition of the global $N_\mathrm{m} \times N_\mathrm{m} \times N_\mathrm{m}$ mesh is then the N_p meshes $N_\mathrm{m}/N_\mathrm{p} \times N_\mathrm{m} \times N_\mathrm{m}$. With the domains chosen as described above, the local meshes are not embedded within the domains, making mass assignment impossible. To circumvent this problem, yet another mesh is introduced, which *is* embedded within the local domain. That is, both meshes are embedded within the box in a global sense, but they are distributed differently among the processes. To distinguish between the two meshes, we shall refer to the slab distributed mesh as the slab mesh, and the newly introduce mesh as the domain mesh. The domain mesh then takes the role of the mesh as described in section 2.2, with regards to mass assignment, potential differencing and force interpolation. After assigning masses to the domain mesh, we then communicate the values to the slab mesh, which then gets Fourier transformed. When all operations in (transposed) Fourier space has been done and the slab mesh has been transformed back to real space, its potential values are then communicated back to the domain mesh. From there, we can forget that we ever had another mesh and use the domain mesh to compute the forces.

$\mathbf{P^3M}$

The parallelization of the long-range force computation in the $\mathrm{P^3M}$ method is exactly as that of the the force in the PM method, described above. We have not yet discussed how to handle the sum in the short-range force (2.65), serially or in parallel. The following discussion of this matter is based on the relatively

simple P^{3}M implementation used within CONCEPT. More sophisticated parallel implementations of the P^{3}M method are possible.

As stated earlier, some pre-existing spatial sorting of the particles are needed, so that we know for which particles the condition $|\boldsymbol{r}_j - \boldsymbol{r}_i| < R_\mathrm{s}$ is true. The division of particles into domains, enforced by the parallelization, is exactly such a sorting. Within a domain, the particles remain unsorted relative to each other, and so particle i must still be paired with all particles $j \neq i$ within the domain containing particle i. If R_s is chosen smaller than the (smallest) linear size of a domain, we are guaranteed that the only other particles that can contribute to the sum (2.65) are those in the 26 neighbouring domains. If the simulation is run on hundreds or thousands of processes, we see why this now gives a tremendous speedup compared to the pairwise sum of the PP method.

The communication of the particles between the domains happens as follows. All particles within a distance of R_s to the right boundary of their local domain are identified, as well as those within a distance of R_s to the left boundary. The particles near the right boundary are now send to the process governing the domain to the right. This means that every process now receives particles from the domain to their left. The received particles are precisely those which should be paired with the previously identified particles near the left domain boundary. The forces between these two groups of particles are computed in accordance with (2.65), after which the local forces are applied to the local particles and the external forces are send back to the process governing the domain to the left. The forces are received from the right and applied to the local particles near the right domain boundary. All particles have now received short-range force contributions from the domain to their right and to their left. The same is now done for the remaining 12 directions.

We stated earlier that $R_\mathrm{s} < \min_{w\in\{x,y,z\}} L_{\mathrm{dom},w}$ should hold true for the P^{3}M method to function, where $L_{\mathrm{dom},w}$ is the linear size of a domain in the w direction. In fact the condition is slightly more involved:

$$R_\mathrm{s} < \begin{cases} \dfrac{1}{2} \min\limits_{w\in\{x,y,z\}} L_{\mathrm{dom},w} & \text{if } \max\limits_{w\in\{x,y,z\}} L_{\mathrm{dom},w} \geq \dfrac{L}{2}\,, \\ \min\limits_{w\in\{x,y,z\}} L_{\mathrm{dom},w} & \text{otherwize.} \end{cases} \tag{2.66}$$

The top condition ensures that if the left and the right domain are really one and the same, no specific particle-particle interaction happens twice. As stated previously, the bottom condition ensures that all particles within a distance R_s is within the current domain or the surrounding 26 domains. If the condition (2.66) is now satisfied, CONCEPT will fail with an error message.

With the P^{3}M fully constructed, we wish to test it against the PP and the PM method. Figure 2.8 shows the power spectra of simulations run with all three methods. It is clear that the P^{3}M method has produced results almost exactly equal to those of the PP method at all scales (probed by the power

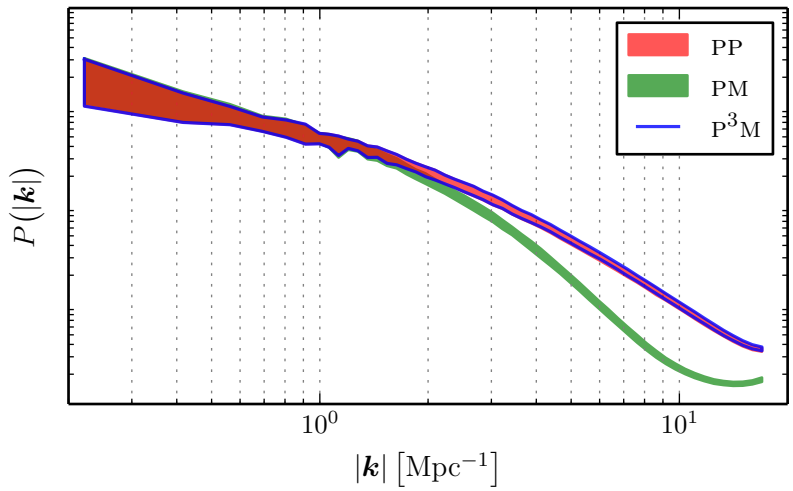


Figure 2.8 – Power spectra of PP (red), PM (green) and P^{3}M (blue) simulations with identical initial conditions. The specifics of the simulations are equal to those of Figure 2.5. The PP and the PM power spectrum are identical to those of Figure 2.7. Because the P^{3}M power spectrum is nearly identical to the PP power spectrum, only its periphery is shown.

spectrum), which was what we were striving for. We now have a parallel gravitational solver which is both accurate and efficient.

2.3.3 Modern Methods

We are done describing the gravitational capabilities of the CONCEPT code. The most sophisticated cosmological N-body codes goes further, utilizing yet more sophisticated methods to speed up the force computation. No increase in accuracy is obtained, however. We shall here briefly describe the idea behind the most common of these modern methods.

The Tree Method

Accuracies comparable to those obtained by direct particle-particle methods can be achieved by tree methods, which have the same scaling* [1] $\mathcal{O}\big(N \log N\big)$ as PM methods. In tree methods, the entire simulation box is cubically subdivided. That is, each mother cube (starting from the entire box) is divided into eight daughter cubes, which again is divided into eight granddaughter cubes, and so on, arranging the entire box into an octree. Each branch of this tree stores a low-order multipole expansion of the enclosed mass distribution, allowing the branch to approximately represent the configuration of all interior particles. In the case of tree method implementation used in GADGET-2, a simple monopole expansion is used. That is, the total enclosed mass as well as the position

*Altohugh the scaling is the same, PM methods are generally the fastest of the two.

of the center of mass is stored within each branch, representing the particle distribution within it as a single, collective particle. To calculate the force $\boldsymbol{F}_i$, the tree is simply "walked" from the trunk down, with each successive subbranch "opened" if a more detailed description of the corresponding interior particles is required for calculating $\boldsymbol{F}_i$ to the desired precision. When a branch with a good enough description of its interior mass distribution is reached, its multipole expansion is used to represent all particles at a deeper level of the tree, lowering the number of effective particles in the familiar pairwise sum substantially.

The criterion determining whether a branch should be used or opened can be chosen in many different ways. The standard criterion used in the GADGET-2 code is the following. When walking the tree during the computation of $\boldsymbol{F}_i$, a branch is used if

$$\frac{GM}{r^2}\left(\frac{L_{\text{branch}}}{r}\right)^2 \leq \alpha|\boldsymbol{a}_i|\,, \tag{2.67}$$

where L_{branch} is the linear size of the branch, $\boldsymbol{a}_i$ is the current acceleration of particle i and α is some dimensionless parameter controlling the accuracy of the force computation.

The TreePM Method

The standard gravitational method used in the GADGET-2 code is the hybrid treePM method. It is essentially the P^{3}M method with the particle-particle forces supplied by the tree method. The tree method works because the force on a particle due to a distant group of particles is not very sensitive to the exact configuration of these particles, which can then be accurately represented as a low-order multipole expansion. In the treePM method, where only the short-range force (2.65) is supplied by the tree walk, $\boldsymbol{F}_i$ have an even weaker dependence on the exact configuration of the distant particle group. This means for a any desired accuracy, the treePM method may use a larger value of α in the opening criterion (2.67), than what may be used in the bare tree method.

Spatial Adaptivity

Some modern N-body codes uses adaptive gravitational methods. We can imagine a particular dense region of space, where a better spatial resolution than elsewhere is requires to properly resolve the structure formation. Such a local increase in resolution can be achieved by an additional, finer mesh embedded within this region. The GADGET-2 code contains this optional feature, there referred to as *zoom*.

We can take this idea of submeshes further, and imagine subregions within the dense region where even higher resolutions are needed, calling for yet finer submeshes. Such hierarchical submeshes was first introduced by [17].

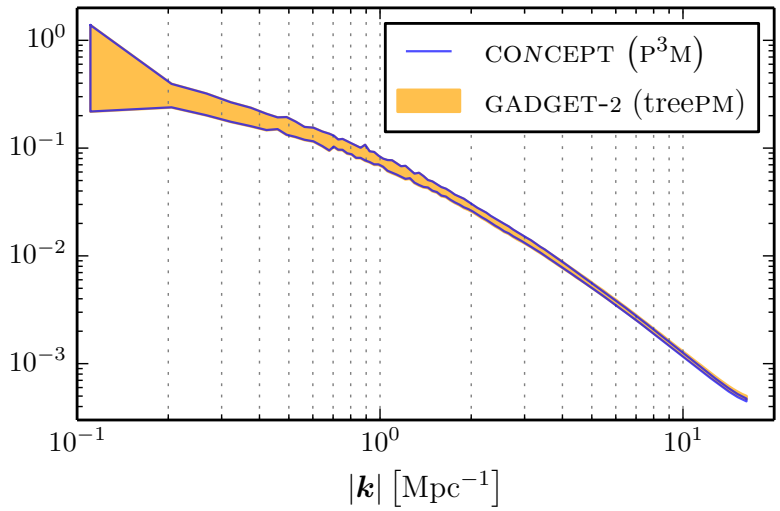


Figure 2.9 – Power spectra at $a = 1$ for a CONCEPT (blue) and a GADGET-2 (orange) run of identical initial conditions. The gravitational methods used by the two N-body codes are the P^{3}M and the treePM method, respectively. For the simulations, $N = 128^3$, $L = 128\,\mathrm{Mpc}/h$ has been used, together with a softening length of $\epsilon = 30\,\mathrm{kpc}/h$. The cosmology is that of the previous simulations considered, $\Omega_\Lambda = 0.7$, $\Omega_\mathrm{m} = 0.3$, $H_0 = 70\,\mathrm{km\ s^{-1}\ Mpc^{-1}}$. The mesh used in the simulations as well as in the production of the power spectra themselves, have a size of $N_\mathrm{m} = 1024$. Since both the box and the mesh have been increases by the same linear factor relative to the simulations of Figure 2.8, the smallest resolved scale shown here is the same as that shown in Figure 2.8. The power spectra shown here do extend to larger scales though.

Later, methods for automatic, adaptive insertions of these submeshes became available, as first introduced by [18]. Such methods are generally referred to as AMR (Adaptive Mesh Refinement) methods. Naturally, these can be combined with PP or tree methods.

With the major trends in modern gravitational methods laid out, it is time to compare the results of CONCEPT against those of a well established N-body code. Figure 2.9 shows powerspectra of the same simulation run with the CONCEPT (using the P^{3}M method) and GADGET-2 (using the treePM method). The powerspectra match almost perfectly, though at the smallest resolved scales, GADGET-2 produces slightly more structure then CONCEPT. This is most likely because of the adaptive timestep size implemented in GADGET-2, the functionality of which are lacking in the CONCEPT code. This then concludes our quest for gravity and point us in the direction of time evolution, the subject of the next chapter.

3 Collisionless Dynamics

In the previous chapter we established methods for computing the gravitational force $\boldsymbol{F}_i$ on each particle i due to all others. In all cases, the forces were softened, resulting in collisionless interactions. We now need to establish a scheme for applying these forces to the particles, evolving the system through time. In the derivation of the forces, Euclidean space was assumed. We therefore have to modify the forces from the previous chapter to account for the Hubble expansion. The objective of this chapter is then twofold: Derive the Newtonian equations of motions in comoving coordinates, including the transformation between Euclidean and comoving forces. Next, construct a numerical scheme for integrating the derived equations of motions through time. Such a time integration scheme together with a gravitational solver from the previous chapter, collectively constitutes a complete N-body code.

3.1 Equations of Motion in Comoving Coordinates

In this section we develop the equations of motion for the system of N particles, using comoving coordinates.

3.1.1 The Single-Particle Hamiltonian

Let us develop the equations of motion for a particle in comoving coordinates, using the Hamiltonian formalism. To do this we imagine an observer comoving with the Hubble flow. The proper coordinates of particle i, as seen from the observer, is denoted $\boldsymbol{r}_i$. The proper velocity of the particle is simply $\dot{\boldsymbol{r}}_i$, where $\cdot = \mathrm{d}/\mathrm{d}t$ and t is the proper time measured by the observer (the cosmic time). The Lagrangian for this single particle is then given by

$$\mathcal{L}_i(\boldsymbol{r}_i, \dot{\boldsymbol{r}}_i, t) = \frac{1}{2} m_i \dot{\boldsymbol{r}}_i^2 - m_i \phi(\boldsymbol{r}_i)\,, \tag{3.1}$$

where m_i is the mass of the particle and ϕ is the proper potential. The explicit time dependence of the Lagrangian comes from the Hubble flow, which is implicitly embedded within $\boldsymbol{r}_i$. The comoving coordinates $\boldsymbol{x}_i$ are given by

$$\boldsymbol{r}_i = a\boldsymbol{x}_i\,. \tag{3.2}$$

The time derivative is then

$$\begin{aligned}\dot{\boldsymbol{r}}_i &= \dot{a}\boldsymbol{x}_i + a\dot{\boldsymbol{x}}_i \\ &= H\boldsymbol{r}_i + \boldsymbol{u}_i\,,\end{aligned} \tag{3.3}$$

where $\boldsymbol{u}_i$ is the peculiar velocity of particle i. Any changes to $\boldsymbol{u}_i$ are purely the result of forces acting upon the particle. Choosing $\boldsymbol{u}_i$ rather than* $\dot{\boldsymbol{r}}_i$ as the velocity variable thus allow us to forget the expansion and concentrate on the dynamics.

The first step in deriving the equations of motion in comoving coordinates is to write the Lagrangian (3.1) in these coordinates. Inserting (3.2) and (3.3) into (3.1), we get

$$\mathcal{L}_i(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i, t) = \frac{1}{2}m_i(\dot{a}\boldsymbol{x}_i + a\dot{\boldsymbol{x}}_i)^2 - m_i\phi(a\boldsymbol{x}_i)\,, \tag{3.4}$$

where now a is responsible for the time-dependence of the Lagrangian. One important artefact of the introduction of comoving coordinates into the Lagrangian, is that it is no longer separable into kinetic† and potential terms. That is, there exists a term in (3.4) containing both $\boldsymbol{x}_i$ and $\dot{\boldsymbol{x}}_i$. This complicates further analysis, and so we wish to bring the Lagrangian back to the original, separable form. We do this by applying a gauge transformation of the form

$$\mathcal{L}_i(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) \to \mathcal{L}_i(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) - \frac{\mathrm{d}}{\mathrm{d}t}\frac{1}{2}m_i a\dot{a}\boldsymbol{x}_i^2\,.$$

Applying this transformation, we get

$$\begin{aligned}\mathcal{L}_i(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i, t) \to &+ \frac{1}{2}m_i\big(\dot{a}^2\boldsymbol{x}_i^2 + a^2\dot{\boldsymbol{x}}_i^2 + 2a\dot{a}\boldsymbol{x}_i\dot{\boldsymbol{x}}_i\big) - m_i\phi(a\boldsymbol{x}_i) \\ &- \frac{1}{2}m_i\big(\dot{a}^2\boldsymbol{x}_i^2 + a\ddot{a}\boldsymbol{x}_i^2 + 2a\dot{a}\boldsymbol{x}_i\dot{\boldsymbol{x}}_i\big) \\ &= \frac{1}{2}m_i a^2\dot{\boldsymbol{x}}_i^2 - m_i\phi(a\boldsymbol{x}_i) - \frac{1}{2}m_i a\ddot{a}\boldsymbol{x}_i^2 \\ &= \frac{1}{2}m_i a^2\dot{\boldsymbol{x}}_i^2 - \frac{m_i\varphi(\boldsymbol{x}_i)}{a}\,, \quad \varphi(\boldsymbol{x}_i) \equiv a\phi(a\boldsymbol{x}_i) + \frac{1}{2}a^2\ddot{a}\boldsymbol{x}_i^2\,,\end{aligned} \tag{3.5}$$

where φ is the peculiar potential, the same as in (1.31). Notice that (3.5) is indeed separated into a kinetic and a potential term.

*Actually we can choose any velocity variable which is not explicitly spatially dependent and is proportional to $\dot{\boldsymbol{r}}_i$. In the end we shall use the comoving canonical momentum, conjugate to $\boldsymbol{x}_i$.

†Here, "kinetic" and "potential" should be understood in the generalized sense. It is not important that the Lagrangian can be expressed as the difference between the actual kinetic and potential energy, but rather that there exist *some* quantities in which the Lagrangian is separable.

With a separable Lagrangian in comoving coordinates, we now define the comoving canonical momentum $\boldsymbol{p}_i$ conjugate to $\boldsymbol{x}_i$:

$$\begin{aligned}\boldsymbol{p}_i &\equiv \frac{\partial \mathcal{L}_i}{\partial \dot{\boldsymbol{x}}_i} \\ &= m_i a^2 \dot{\boldsymbol{x}}_i \,. \end{aligned} \tag{3.6}$$

Using (3.6), the Lagrangian (3.5) can then be expressed as

$$\mathcal{L}_i(\boldsymbol{x}_i, \boldsymbol{p}_i, t) = \frac{\boldsymbol{p}_i^2}{2m_i a^2} - \frac{m_i \varphi(\boldsymbol{x}_i)}{a} \,.$$

The Hamiltonian is then

$$\begin{aligned}\mathcal{H}_i(\boldsymbol{x}_i, \boldsymbol{p}_i, t) &\equiv \dot{\boldsymbol{x}}_i \boldsymbol{p}_i - \mathcal{L}_i(\boldsymbol{x}_i, \boldsymbol{p}_i, t) \\ &= \frac{\boldsymbol{p}_i^2}{m_i a^2} - \mathcal{L}_i(\boldsymbol{x}_i, \boldsymbol{p}_i, t) \\ &= \underbrace{\frac{\boldsymbol{p}_i^2}{2m_i a^2}}_{\mathcal{H}_{\boldsymbol{p}_i}} + \underbrace{\frac{m_i \varphi(\boldsymbol{x}_i)}{a}}_{\mathcal{H}_{\boldsymbol{x}_i}} \,, \end{aligned} \tag{3.7}$$

which again is separated into a kinetic and a potential term, here denoted $\mathcal{H}_{\boldsymbol{p}_i}$ and $\mathcal{H}_{\boldsymbol{x}_i}$, respectively. Hamilton's equations now give us the equations of motion, expressed purely in comoving quantities:

$$\begin{aligned}\dot{\boldsymbol{x}}_i &= \frac{\partial \mathcal{H}_i(\boldsymbol{x}_i, \boldsymbol{p}_i, t)}{\partial \boldsymbol{p}_i} & \dot{\boldsymbol{p}}_i &= -\frac{\partial \mathcal{H}_i(\boldsymbol{x}_i, \boldsymbol{p}_i, t)}{\partial \boldsymbol{x}_i} \\ &= \frac{\boldsymbol{p}_i}{m_i a^2} \,. & &= \frac{\boldsymbol{f}_i}{a} \,, \end{aligned} \tag{3.8}$$

where we have written the change in momentum in terms of the comoving force

$$\boldsymbol{f}_i \equiv -m_i \nabla_{\boldsymbol{x}} \varphi(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{x}_i} \,, \tag{3.9}$$

where a subscript $\boldsymbol{x}$ designates differentiation with respect to comoving coordinates.

The Hamiltonian equations (3.8) with the comoving force defined by (3.9) constitute the sought equations of motion. In the CONCEPT code, a given particle i at some time is completely defined by $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$. The particle masses m_i are all equal*, and so do not form part the data defining the individual particles. In other N-body codes, additional information is kept around, such as the acceleration (needed explicitly in e.g. the opening criterion of the tree method (2.67)) and a unique identifier corresponding to the label i. Because the particle data is communicated between the processes during the simulation, the memory addresses of the particle data cannot be used to label the particles, and so an explicit identifier is needed if we wish to follow the trajectory of a specific particle.

*It is however possible to create separate groups of particles, each group with its own particle mass.

3.1.2 The Comoving Force

The rate of change of the comoving momentum is proportional to the so-called comoving force $\boldsymbol{f}_i$, as defined in (3.8). The comoving force can be written in terms of the peculiar potential, as in (3.9), where the peculiar potential itself is given in (3.5). The relation between this comoving force $\boldsymbol{f}_i$ and the (Euclidean) force $\boldsymbol{F}_i$ from the previous chapter is not at all clear. Replacing φ by its definition (3.5), the comoving force becomes

$$\begin{aligned}\boldsymbol{f}_i &= -m_i\big(a^2\ddot{a}\boldsymbol{x}_i + a\,\nabla_{\boldsymbol{x}_i}\phi(a\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_i}\big)\\ &= -m_i\big(a^2\ddot{a}\boldsymbol{x}_i + a^2\,\nabla_{\boldsymbol{r}}\phi(\boldsymbol{r})|_{\boldsymbol{r}=\boldsymbol{r}_i}\big)\,,\end{aligned} \tag{3.10}$$

where $\boldsymbol{r} = a\boldsymbol{x} \Rightarrow \nabla_{\boldsymbol{x}} = a\nabla_{\boldsymbol{r}}$ has been used. Given that $\dot{\boldsymbol{p}}_i \propto \boldsymbol{f}_i$ (3.8), the term proportional to $\boldsymbol{x}_i$ in (3.10) is worrying. The gravitational force ought not to depend on our choice of origin! It turns out that this apparent problem is actually the same as the one we encountered when doing the infinite sum over particle images, eventually leading us to the Ewald summation method. As discussed in chapter 2, the sum over particle images (2.12) is only conditionally convergent. To ensure absolute convergence, one must specify the order of summation. The correct order of summation, perhaps unsurprisingly, turned out to be radially outwards, with the position of the particle in question as center. In a analogous manner, the comoving force as stated in (3.10) should always be calculated with the origin placed at $\boldsymbol{x}_i$. This may seem like a bit of a cheat, but it is the only way of making sense of (3.10). In the following, we remove the term in a more rigorous way, although it really do just amount to a change in origin, done separately for each particle.

We know how to write the proper gradient of the proper potential, appearing in (3.10), as a sum over all particles but i. To emulate an infinite universe, a sum over particle *images* (2.12) should be used:

$$\boldsymbol{f}_i = m_i\left(-a^2\ddot{a}\boldsymbol{x}_i + G\sum_{\substack{j=1\\ j\neq i}}^{N} m_j \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \frac{\boldsymbol{x}_j + \boldsymbol{n}L_a - \boldsymbol{x}_i}{|\boldsymbol{x}_j + \boldsymbol{n}L_a - \boldsymbol{x}_i|^3}\right),$$

where the proper coordinates have been converted back to comoving coordinates and $L_a = L/a$ is the comoving box size. We now move to continuous space by writing the sums as an integral over delta functions,

$$\boldsymbol{f}_i = m_i\left(-a^2\ddot{a}\boldsymbol{x}_i + G\int \frac{\boldsymbol{x} - \boldsymbol{x}_i}{|\boldsymbol{x} - \boldsymbol{x}_i|^3} \sum_{\substack{j=1\\ j\neq i}}^{N} m_j \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \delta(\boldsymbol{x}_j + \boldsymbol{n}L_a - \boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\right).$$

It is easy to see that if we were to erase the prefactor $(\boldsymbol{x} - \boldsymbol{x}_i)/|\boldsymbol{x} - \boldsymbol{x}_i|^3$ from the integral, it would diverge due to the sums counting up all the mass in the Universe. The prefactor regulates the summation, making the integral conditionally convergent — If we happen to sum in the correct order, the

integral will be finite. We may subtract the mean, comoving density $\bar{\varrho}$ from the sums, in order to ensure absolute convergence.

$$\boldsymbol{f}_i = m_i \Bigg(-a^2 \ddot{a} \boldsymbol{x}_i + G\bar{\varrho} \int \frac{\boldsymbol{x} - \boldsymbol{x}_i}{|\boldsymbol{x} - \boldsymbol{x}_i|^3} \, \mathrm{d}\boldsymbol{x} \\ + G \int \frac{\boldsymbol{x} - \boldsymbol{x}_i}{|\boldsymbol{x} - \boldsymbol{x}_i|^3} \left[-\bar{\varrho} + \sum_{\substack{j=1 \\ j \neq i}}^{N} m_j \sum_{\boldsymbol{n} \in \mathbb{Z}^3} \delta(\boldsymbol{x}_j + \boldsymbol{n} L_a - \boldsymbol{x}) \right] \mathrm{d}\boldsymbol{x} \Bigg) , \tag{3.11}$$

where the additional integral is introduced to cancel any effect of the inserted mean density. We now wish to show the equality between the two upper terms of (3.11),

$$a^2 \ddot{a} \boldsymbol{x}_i = G\bar{\varrho} \int \frac{\boldsymbol{x} + \boldsymbol{x}_i}{|\boldsymbol{x} - \boldsymbol{x}_i|^3} \, \mathrm{d}\boldsymbol{x} \, . \tag{3.12}$$

Using the second Friedmann equation (1.13), the first term becomes

$$\begin{aligned} a^2 \ddot{a} \boldsymbol{x}_i &= -\frac{4\pi G}{3} a^3 \bar{\rho} \boldsymbol{x}_i \\ &= -\frac{4\pi G}{3} \bar{\varrho} \boldsymbol{x}_i \, . \end{aligned} \tag{3.13}$$

We can prove the equality between the vectors (3.12) by showing that they have the same divergence and curl. With the new left-hand-side (3.13) for equation (3.12), taking the divergence of both sides yields

$$\begin{aligned} a^2 \ddot{a} \nabla_{\boldsymbol{x}_i} \cdot \boldsymbol{x}_i &= -\frac{4\pi G}{3} \bar{\varrho} \nabla_{\boldsymbol{x}_i} \cdot \boldsymbol{x}_i \\ &= -4\pi G \bar{\varrho} \, , \\ G\bar{\varrho} \int \nabla_{\boldsymbol{x}_i} \cdot \frac{\boldsymbol{x} - \boldsymbol{x}_i}{|\boldsymbol{x} - \boldsymbol{x}_i|^3} \, \mathrm{d}\boldsymbol{x} &= -4\pi G \bar{\varrho} \int \delta(\boldsymbol{x} - \boldsymbol{x}_i) \, \mathrm{d}\boldsymbol{x} \\ &= -4\pi G \bar{\varrho} \, . \end{aligned}$$

Now in both cases the curl vanishes, which it of course must do due to gravity being conservative. We have then proven (3.12). We can thus erase these terms in the comoving force (3.11), ridding us from the unwanted term proportional to $\boldsymbol{x}_i$:

$$\boldsymbol{f}_i = G m_i \int \frac{\boldsymbol{x} - \boldsymbol{x}_i}{|\boldsymbol{x} - \boldsymbol{x}_i|^3} \left[-\bar{\varrho} + \sum_{\substack{j=1 \\ j \neq i}}^{N} m_j \sum_{\boldsymbol{n} \in \mathbb{Z}^3} \delta(\boldsymbol{x}_j + \boldsymbol{n} L_a - \boldsymbol{x}) \right] \mathrm{d}\boldsymbol{x} \, . \tag{3.14}$$

The subtraction of $a^2 \ddot{a} \boldsymbol{x}_i$ in the comoving potential should then really be understood as a subtraction of the mean density, just as we did for the periodic

potential in chapter 2. This remarkable result is known as "Jeans' swindle", although it is fully rigorous [19].

As also discussed in chapter 2, subtracting the mean density of the Universe cannot perturb the gravitational force. We ought then to be able to remove the integral over the mean density from (3.14) in its entirety. As we can freely shift $\boldsymbol{x}$ when doing this infinite integral, doing $\boldsymbol{x} \to \boldsymbol{x} + \boldsymbol{x}_i$ turns the integrand odd. Once again then, the choice of $\boldsymbol{x}_i$ as the origin has proven to be the key when it comes to removal of unwanted terms. With $\bar{\varrho}$ removed from (3.14), we delete the integral and simultaneously remove the delta function:

$$\boldsymbol{f}_i = Gm_i \sum_{\substack{j=1 \\ j\neq i}}^{N} m_j \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \frac{\boldsymbol{x}_j + \boldsymbol{n}L_a - \boldsymbol{x}_i}{|\boldsymbol{x}_j + \boldsymbol{n}L_a - \boldsymbol{x}_i|^3} \,.$$

Finally we recognize the comoving force $\boldsymbol{f}_i$ for what it is; simply the direct analogue of the proper force: $\boldsymbol{f}_i = a^2\boldsymbol{F}_i$.

3.2 Time Integration

Given some initial state of the particle system, the solution to the $6N$ equations of motion (3.8) is a uniquely defined phase space trajectory. The problem of solving these equations are referred to as the N-body problem, which generally have no analytical solution. To find approximate solutions numerically, time is discretized into a finite number of steps, converting the differential equations of motion into difference equations. Time then evolves by iterating through the time steps, with errors being introduced at each iteration. This implies that the computed solution is not unique, but depend on the errors introduced. We can of course control the size of these errors at will, shrinking them below any desired maximum value by increasing the number of time steps and/or using a higher order integration scheme. Increasing the number of steps amount to lowering their separation Δt, called the time step size.

In the following subsections, time evolution for our Hamiltonian N-body system is formally derived and then numerically implemented.

3.2.1 The Drift and Kick Operators

Even though the N-body problem can not be generally solved analytically, we can still write down the formal solution by introducing a time evolution operator. This will turn out to be advantageous when constructing numerical time integration schemes.

In the limit of infinite N (keeping the density finite), the potential φ becomes independent of any one particular particle, effectively decoupling φ from the particle positions and giving it a life of its own. In this limit, the particles are thus independent of each other. This means that the total

Hamiltonian $\mathcal{H}$ of the system is just the sum of single-particle Hamiltonians. We adopt this approximation, known as the mean field approximation, for our system of finite but large N:

$$\mathcal{H}(\vec{\boldsymbol{x}}, \vec{\boldsymbol{p}}, t) = \sum_{i=1}^{N} \mathcal{H}_i(\boldsymbol{x}_i, \boldsymbol{p}_i, t)\,, \tag{3.15}$$

where $\vec{\boldsymbol{x}} \equiv (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, $\vec{\boldsymbol{p}} \equiv (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N)$. We now have a single Hamiltonian from which the dynamics of all particles can be derived. We can even express the evolution of the entire system through a single time evolution operator. To do this we first need the Hamiltonian *operator* for the system. Writing Hamilton's equations of motion using Poisson brackets $\{\bullet, \bullet\}$ allow us to define this operator in a completely symmetric way with respect to position and momentum:

$$\begin{aligned} \dot{\boldsymbol{x}}_i &= \big\{\boldsymbol{x}_i, \mathcal{H}(\vec{\boldsymbol{x}}, \vec{\boldsymbol{p}}, t)\big\} + \underbrace{\frac{\partial \boldsymbol{x}_i}{\partial t}}_{0} & \dot{\boldsymbol{p}}_i &= \big\{\boldsymbol{p}_i, \mathcal{H}(\vec{\boldsymbol{x}}, \vec{\boldsymbol{p}}, t)\big\} + \underbrace{\frac{\partial \boldsymbol{p}_i}{\partial t}}_{0} \\ &\equiv \hat{H}\boldsymbol{x}_i\,, & &\equiv \hat{H}\boldsymbol{p}_i\,, \end{aligned} \tag{3.16}$$

where $\hat{H} = \{\bullet, \mathcal{H}\}$ is the Hamiltonian operator, generating time evolution for the entire system. The partial derivatives vanish because $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$ do not have any explicit time dependence — they do of course evolve in time, but this temporal evolution is encoded entirely in the Hamiltonian and is not an explicit part of their definition. The solution to (3.16) is simply

$$\begin{aligned} \boldsymbol{x}_i(t) &= \exp\!\left(\int_0^t \hat{H}\big(t'\big)\,\mathrm{d}t'\right)\boldsymbol{x}_i(0) & \boldsymbol{p}_i(t) &= \exp\!\left(\int_0^t \hat{H}\big(t'\big)\,\mathrm{d}t'\right)\boldsymbol{p}_i(0) \\ &\equiv \hat{U}(t)\boldsymbol{x}_i(0)\,, & &\equiv \hat{U}(t)\boldsymbol{p}_i(0)\,, \end{aligned} \tag{3.17}$$

where $\hat{U}$ is the sought time evolution operator. These equations are coupled through the $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$ dependence of $\hat{H}$ and should therefore be solved simultaneously. We can express the action of $\hat{U}$ on the entire system at once as

$$\hat{U}(t)\,|\vec{\boldsymbol{x}}(0),\, \vec{\boldsymbol{p}}(0)\rangle = \big|\vec{\boldsymbol{x}}(t),\, \vec{\boldsymbol{p}}(t)\big\rangle\,, \tag{3.18}$$

where we have allowed ourselves to use $|\vec{\boldsymbol{x}}(t), \vec{\boldsymbol{p}}(t)\rangle$ to denote the (classical) state of the system at time t. Since $\hat{U}$ (and $\hat{H}$) is defined to act on canonical variables, in order to be precise we should augment (3.18) with a rule for how these canonical operators act on state kets;

$$\begin{aligned} \hat{U}(t)\,|\vec{\boldsymbol{x}}(0),\, \vec{\boldsymbol{p}}(0)\rangle &= \big|\hat{U}(t)\vec{\boldsymbol{x}}(0),\, U(t)\vec{\boldsymbol{p}}(0)\big\rangle \\ &= \big|\hat{U}(t)\boldsymbol{x}_1(0), \ldots, \hat{U}(t)\boldsymbol{x}_N(0),\, \hat{U}(t)\boldsymbol{p}_1(0), \ldots, \hat{U}(t)\boldsymbol{p}_N(0)\big\rangle \\ &= \big|\hat{U}(t)\boldsymbol{x}_1(0)\big\rangle \cdots \big|\hat{U}(t)\boldsymbol{x}_N(0)\big\rangle\big|\hat{U}(t)\boldsymbol{p}_1(0)\big\rangle \cdots \big|\hat{U}(t)\boldsymbol{p}_N(0)\big\rangle \\ &= \bigotimes_{i=1}^{N} \big|\hat{U}(t)\boldsymbol{x}_i,\, \hat{U}(t)\boldsymbol{p}_i\big\rangle \end{aligned}$$

(and similar for $\hat{H}$). In the last equality the large $\otimes$ operator is used to compound single-particle states into the (many-particle) state of the entire system.

The bare formal definition of the time evolution operator (3.17) is of little use when it comes to actually applying it numerically. For this we need to know explicitly how $\hat{U}$ acts on the canonical variables. To make progress we remember that the single-particle Hamiltonian is separated into kinetic and potential terms. The total kinetic and potential Hamiltonians can then be written analogous to (3.15):

$$\begin{aligned}\mathcal{H}_{\vec{\boldsymbol{p}}}(\vec{\boldsymbol{p}}, t) &= \sum_{i=1}^{N} \mathcal{H}_{\boldsymbol{p}_i}(\boldsymbol{p}_i, t)\,, \\ \mathcal{H}_{\vec{\boldsymbol{x}}}(\vec{\boldsymbol{x}}, t) &= \sum_{i=1}^{N} \mathcal{H}_{\boldsymbol{x}_i}(\boldsymbol{x}_i, t)\,,\end{aligned}$$

where the single-particle kinetic and potential Hamiltonians are given in (3.7). The separation of the Hamiltonian means that the time evolution of the positions depend only on $\mathcal{H}_{\vec{\boldsymbol{p}}}$ while the time evolution of the momenta depend only on $\mathcal{H}_{\vec{\boldsymbol{x}}}$. We can therefore define kinetic $\hat{T} = \{\bullet, \mathcal{H}_{\vec{\boldsymbol{p}}}\}$ and potential $\hat{V} = \{\bullet, \mathcal{H}_{\vec{\boldsymbol{x}}}\}$ Hamiltonian operators, where $\dot{\vec{\boldsymbol{x}}} = \hat{T}\vec{\boldsymbol{x}}$ and $\dot{\vec{\boldsymbol{p}}} = \hat{V}\vec{\boldsymbol{p}}$. The linearity of the Poisson bracket together with $\mathcal{H} = \mathcal{H}_{\vec{\boldsymbol{p}}} + \mathcal{H}_{\vec{\boldsymbol{x}}}$ guarantees that $\hat{H} = \hat{T} + \hat{V}$. Finally we can construct separate time evolution operators for the kinetic and potential parts as

$$\hat{D}(t) = \exp\!\left(\int_0^t \hat{T}\big(t'\big)\,\mathrm{d}t'\right), \qquad \hat{K}(t) = \exp\!\left(\int_0^t \hat{V}\big(t'\big)\,\mathrm{d}t'\right). \tag{3.19}$$

These are known as the drift and kick operator, respectively. The drift operator $\hat{D}(t)$ translates all particle positions t forward in time while leaving the momenta fixed. The kick operator $\hat{K}(t)$ changes all particle momenta by applying the gravitational force over t time, while leaving the positions fixed. That is, they solve the two Hamiltonian equations of motion (3.8) without taking into account the coupling between them. This is of course not quite what we want, but at least we can explicitly write down how these operators act on the system:

$$\begin{aligned}\hat{D}(t)\,|\vec{\boldsymbol{x}}(0),\, \vec{\boldsymbol{p}}(0)\rangle &= \bigotimes_{i=1}^{N} \left|\boldsymbol{x}_i(0) + \frac{\boldsymbol{p}_i(0)}{m_i}\int_0^t \frac{\mathrm{d}t'}{a^2(t')},\, \boldsymbol{p}_i(0)\right\rangle, \\ \hat{K}(t)\,|\vec{\boldsymbol{x}}(0),\, \vec{\boldsymbol{p}}(0)\rangle &= \bigotimes_{i=1}^{N} \left|\boldsymbol{x}_i(0),\, \boldsymbol{p}_i(0) + \boldsymbol{f}_i(0)\int_0^t \frac{\mathrm{d}t'}{a(t')}\right\rangle,\end{aligned} \tag{3.20}$$

where the changes in positions and momenta are read off of (3.8). We remember that the force on particle i only undergo temporal changes through the temporal

changes in the comoving positions, which is why it can be taken outside the integral in the kick operation. The equations (3.20) are closed-form expressions for "drifting" and "kicking" the entire system. That is, for solving either of the two coupled Hamiltonian equations of motion at a time, but not both simultaneously. We now need to establish a connection between the drift and kick operators (3.19) and the time evolution operator (3.17), which solves the *coupled* Hamiltonian equations of motion. We know that the generators of $\hat{U}$, $\hat{D}$ and $\hat{K}$ are additive; $\hat{H} = \hat{T} + \hat{V}$. Therefore $\hat{U}(t) = \exp\big(\int_0^t \big[\hat{T}\big(t'\big) + \hat{V}\big(t'\big)\big]\,\mathrm{d}t'\big)$. This is not simply the product of $\hat{D}$ and $\hat{K}$ because these (and $\hat{T}$ and $\hat{V}$) do not commute. After all, translating positions and then updating the momenta yields different final positions than first updating the momenta and then do the translation. To connect $\hat{U}$ to $\hat{D}$ and $\hat{K}$ one therefore must use the Zassenhaus formula:

$$\begin{aligned}
\hat{U}(t) &= \exp\left(\int_0^t \hat{T}\big(t'\big)\,\mathrm{d}t' + \int_0^t \hat{V}\big(t'\big)\,\mathrm{d}t'\right) \\
&= \exp\left(\int_0^t \hat{T}\big(t'\big)\,\mathrm{d}t'\right) \exp\left(\int_0^t \hat{V}\big(t'\big)\,\mathrm{d}t'\right) \\
&\quad \times \exp\left(-\frac{1}{2}\left[\int_0^t \hat{T}\big(t'\big)\,\mathrm{d}t',\, \int_0^t \hat{V}\big(t'\big)\,\mathrm{d}t'\right]\right) \cdots \\
&= \hat{D}(t)\hat{K}(t) \exp\left(-\frac{1}{2}\left[\int_0^t \hat{T}\big(t'\big)\,\mathrm{d}t',\, \int_0^t \hat{V}\big(t'\big)\,\mathrm{d}t'\right]\right) \cdots ,
\end{aligned} \tag{3.21}$$

where $[\bullet,\, \bullet]$ denotes the commutator and each additional factor in the infinite product is the exponential of some expression involving still more deeply nested commutators of $\int_0^t \hat{T}(t')\,\mathrm{d}t'$ and $\int_0^t \hat{V}(t')\,\mathrm{d}t'$. The $\hat{T}$ and $\hat{V}$ operators appear symmetrically in the definition of $\hat{U}$, but not in the final equalities of (3.21). This means that switching $\hat{T} \leftrightarrow \hat{V}$ and $\hat{D} \leftrightarrow \hat{K}$ in (3.21) produces an equally valid equation. To use (3.21) numerically we of course have to truncate the infinite product somehow. In the next subsection we shall see exactly how this is done.

3.2.2 The Symplectic Leapfrog Integrator

A naïve time integration scheme was included in listing 2.1. In comoving variables, this scheme looks like

$$\begin{aligned}
\boldsymbol{p}_i(t + \Delta t) &= \boldsymbol{p}_i(t) + \boldsymbol{f}_i(t)\Delta t\,, \\
\boldsymbol{x}_i(t + \Delta t) &= \boldsymbol{x}_i(t) + \frac{\boldsymbol{p}_i(t + \Delta t)}{m_i}\Delta t\,,
\end{aligned} \tag{3.22}$$

where Δt is the size of the time step, somehow incorporating the integrals with the scale factor (3.20), embedded within $\hat{D}$ and $\hat{K}$. It is clear that this indeed is a possible numerical implementation of the kick and drift operators defined

in (3.20). However, (3.22) treats positions and momenta asymmetrically, as the current positions are uses to update the momenta (through $\boldsymbol{f}_i(t)$) while the future momenta are used to update the positions. This could of course be chosen the other way around, or even symmetrically*. The consequences of such a choice are not obvious and depend heavily on the nature of the system in question.

Hamiltonian systems are not structurally stable against non-Hamiltonian perturbations, as such perturbations generally do not preserve phase space volume. That is, numerical errors in the temporal evolution have better be Hamiltonian themselves, or the geometry of the solution (the phase space trajectory) can be severely distorted. Trying to use an ordinary numerical integration method (e.g. Runge-Kutta) to integrate a Hamiltonian system will thus break Liouville's theorem of conservation of phase space volume. Ensuring that the errors are themselves Hamiltonian is thus of great importance, even more so than minimizing their sizes, as demonstrated in [1]. Errors which are Hamiltonian in nature can be achieved if each step of the integration is itself a canonical transformation. Such canonical integration schemes are called *symplectic*. The Hamiltonian errors in symplectic integration schemes then ensures conservation of phase space volume, but only in a time-averaged sense; the phase space volume will temporally oscillate around the true value.

We thus wish to construct a symplectic time integrator for our system of particles. We start by writing the drift and kick operators as incremental operators, evolving the system forwards by Δt regardless of the current time t:

$$\begin{aligned}
\hat{D}(\Delta t)\,|\vec{\boldsymbol{x}}(t),\,\vec{\boldsymbol{p}}(t)\rangle &= \bigotimes_{i=1}^{N} \left| \boldsymbol{x}_i(t) + \frac{\boldsymbol{p}_i(t)}{m_i} \int_t^{t+\Delta t} \frac{\mathrm{d}t'}{a^2\big(t'\big)},\, \boldsymbol{p}_i(t) \right\rangle, \\
\hat{K}(\Delta t)\,|\vec{\boldsymbol{x}}(t),\,\vec{\boldsymbol{p}}(t)\rangle &= \bigotimes_{i=1}^{N} \left| \boldsymbol{x}_i(t),\, \boldsymbol{p}_i(t) + \boldsymbol{f}_i(t) \int_t^{t+\Delta t} \frac{\mathrm{d}t'}{a\big(t'\big)} \right\rangle.
\end{aligned} \tag{3.23}$$

The drift and kick operators treat $\vec{\boldsymbol{x}}$ and $\vec{\boldsymbol{p}}$ as temporally discrete variables, while a is treated as being continuous. We could choose to discretize a as well, but this would bring no real benefit as the integrals in (3.23) are relatively fast to compute. In the CONCEPT code, the Runge–Kutta–Fehlberg method is implemented to do these integrals.

The drift and kick operators of (3.23) constitute the building blocks of our symplectic time integrator. As they are build out of the canonical operators $\hat{T}$ and $\hat{V}$ (3.19), they are themselves symplectic. Any composition of drift and kick operators is then also symplectic. We now wish to define a numerical approximation to $\hat{U}$ which is directly computable, as one such composition. The only strict requirement is that the amount of "drifting" should equal the amount of "kicking", so that $\vec{\boldsymbol{x}}$ and $\vec{\boldsymbol{p}}$ are equally evolved. The simplest such choices

*Which would require us to save an intermediate copy of one of the variables.

amounts to the time evolution operators $\hat{D}(\Delta t)\hat{K}(\Delta t)$ and $\hat{K}(\Delta t)\hat{D}(\Delta t)$, the latter being similar to (3.22). Both of these two possible choices for the approximation to $\hat{U}(\Delta t)$ are valid, although they are unequal as $\hat{D}$ and $\hat{K}$ do not commute. From the Zassenhaus expansion (3.21) of the true time evolution operator, we see that the two proposed approximations are only first order accurate in Δt. We would like to opt for a second-order accurate approximation to $\hat{U}$, and so the search continues. One initial thought might be to just calculate the commutator $[T, V]$ to gain the missing accuracy, but it is not clear how one would go about this, as we are without any concrete expressions for $\hat{T}$ and $\hat{V}$.

Instead of resorting to calculating commutators, we can use symmetry arguments to construct a second-order accurate approximation to $\hat{U}$. Let us start by considering the time-reversal of the first-order approximations proposed above:

$$\begin{aligned}\big[\hat{K}(\Delta t)\hat{D}(\Delta t)\big]^{-1} &= \left[\exp\!\left(\int_t^{t+\Delta t}\hat{V}\big(t'\big)\,\mathrm{d}t'\right)\exp\!\left(\int_t^{t+\Delta t}\hat{T}\big(t'\big)\,\mathrm{d}t'\right)\right]^{-1}\\ &= \exp\!\left(-\int_t^{t+\Delta t}\hat{T}\big(t'\big)\,\mathrm{d}t'\right)\exp\!\left(-\int_t^{t+\Delta t}\hat{V}\big(t'\big)\,\mathrm{d}t'\right),\end{aligned}$$

where inversion precisely amounts to time-reversal for any time evolution operator. Time-reversing $\hat{K}\hat{D}$ is then done unsurprisingly by negating $\hat{T}$ and $\hat{V}$, but the more subtle switch $\hat{T} \leftrightarrow \hat{V}$ is also needed. The above is true because from it, it follows that

$$\big[\hat{K}(\Delta t)\hat{D}(\Delta t)\big]\big[\hat{K}(\Delta t)\hat{D}(\Delta t)\big]^{-1} = 1 = \big[\hat{K}(\Delta t)\hat{D}(\Delta t)\big]^{-1}\big[\hat{K}(\Delta t)\hat{D}(\Delta t)\big].$$

We could have done the same for the other composition, $\hat{D}(\Delta t)\hat{K}(\Delta t)$. As the physical drifting and kicking of particles occur simultaneously, switching $\hat{D} \leftrightarrow \hat{K}$ in any expression regarding time evolution produces an equally valid (though different) expression. As not to state everything twice then, we shall resort to only writing one form of such expressions. It is now clear that a more symmetric (with regards to time reversal) time evolution operator would be

$$\hat{U}_{\text{🍁}}(\Delta t) = \hat{K}\left(\frac{\Delta t}{2}\right)\hat{D}(\Delta t)\hat{K}\left(\frac{\Delta t}{2}\right), \tag{3.24}$$

with the nice property $\hat{U}_{\text{🍁}}^{-1}(\Delta t) = \hat{U}_{\text{🍁}}(-\Delta t)$. Since this is the case we can argue that in a Zassenhaus-like expansion of $\hat{U}_{\text{🍁}}(\Delta t)$, all even powers of Δt must vanish. If not, equal terms with the same sign would appear inside the exponentials of $\hat{U}_{\text{🍁}}(\Delta t)$ and $\hat{U}_{\text{🍁}}^{-1}(\Delta t)$, generating the same nontrivial evolution whether the system is propagated forwards or backwards in time, inconsistent with the known time reversibility of $\hat{U}_{\text{🍁}}$. Note that this argument does not work for the previously proposed first-order accurate approximations to $\hat{U}_{\text{🍁}}$, $\hat{K}\hat{D}$, because here, time reversal meant not just negating Δt but also effectively

swapping $\hat{T} \leftrightarrow \hat{V}$. As can be seen in (3.21) this swap introduces a minus sign in the second order term due to the commutator. The same happens for all higher, even terms.

The expansion of $\hat{U}_{\text{🦔}}$ (3.24) argued for above do indeed exist and is derived in [20]:

$$\begin{aligned}
\hat{U}_{\text{🦔}}(\Delta t) &= \exp\biggl(\int_t^{t+\Delta t/2} \hat{V}\bigl(t'\bigr)\,\mathrm{d}t'\biggr) \exp\biggl(\int_t^{t+\Delta t} \hat{T}\bigl(t'\bigr)\,\mathrm{d}t'\biggr) \\
&\qquad\qquad \times \exp\biggl(\int_t^{t+\Delta t/2} \hat{V}\bigl(t'\bigr)\,\mathrm{d}t'\biggr) \\
&= \exp\biggl(\int_t^{t+\Delta t} \Bigl(\hat{T}\bigl(t'\bigr) + \hat{V}\bigl(t'\bigr)\Bigr)\,\mathrm{d}t' \\
&\qquad + \frac{1}{24} \iiint_t^{t+\Delta t} \left\{ \begin{array}{l} 2\Bigl[\hat{T}\bigl(t'\bigr), \bigl[\hat{T}\bigl(t''\bigr), \hat{V}\bigl(t'''\bigr)\bigr]\Bigr] \\ -\Bigl[\hat{V}\bigl(t'\bigr), \bigl[\hat{V}\bigl(t''\bigr), \hat{T}\bigl(t'''\bigr)\bigr]\Bigr] \end{array} \right\} \mathrm{d}t'\,\mathrm{d}t''\,\mathrm{d}t''' \\
&\qquad + \mathcal{O}\bigl(\Delta t^5\bigr)\biggr),
\end{aligned}$$

where the three integrals all share the same limits. This expansion do indeed conform to the above argument, and it confirms that the time evolution operator $\hat{U}_{\text{🦔}}$ (3.24) is a second order accurate approximations to $\hat{U}$ (3.21). The time evolution operator (3.24) is known as the kick-drift-kick (KDK) time evolution operator, while its $\hat{K} \leftrightarrow \hat{D}$ dual is known as the drift-kick-drift (DKD) time evolution operator. Because positions and momenta are updated at interleaved times within a single time step, $\vec{\boldsymbol{x}}$ and $\vec{\boldsymbol{p}}$ temporally "leapfrog" over each other. The KDK and DKD integrators are therefore known as leapfrog integrators, which finally explains the amphibious subscript on $\hat{U}_{\text{🦔}}$.

As $\hat{U}_{\text{🦔}}(\Delta t)$ is only accurate to second order in Δt, it is of course important to choose the time step size Δt small. To evolve the system an amount $N_{\text{ts}}\Delta t \gg \Delta t$ forwards in time, $\hat{U}_{\text{🦔}}(\Delta t)$ is simply applied many times:

$$\begin{aligned}
\hat{U}(N_{\text{ts}}\Delta t) &\approx \overbrace{\hat{U}_{\text{🦔}}(\Delta t)\hat{U}_{\text{🦔}}(\Delta t)\cdots\hat{U}_{\text{🦔}}(\Delta t)}^{N_{\text{ts}}\text{ times}} \\
&= \hat{K}\left(\frac{\Delta t}{2}\right)\hat{D}(\Delta t)\hat{K}\left(\frac{\Delta t}{2}\right)\hat{K}\left(\frac{\Delta t}{2}\right)\hat{D}(\Delta t)\hat{K}\left(\frac{\Delta t}{2}\right) \\
&\qquad\qquad \cdots \hat{K}\left(\frac{\Delta t}{2}\right)\hat{D}(\Delta t)\hat{K}\left(\frac{\Delta t}{2}\right) \\
&= \hat{K}\left(\frac{\Delta t}{2}\right)\hat{D}(\Delta t) \\
&\qquad \times \underbrace{\hat{K}(\Delta t)\hat{D}(\Delta t)\hat{K}(\Delta t)\hat{D}(\Delta t)\cdots\hat{K}(\Delta t)\hat{D}(\Delta t)}_{(N_{\text{ts}}-1)\text{ times}}\hat{K}\left(\frac{\Delta t}{2}\right),
\end{aligned} \tag{3.25}$$

where the composition property* of the kick operator has been used to write $\hat{K}(\Delta t/2)\hat{K}(\Delta t/2) = \hat{K}(\Delta t)$. We shall refer to (3.25) as the KDK leapfrog integration scheme. Again, the DKD leapfrog integration scheme, where we begin and end with a drift operator, is equally valid. The leapfrog integration scheme (3.25) is not only symplectic and second order accurate, but also very computationally efficient. Except for the first and last operation, time integration is done through an alternating series of drift $\hat{D}(\Delta t)$ and kick $\hat{K}(\Delta t)$ operators, exactly as would be the case for the first order accurate scheme first considered. That means that the upgrade from first to second order accuracy only costs one additional kick/drift operation for the entire simulation run!

3.2.3 Time Step Size

We need to define the time step size Δt, used by the symplectic leapfrog integrator (3.25). The basic criterion is that Δt should be small enough to accurately resolve the dynamics. How one chooses to quantify this can of course be done in many different ways. A time scale common to all particles is the instantaneous age of the Universe. Naturally, Δt should be chosen smaller than this age, and so

$$\Delta t(t) = \beta t \tag{3.26}$$

may be used to define the time step size, where $\beta \ll 1$ is some constant controlling the accuracy of the time integration and t is the instantaneous age of the Universe. We then see that the time step size increases as the Universe evolve. Not having a constant Δt is very important for the efficiency of the code, as otherwise computational resource are being wasted on resolving time finer than what is needed. It should be noted that having a finite time step size suppresses structure at small scales, as gravity "has no time to act" for scales below $\Delta t|\boldsymbol{p}|/m$. It is then of no use having a good spatial resolution (large N) without a correspondingly good temporal resolution (small β). Determining Δt solely based on the instantaneous age of the Universe is however rather primitive, as different particles require different Δt in order to produce results of similar accuracy. In the CONCEPT code, the bare global criterion (3.26) is all that is implemented.

The errors produced in the equations of motion due to the finite time step is different for each particle. The dynamical time scales for particles in overdense regions are vastly smaller than those of particles in underdense regions. To achieve similar accuracies across all particles then, individual and adaptive values of Δt are needed. Determining the relative density within the region surrounding a given particle is not straight forward, and so often a tracer for the over/underdensity is used instead. Tracers easy to come by are the particle velocity and acceleration: Generally, particles in overdense regions have larger

*Which follows trivially from its definition (3.24) and the additive property of the integral.

velocities and accelerations. In GADGET-2, the individual time step sizes are then determined by*

$$\Delta t(t) = \beta t \min\left(1,\, 2^{\left\lfloor \log_2\left(\frac{1}{\beta t}\sqrt{\frac{2\eta\epsilon}{|\boldsymbol{a}|}}\right)\right\rfloor}\right). \tag{3.27}$$

where ϵ is the softening length, $|\boldsymbol{a}|$ is the current particle acceleration and η is a dimensionless parameter, controlling the accuracy. The expression within the logarithm sets $\Delta t(t)$ to the time it would take the particle to travel the distance $\eta\epsilon$, if initially at rest. For the case of parallel acceleration and momentum, η is then the fractional error in the position update due to having a finite time step size. If the time step size computed in this way increases beyond the global time step size βt, it is capped at this maximum value. In addition, the time step size is rounded down in such a way as to force $\Delta t(t)$ to be discretized in a power of two hierarchy, where $\Delta t(t) \in \{2^n \beta t \,|\, n \in \mathbb{N}\}$. This discretization is important once we allow for individual time step sizes, as it allows for synchronization time points, where all particles have been evolved equally. A series of leapfrog integrators, each with its own n, are then used to evolve the particle system. At synchronization points, the particles can then be transferred to another integrator, if the needed time step size for the particle has changed. For the power of two hierarchy of (3.27), a particle can then always jump down to the integrator with half it's current time step size, but up to the integrator of double the current time step size only at every second time step. As found in [21], allowing for individual and adaptive time steps can potentially lead to speedups as large as ~ 50.

The reason for introducing individual and adaptive time steps in N-body codes comes down to the fact that the rate of change of the gravitational force is different for each particle, determined by the local surroundings of the particle. For the gravitational methods utilizing force splitting (e.g. the P^3M method), only the short-range need to be treated in a similar way. That is, individual time step sizes are not needed for the long-range force, as its rate of change is similar for all particles due to the homogeneity on large scales. In addition, changes in the long-range force occurs at a larger time scale than changes in the short-range force, which again can be attributed to the more uniform mass distribution encountered at larger scales. To optimize the time integration yet further then, the adaptive time integration is used only for the short-range forces, while the long-range forces are supplied only when all the hierarchical integrators are synchronized. That is, the time step size for the long-range force is simply that of (3.26). This rather neat interplay between the gravitational solver and the time integration is what is implemented in GADGET-2.

*To be precise, GADGET-2 replaces the instantaneous age of the Universe for the instantaneous Hubble time. Also, GADGET-2 additionally implements an alternative, non-hierarchical scheme.

Allowing for individual time steps breaks strict symplecticity, as the force on a particle at a given time is now dependent upon the state of other particles at *different* times, destroying time reversibility. The accompanying non-Hamiltonian perturbation to the Hamiltonian is however small enough that these non-Hamiltonian perturbations can be ignored, if using a small enough time step. Another very interesting effect of allowing for individual and adaptive time steps, is that the KDK and the DKD leapfrog integrators now differ significantly in accuracy. For the KDK leapfrog, the acceleration $|\boldsymbol{a}|$ — used in the determination (3.27) of the size of the upcoming time step — is that of the current time, since a complete time step consists of a drift followed by a kick. In the DKD leapfrog however, a complete time step consists of a kick followed by a drift, which means that $|\boldsymbol{a}|$ is the acceleration of the particle from half a time step ago. The adjusted time step sizes are then determined using a false premise, further increasing the size of the non-Hamiltonian perturbations by a factor of 2 [1]. The KDK leapfrog is thus superior to the DKD leapfrog, once we allow for individual and adaptive time step sizes.

4 Review and Final Remarks

The goal of this work have been to study the internals of cosmological N-body codes, theoretically as well as practically. A theoretical understanding of the physics that these codes strive to emulate has been developed. We have found that Newtonian mechanics appended with the expansion of space, itself incorporated via the introduction of comoving coordinates, accurately describes the evolution of the matter within the Universe. A reoccurring surprise was the strange behaviour of the Newtonian gravitational potential in an infinite universe, which had to be regulated using Jean's swindle to remain well defined. We saw that the N-body approach were not needed until about the beginning of matter domination, as prior to this during the radiation era, linear perturbation theory were fully adequate.

A theoretical understanding of the methods used by N-body codes has also been obtained. An infinite universe could be emulated using a periodic box. This could either be put in by hand as in the Ewald method, or achieved automatically using Fourier techniques. Having a finite N led to collisions between the particles, which were supposed to represent collisionless fluid elements. To counteract these collisions, the gravitational potential were softened. Again, this could either be done by hand by representing the particles as Plummer spheres rather than delta functions, or achieved automatically by discretizing space, using mesh methods. A general trick used by the gravity solvers were to split the $1/\boldsymbol{r}^2$ force into a long-range and a short-range component, which were then solved in Fourier and real space, respectively. This splitting served different purposes for the different methods. In the Ewald method, this force split were responsible for the absolute (and fast) convergence of the summation. In the P^3M method, the force split was used to effectively remove the discretization of space imposed by the mesh (at least for particles near each other), which greatly approved upon the accuracy of the albeit fast PM method.

Numerical time evolution was studied in detail. It turned out that to integrate a Hamiltonian system forwards in time, special considerations were needed in order not to perturb the Hamiltonian. We derived the widely used leapfrog integration scheme and discussed the surprising result of the superiority of the KDK leapfrog over its DKD dual, once individual and adaptive time steps were allowed.

All of the methods studied in detail has been implemented into the CONCEPT code. These implementations have been tested against each other and against the GADGET-2 code. The development of the CONCEPT code has resulted in a successful N-body simulator, capable of evolving dark matter particles in any cosmology. It has been written in a highly structured form, anticipating future additions. Besides its primary functionality of evolving a system of particles in time, other features have also been developed. For input/output of initial conditions and snapshots, concept uses its own HDF5 format. To allow for compatibility with GADGET-2, the (second kind of) snapshot format of GADGET-2 is additionally implemented. The functionalities used to produce some of the plots of this thesis are also part of the CONCEPT code. Specifically, the raw box visualizations in Figure 2.4 as well as the decomposition of the particle configuration into a power spectra, as seen plotted in e.g. Figure 2.5, have been produced directly in CONCEPT.

References

[1] Volker Springel. The cosmological simulation code GADGET-2. *Monthly Notices of the Royal Astronomical Society*, 364, 2005.

[2] P.A.R. Ade et al. Planck 2015 results. XIII. Cosmological parameters. 2015.

[3] Barbara Ryden. *Introduction to Cosmology*, page 67. Addison-Wesley, 2003.

[4] Antony Lewis, Anthony Challinor, and Anthony Lasenby. Efficient computation of CMB anisotropies in closed FRW models. *Astrophys. J.*, 538:473–476, 2000. http://camb.info.

[5] W. H. Press and P. Schechter. Formation of Galaxies and Clusters of Galaxies by Self-Similar Gravitational Condensation. *Astrophysical Journal*, 187:425–438, February 1974.

[6] P. P. Ewald. Die berechnung optischer und electrostatischer gitterpotentiale. *Annalen der Physik*, 369, 1921.

[7] M. J. L. Sangster and M. Dixon. Interionic potentials in alkali halides and their use in simulations of the molten salts. *Advances in Physics*, 25:247, 1976.

[8] L. Hernquist, F. R. Bouchet, and Y. Suto. Application of the Ewald method to cosmological N-body simulations. *The Astrophysical Journal*, 75:234, February 1991.

[9] H. C. Plummer. On the problem of distribution in globular star clusters. *Monthly Notices of the Royal Astronomical Society*, 71:460–470, March 1911.

[10] A. Knebe. Lecture notes *Computational Astrophysics: Physical Processes.* http://popia.ft.uam.es/aknebe/page3/files/ComputationalAstrophysics/PhysicalProcesses.pdf, 85, 2005.

[11] E. Puchwein and B. Moster. Lecture notes *Cosmic Structure formation on Supercomputers (and laptops), Lecture 2: Gravity solvers and parallelization.* http://www.ast.cam.ac.uk/~puchwein/NumericalCosmology02.pdf, 13, 2014.

[12] J. Brandbyge. *Quantitative cosmology - Massive neutrinos in non-linear structure formation.* PhD thesis, The Faculty of Science, Aarhus University, Denmark, August 2010, http://phys.au.dk/fileadmin/site_files/publikationer/phd/Jacob_Brandbyge.pdf.

[13] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles.* Taylor & Francis, Inc., Bristol, PA, USA, 1988.

[14] Juhan Kim, Changbom Park, Graziano Rossi, Sang Min Lee, and III Gott, J. Richard. The New Horizon Run Cosmological N-Body Simulations. *J.Korean Astron.Soc.*, 44(issue):217–234, 2011.

[15] Thomas Sauer and Yuan Xu. On multivariate lagrange interpolation. *MATH. COMP*, 64:1147–1170, 1994.

[16] Matteo Frigo, Steven, and G. Johnson. The design and implementation of FFTW3. In *Proceedings of the IEEE*, pages 216–231, 2005.

[17] J. V. Villumsen. A new hierachical particle-mesh code for very large scale cosmological N-body simulations. *Astrophys. J.*, 71:407–431, November 1989.

[18] Chris Jessop, Martin Duncan, and W.Y. Chau. Multigrid methods for n-body gravitational systems. *Journal of Computational Physics*, 115(2):339 – 351, 1994.

[19] Michael Joyce. Infinite self-gravitating systems and cosmological structure formation. *AIP Conf.Proc.*, 970:237, 2008.

[20] Dion O'Neale. Split-step methods for nonlinear highly oscillatory problems. Master's thesis, Mathematischen Institute der Heinrich-Heine-Universität, Düsseldorf, Deutschland, June 2005.

[21] Thomas R. Quinn, Neal Katz, Joachim Stadel, and George Lake. Time stepping N body simulations. *Astrophys.J.*, 1997.