Appendix B

Simulating collisionless dark matter fluids

Contents

> "Simulation:
> **1. a.** The action or practice of simulating, with intent to deceive; false pretence, deceitful profession. (...)
> **2.** A false assumption or display, a surface resemblance or imitation, of something. (...)"
> — The Oxford English Dictionary
> Quoted by Peter Coles (2014)

Abstract

This technical appendix describes the implementation of the simulation codes used in this thesis. It reviews the particle-mesh approach for simulating a collisionless cold dark matter fluid, as well as the COLA modification. The generation of initial conditions using Lagrangian perturbation theory is also discussed.

Many of the projects described in this thesis rely on the particle-mesh (PM) simulation technique. It has originally been introduced and applied in many different areas of physics, such as electromagnetism, hydrodynamics, magnetohydrodynamics, plasma physics and self-gravitating systems (see e.g. the books by Hockney & Eastwood, 1981 and Birdsall & Langdon, 1985). In a cosmological context, the reference papers include Klypin & Shandarin (1983); Efstathiou *et al.* (1985).

This appendix reviews the PM technique, the COLA modification, and the numerical implementation of Lagrangian perturbation theory. More details on cosmological PM codes can be found in the review by Bertschinger (1998) or the lectures notes by Kravtsov (2002); Springel (2014); Teyssier (2014). The reader is also referred

to the COLA papers, Tassev, Zaldarriaga & Eisenstein (2013); Tassev *et al.* (2015); and to Scoccimarro (1998, appendix D), for the implementation of LPT.

This appendix is organized as follows. In section B.1, we write down the equations actually solved by PM/COLA codes. We describe the main PM steps and the required data structures in section B.2. Section B.3 reviews mesh assignments and interpolation schemes; section B.4 discusses the resolution of the Poisson equation and the computation of forces; and section B.5 examines how to update the positions and momenta of particles. Finally, B.6 describes the generation of cosmological initial conditions using Lagrangian perturbation theory.

B.1 Model equations

B.1.1 Model equations in the standard PM code

A PM codes solves the equation of motion for dark matter particles in comoving coordinates (see equation (1.74); below the mass of particles m is absorbed in the definition of the momentum $\mathbf{p}$):

$$\mathbf{p} = a\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau}, \tag{B.1}$$

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}\tau} = -a\nabla\Phi, \tag{B.2}$$

coupled with the Poisson equation for the gravitational potential (equation (1.72)),

$$\Delta\Phi = 4\pi\mathrm{G}a^2\bar{\rho}(\tau)\delta = \frac{3}{2}\Omega_{\mathrm{m}}(\tau)\mathcal{H}^2(\tau)\delta. \tag{B.3}$$

It is convenient to choose the scale factor as time variable. Using $\partial_\tau = a'\,\partial_a = \dot{a}a\,\partial_a$ and $\bar{\rho}(\tau) = \rho^{(0)}a^{-3}$, the equations to solve are rewritten:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}a} = \frac{\mathbf{p}}{a'a} = \frac{\mathbf{p}}{\dot{a}a^2}, \tag{B.4}$$

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}a} = -\frac{a\nabla\Phi}{a'} = -\frac{\nabla\Phi}{\dot{a}}, \tag{B.5}$$

$$\Delta\Phi = 4\pi\mathrm{G}\rho^{(0)}a^{-1}\delta = \frac{3}{2}\Omega_{\mathrm{m}}^{(0)}a^{-1}\delta. \tag{B.6}$$

We will use the equivalent formulation

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}a} = \mathcal{D}(a)\mathbf{p}, \tag{B.7}$$

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}a} = \mathcal{K}(a)\nabla\left(\Delta^{-1}\delta\right), \tag{B.8}$$

where we have combined equations (B.5) and (B.6), and defined $f(a) \equiv \dot{a}^{-1} = a/a' = \mathcal{H}^{-1}(a)$; $\mathcal{D}(a) \equiv f(a)/a^2$ (the "drift prefactor") and $\mathcal{K}(a) \equiv -(3/2)\Omega_{\mathrm{m}}^{(0)}f(a)/a$ (the "kick prefactor").

B.1.2 Model equations with COLA

If one desires to include the COLA scheme (see Tassev, Zaldarriaga & Eisenstein, 2013, and section 7.3.1), then one works in a frame comoving with the Lagrangian displacements. Recall the LPT position of a particle is given by (see section 1.5),

$$\mathbf{x}_{\mathrm{LPT}}(a) = \mathbf{q} - D_1(a)\boldsymbol{\Psi}_1 + D_2(a)\boldsymbol{\Psi}_2. \tag{B.9}$$

Noting $\mathbf{x}(a) = \mathbf{x}_{\mathrm{LPT}}(a) + \mathbf{x}_{\mathrm{MC}}(a)$ the real position of the same particle, including the mode-coupling residual $\mathbf{x}_{\mathrm{MC}}(a)$, one has (see equation (B.9)):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}a} = \frac{\mathrm{d}\mathbf{x}_{\mathrm{LPT}}}{\mathrm{d}a} + \frac{\mathrm{d}\mathbf{x}_{\mathrm{MC}}}{\mathrm{d}a}; \quad \text{with} \quad \frac{\mathrm{d}\mathbf{x}_{\mathrm{LPT}}}{\mathrm{d}a} = -\frac{\mathrm{d}D_1}{\mathrm{d}a}\boldsymbol{\Psi}_1 + \frac{\mathrm{d}D_2}{\mathrm{d}a}\boldsymbol{\Psi}_2 \equiv \mathcal{D}(a)\mathbf{p}_{\mathrm{LPT}}. \tag{B.10}$$

We also define $\mathbf{p}_{\mathrm{MC}}$ such that $\mathrm{d}\mathbf{x}_{\mathrm{MC}}/\mathrm{d}a \equiv \mathcal{D}(a)\mathbf{p}_{\mathrm{MC}}$. Then $\mathbf{p} = \mathbf{p}_{\mathrm{LPT}} + \mathbf{p}_{\mathrm{MC}}$ (see equation (B.7)). Furthermore,

$$\frac{\mathrm{d}\mathbf{p}_{\mathrm{LPT}}}{\mathrm{d}a} = \frac{\mathrm{d}}{\mathrm{d}a}\left(\frac{1}{\mathcal{D}(a)}\frac{\mathrm{d}\mathbf{x}_{\mathrm{LPT}}}{\mathrm{d}a}\right) \equiv -\mathcal{K}(a)\mathcal{V}[\mathbf{x}_{\mathrm{LPT}}](a), \tag{B.11}$$

where the differential operator $\mathcal{V}[\cdot](a)$ is defined by

$$\mathcal{V}[\cdot](a) \equiv -\frac{1}{\mathcal{K}(a)}\frac{\mathrm{d}}{\mathrm{d}a}\left(\frac{1}{\mathcal{D}(a)}\frac{\mathrm{d}\cdot}{\mathrm{d}a}\right). \tag{B.12}$$

With these notations, equation (B.8) reads

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}a} = \frac{\mathrm{d}\mathbf{p}_{\mathrm{LPT}}}{\mathrm{d}a} + \frac{\mathrm{d}\mathbf{p}_{\mathrm{MC}}}{\mathrm{d}a} = -\mathcal{K}(a)\mathcal{V}[\mathbf{x}_{\mathrm{LPT}}](a) + \frac{\mathrm{d}\mathbf{p}_{\mathrm{MC}}}{\mathrm{d}a} = \mathcal{K}(a)\nabla\left(\Delta^{-1}\delta\right). \tag{B.13}$$

It is straightforward to check from equation (B.9) that $\mathcal{V}[\mathbf{x}_{\mathrm{LPT}}](a) = -\mathcal{V}[D_1](a)\mathbf{\Psi}_1 + \mathcal{V}[D_2](a)\mathbf{\Psi}_2$. Using the differential equation verified by D_1 (equation (1.96)) and the second Friedmann equation (equation (1.7)), we get

$$\mathcal{V}[D_1](a) = D_1(a). \tag{B.14}$$

Similarly for the second-order growth factor, using equation (1.118),

$$\mathcal{V}[D_2](a) = D_2(a) - D_1^2(a). \tag{B.15}$$

In the COLA framework, the natural variables are therefore $\mathbf{x}$ and $\mathbf{p}_{\mathrm{MC}}$, and the equations of motion to solve (equivalents of equations (B.7) and (B.8)) are

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}a} = \mathcal{D}(a)\mathbf{p}_{\mathrm{MC}} - \frac{\mathrm{d}D_1}{\mathrm{d}a}\mathbf{\Psi}_1 + \frac{\mathrm{d}D_2}{\mathrm{d}a}\mathbf{\Psi}_2, \tag{B.16}$$

$$\frac{\mathrm{d}\mathbf{p}_{\mathrm{MC}}}{\mathrm{d}a} = \mathcal{K}(a)\left[\nabla\left(\Delta^{-1}\delta\right) - \mathcal{V}[D_1](a)\mathbf{\Psi}_1 + \mathcal{V}[D_2](a)\mathbf{\Psi}_2\right]. \tag{B.17}$$

In the initial conditions, generated with LPT (see section B.6), we have $\mathbf{p} = \mathbf{p}_{\mathrm{LPT}}$; therefore the mode-coupling momentum residual in the rest frame of LPT observers, $\mathbf{p}_{\mathrm{MC}}$, should be initialized to zero (this corresponds to the L_- operator in Tassev, Zaldarriaga & Eisenstein, 2013, appendix A). At the end, the LPT momentum $\mathbf{p}_{\mathrm{LPT}}$ has to be added to $\mathbf{p}_{\mathrm{MC}}$ to recover the full momentum of particles, $\mathbf{p}$ (this corresponds to the L_+ operator in Tassev, Zaldarriaga & Eisenstein, 2013, appendix A). In the following, wherever we do not make the explicit distinction between the standard PM and the COLA approaches, we will drop the subscript "MC" for COLA momenta and simply note $\mathbf{p}$; however, one should keep in mind these two transformations at the beginning and at the end.

B.2 Steps and data structures

B.2.1 Main PM steps

Equations (B.7) and (B.8) are solved iteratively in a PM code, which consists of three main steps:

1. estimate the density field on the grid from current particle positions; solve the Poisson equation on the grid to get the potential; take the gradient of the potential to get the accelerations on the grid; and interpolate back to particles (see sections B.3 and B.4),
2. advance particle momenta using the new accelerations (equation (B.8); see section B.5)
3. update particle positions using their new momenta (equation (B.7); see section B.5).

In the COLA scheme, steps 2 and 3 are replaced with the equivalents that come from equations (B.17) and (B.16), respectively.

B.2.2 Definitions and data structures

Grids and box size. A PM cosmological simulation is characterized by

- the number of particles, N_{p} (if particles start from a regular Lagrangian grid – see section B.6 –, we note N_{p0}, N_{p1}, N_{p2} the number of particles along each direction, such that $N_{\mathrm{p}} \equiv N_{\mathrm{p0}}N_{\mathrm{p1}}N_{\mathrm{p2}}$);

- the size of the periodic box along each direction, L_0, L_1, L_2 (the total volume simulated is therefore $V \equiv L_0 L_1 L_2$);
- and the number of cells of the PM grid (i.e. the grid on which density and potential are defined) along each direction, $N_{\mathrm{g}0}$, $N_{\mathrm{g}1}$, $N_{\mathrm{g}2}$, with $N_{\mathrm{g}} \equiv N_{\mathrm{g}0} N_{\mathrm{g}1} N_{\mathrm{g}2}$.

In many cases we will assume that the box is cubic, and that the particle grid and the PM grid are isotropic: $L_0 = L_1 = L_2 \equiv L$; $N_{\mathrm{p}0} = N_{\mathrm{p}1} = N_{\mathrm{p}2}$; $N_{\mathrm{g}0} = N_{\mathrm{g}1} = N_{\mathrm{g}2}$. In the following, we denote the side lengths of cells by $\Delta x \equiv L_0/N_{\mathrm{g}0}$, $\Delta y \equiv L_1/N_{\mathrm{g}1}$, $\Delta z \equiv L_2/N_{\mathrm{g}2}$ and their volume by $V_{\mathrm{c}} \equiv \Delta x \Delta y \Delta z$. We have $V = N_{\mathrm{g}} V_{\mathrm{c}}$.

Particle variables. Assuming that particles all have the same mass,[1] a PM code needs a minimum of six real numbers (`float` or `double`) for each particle: three coordinates and three momenta. If the COLA modification is included (see section 7.3.1), a minimum of nine (for LPT at order one) or twelve (for LPT at order two) real numbers per particle is required (three additional real numbers per particle to store the LPT displacements at each order).

We call these arrays `x[mp]`, `y[mp]`, `z[mp]` (particles' positions); `px[mp]`, `py[mp]`, `pz[mp]` (particles' momenta); and if COLA is enabled, `psix_1[mp]`, `psiy_1[mp]`, `psiz_1[mp]` (for the ZA displacements, $\boldsymbol{\Psi}_1$), `psix_2[mp]`, `psiy_2[mp]`, `psiz_2[mp]` (for the 2LPT displacements, $\boldsymbol{\Psi}_2$). Here `mp` indexes a particle. It is interesting to note that the arrays containing the Lagrangian displacements are constants, i.e. that they are never updated within the code (their time-independence can be checked in equations (B.16) and (B.17)). Convenient data structures are 1D arrays of size N_{p} for particles' variables.

Grid variables. In addition, the code needs real numbers (`float` or `double`) for the density contrast δ and the potential Φ at each grid cell. An array of size N_{g} is needed to store such grid variables. This array can be shared between density and potential: we first use it to store the density contrast δ, then replace its values with the potential when the Poisson equation is solved.[2]

We call this array `density_or_Phi`. A convenient data structure is a 3D array, such that the grid quantity at position (i, j, k) is `density_or_Phi[i,j,k]` (with $0 \leq i < N_{\mathrm{g}0}$, $0 \leq j < N_{\mathrm{g}1}$, $0 \leq k < N_{\mathrm{g}2}$). Equivalently, we decided to implement `density_or_Phi` as a 1D array of size N_{g}, such that the grid quantity at position (i, j, k) is given by `density_or_Phi[mc]` where the current cell is indexed by $\texttt{mc} = k + N_{\mathrm{g}2} \times (j + N_{\mathrm{g}1} \times i)$.

Accelerations. It is also convenient to have three additional arrays of size N_{g} to store the components of the acceleration on the grid, and three arrays of size N_{p} to store the components of particles' acceleration.[3] In the following, we note these arrays `gx[mc]`, `gy[mc]`, `gz[mc]`, `gpx[mp]`, `gpy[mp]`, `gpz[mp]`, where $0 \leq \texttt{mc} < N_{\mathrm{g}}$ indexes a grid cell and $0 \leq \texttt{mp} < N_{\mathrm{p}}$ indexes a particle.[4]

B.3 Mesh assignments and interpolations

This section describes how to assign to the grid a quantity carried by particles (the "mesh assignment" operation, from particles to the grid), and how to distribute to particles a quantity that is known on the grid (the "interpolation" operation, from the grid to particles).

In a PM code, the first operation is used to compute the density on the grid from particle positions; and the second operation is used to assign an acceleration to each particle from grid values. Both are used in step 1 of the main PM steps (see section B.2.1).

---

[1] From the definition of $\Omega_{\mathrm{m}}^{(0)}$, it is easy to see that the mass carried by each particle is $m = \dfrac{3\Omega_{\mathrm{m}}^{(0)} H_0^2}{8\pi \mathrm{G}} \dfrac{V}{N_{\mathrm{p}}}$ (this number is called the *mass resolution*).

[2] The quantity stored is actually the reduced gravitational potential, $\widetilde{\Phi} \equiv \Delta^{-1}\delta$, as the overall time-dependent coefficients needed to go from $\widetilde{\Phi}$ to Φ are factored out in $\mathcal{K}(a)$ (see equations (B.8) and (B.17)).

[3] Actually the reduced acceleration $\tilde{g} \equiv \nabla\left(\Delta^{-1}\delta\right)$ instead of the physical acceleration, see footnote 2.

[4] These arrays are not absolutely required. Indeed, it is possible to get rid of them and to make the code more memory-efficient, if one performs in one step the finite difference (to go from $\Delta^{-1}\delta$ to $\nabla(\Delta^{-1}\delta)$), the interpolation (from the grid quantities to particles, see section B.3) and the kick operation (see section B.5).

B.3.1 The mesh assignment function

The general idea to assign particles to the grid is to assume that they have a "shape" S that intersects the grid. Let us first describe the one-dimensional case, where $S(x)$ is the 1D particle shape. The fraction of the particle at x_p assigned to the cell at x_c is the shape function averaged over this cell:

$$W(x_\mathrm{p} - x_\mathrm{c}) \equiv \int_{x_\mathrm{c}-\Delta x/2}^{x_\mathrm{c}+\Delta x/2} S(x' - x_\mathrm{p})\,\mathrm{d}x' = \int \Pi\left(\frac{x' - x_\mathrm{c}}{\Delta x}\right) S(x' - x_\mathrm{p})\,\mathrm{d}x' \tag{B.18}$$

The assignment function is hence the convolution:

$$W(x) = \Pi\left(\frac{x}{\Delta x}\right) * S(x) \quad \text{where} \quad \Pi(s) = \begin{cases} 1 & \text{if } |s| \leq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases} \tag{B.19}$$

In 3D,

$$W(\mathbf{x}_\mathrm{p} - \mathbf{x}_\mathrm{c}) \equiv W(x_\mathrm{p} - x_\mathrm{c})W(y_\mathrm{p} - y_\mathrm{c})W(z_\mathrm{p} - z_\mathrm{c}). \tag{B.20}$$

For some quantity A, if A_p are the values carried by the particles at positions $\mathbf{x}_p$, the quantity A at position $\mathbf{x}_\mathrm{c}$ on the grid is

$$A(\mathbf{x}_\mathrm{c}) = \sum_{\{\mathbf{x}_\mathrm{p}\}} A_\mathrm{p} W(\mathbf{x}_\mathrm{p} - \mathbf{x}_\mathrm{c}). \tag{B.21}$$

In particular, for gravitational PM codes, the quantity carried by particles is their mass m. The density on the mesh is then a sum over the contributions of each particle as given by the assignment function,

$$\rho(\mathbf{x}_\mathrm{c}) = \frac{1}{V_\mathrm{c}} \sum_{\{\mathbf{x}_\mathrm{p}\}} m W(\mathbf{x}_\mathrm{p} - \mathbf{x}_\mathrm{c}). \tag{B.22}$$

The mean density is $\bar{\rho} = mN_\mathrm{p}/V$, from which we deduce the density contrast $\delta \equiv \rho/\bar{\rho} - 1$ on the mesh,

$$\delta(\mathbf{x}_\mathrm{c}) = \left(\frac{N_\mathrm{g}}{N_\mathrm{p}} \sum_{\{\mathbf{x}_\mathrm{p}\}} W(\mathbf{x}_\mathrm{p} - \mathbf{x}_\mathrm{c})\right) - 1. \tag{B.23}$$

B.3.2 Low-pass filtering

The Nyquist-Shannon sampling theorem (Nyquist, 1928; Shannon, 1948, 1949) states that the information content of a sampled signal can be correctly recovered if two conditions hold: the signal must be band-limited, and the sampling frequency must be greater than twice the maximum frequency present in the signal. If this is not the case, replicated spectra cannot be separated of the signal we seek to recover, a phenomenon known as aliasing (e.g. Manolakis, Ingle & Kogon, 2000). Natural signals, however, are generally not band-limited, so must be low-pass filtered before they are sampled. Equivalently, the sampling operation must include some form of local averaging, reflecting the finite spatial resolution.

The Fourier representation[5] of the ideal low-pass filter that one should use as assignment function is given as

$$W(k) = \frac{1}{\sqrt{2\pi}} \Pi\left(\frac{k}{k_{\mathrm{Nyq},x}}\right) = \frac{1}{\sqrt{2\pi}} \times \begin{cases} 1 & \text{if } |k| < k_\mathrm{max} \\ 0 & \text{if } |k| \geq k_\mathrm{max}, \end{cases} \tag{B.24}$$

where $k_\mathrm{max} \equiv k_{\mathrm{Nyq},x}/2$, and $k_{\mathrm{Nyq},x} \equiv 2\pi/\Delta x$ is the Nyquist wavenumber. This filter is ideal in the sense that it has unity gain in the pass-band region and it perfectly suppresses all the power in the stop-band regions. The configuration space representation is

$$W(x) = \frac{1}{\Delta x} \operatorname{sinc}\left(\frac{x}{\Delta x}\right), \tag{B.25}$$

where $s \mapsto \operatorname{sinc}(s) \equiv \frac{\sin(\pi s)}{\pi s}$ is the cardinal sine function (using the signal processing convention). It is interesting to note that $W(x)$ is not always positive. Therefore, the physical property of a continuous density field to be positive will not be reflected in its discretized representation, using ideal low-pass filtering. The loss of physicality is an expression of a fundamental problem of any data processing procedure: the loss of information due to discretizing the continuous signal.

[5] Here, we use the conventions for forward and inverse Fourier transforms as introduced in section 1.2.4.1.

Furthermore, due to the infinite support of the cardinal sine function in configuration space, the ideal sampling method is generally not tractable, because computationally too expensive. For this reason, practical approaches often rely on approximating the ideal cardinal sine operator by less accurate, but faster calculable functions (often with compact support in configuration space). In Fourier space, this will generally introduce artificial attenuation of the pass-band modes and leakage of stop-band modes into the signal (i.e. incomplete suppression of the aliasing power). The optimal choice of a low-pass filter approximation is therefore always a choice between accuracy and computational speed (see e.g. Manolakis, Ingle & Kogon, 2000, for detailed studies). In the following section we discuss common approaches used in particle simulations.

B.3.3 Common mesh assignment schemes

Commonly used particle shape functions and assignment schemes are often presented as a hierarchy (Hockney & Eastwood, 1981). The simplest scheme is to consider that particles are punctual and to assign each of them to the nearest grid point: $W(x_\mathrm{p} - x_\mathrm{c}) = 1$ if $x_\mathrm{c} - \frac{\Delta x}{2} \leq x_\mathrm{p} \leq x_\mathrm{c} + \frac{\Delta x}{2}$, 0 otherwise. The shape function is therefore

$$S_\mathrm{NGP}(x) \equiv \delta_\mathrm{D}(x) \quad \text{and} \quad S_\mathrm{NGP}(\mathbf{x}) \equiv \delta_\mathrm{D}(x)\delta_\mathrm{D}(y)\delta_\mathrm{D}(z). \tag{B.26}$$

This is the Nearest Grid Point (NGP) assignment scheme.

The second particle shape function in the hierarchy is a rectangular parallelepiped (a "cloud") of side length Δx, Δy, Δz. This scheme involves the 8 nearest cells for each particle and is called the Cloud-in-Cell (CiC) scheme. The shape function is

$$S_\mathrm{CiC}(x) \equiv \frac{1}{\Delta x}\Pi\left(\frac{x}{\Delta x}\right) \quad \text{and} \quad S_\mathrm{CiC}(\mathbf{x}) \equiv \frac{1}{\Delta x \Delta y \Delta z}\Pi\left(\frac{x}{\Delta x}\right)\Pi\left(\frac{y}{\Delta y}\right)\Pi\left(\frac{z}{\Delta z}\right). \tag{B.27}$$

This shape function can be seen as the convolution $\frac{1}{\Delta x}\Pi\left(\frac{x}{\Delta x}\right) * \delta_\mathrm{D}(x)$. Higher-order assignment schemes are obtained by successively convolving with $\frac{1}{\Delta x}\Pi\left(\frac{x}{\Delta x}\right)$ along each direction. For example, the third-order scheme is called the Triangular Shaped Cloud (TSC) and involves the 27 neighboring cells for each particle. In one-dimension, the shape function is

$$S_\mathrm{TSC}(x) \equiv \frac{1}{\Delta x}\Pi\left(\frac{x}{\Delta x}\right) * \frac{1}{\Delta x}\Pi\left(\frac{x}{\Delta x}\right). \tag{B.28}$$

The Fourier transform of $x \mapsto \frac{1}{\Delta x}\Pi\left(\frac{x}{\Delta x}\right)$ is $k \mapsto \frac{1}{\sqrt{2\pi}}\operatorname{sinc}\left(\frac{k}{k_{\mathrm{Nyq},x}}\right)$. Therefore, in Fourier space, building the hierarchy is taking successive powers of $\frac{1}{\sqrt{2\pi}}\operatorname{sinc}\left(\frac{k}{k_{\mathrm{Nyq},x}}\right)$. The assignment function W is found by an additional convolution of S with $x \mapsto \Pi\left(\frac{x}{\Delta x}\right)$, which means, in Fourier space, an additional multiplication by $\frac{\Delta x}{\sqrt{2\pi}} \times \operatorname{sinc}\left(\frac{k}{k_{\mathrm{Nyq},x}}\right)$. In figure B.1, we show the shape functions S for the NGP, CiC and TSC schemes (first row), the corresponding assignment functions W (second row) and their normalized Fourier transforms, $\hat{W}/\Delta x$ (rescaled such that $\hat{W}(k=0)/\Delta x = 1$; third row).

High order schemes are obviously more expensive numerically, but they also give more precise results: from equation (B.21) and the shape functions, we see that resulting quantities on the grid (density, forces) are piecewise constant in cells (NGP); C^0 and piecewise linear (CiC); C^1 with piecewise linear first derivative (TSC), etc. (see figure B.1). The choice is a tradeoff between accuracy and computational expense.

We summarize the results of this section in table B.1. In the following, we further comment on the well-known CiC scheme, which is the prescription used to assign particles to the grid throughout this thesis, including in PM and COLA implementations.

Let us consider the CiC density assignment for a particle with coordinates $(x_\mathrm{p}, y_\mathrm{p}, z_\mathrm{p})$. The cell containing the particle has indexes given by

$$i = \left\lfloor \frac{x_\mathrm{p}}{\Delta x} \right\rfloor; \quad j = \left\lfloor \frac{y_\mathrm{p}}{\Delta y} \right\rfloor; \quad k = \left\lfloor \frac{z_\mathrm{p}}{\Delta z} \right\rfloor, \tag{B.29}$$

where $\lfloor\cdot\rfloor$ is the integer floor function. We consider that the cell center is at $(x_\mathrm{c}, y_\mathrm{c}, z_\mathrm{c}) = (i\times\Delta x, j\times\Delta y, k\times\Delta z)$.[6]

[6] The other common convention is to displace the cell center by half a voxel with respect to $(i \times \Delta x, j \times \Delta y, k \times \Delta z)$, i.e. $(x_\mathrm{c}, y_\mathrm{c}, z_\mathrm{c}) = (i \times \Delta x + \frac{\Delta x}{2}, j \times \Delta y + \frac{\Delta y}{2}, k \times \Delta z + \frac{\Delta z}{2})$.

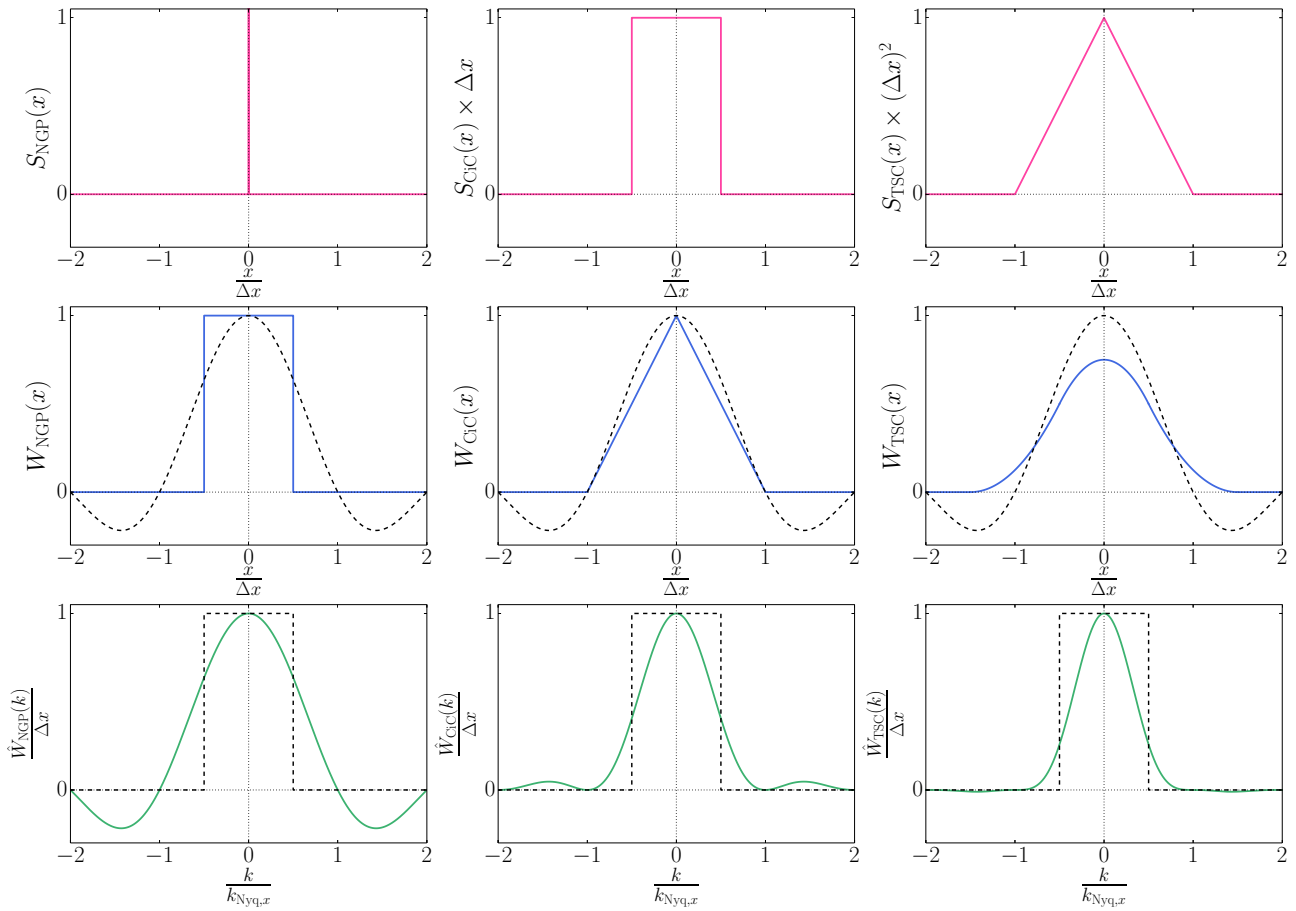

Figure B.1: Shape functions in configuration space for the first three schemes of the natural hierarchy of mesh assignments (S, first row); the corresponding assignment functions (W, second row) and their normalized Fourier transform ($\hat{W}$, third row). From left to right, the schemes are: Nearest Grid Point (NGP), Cloud-in-Cell (CiC), Triangular Shaped Cloud (TSC). The Nyquist wavenumber is defined by $k_{\mathrm{Nyq},x} \equiv 2\pi/\Delta x$. For comparison, the dashed black lines show the configuration and Fourier space representations of the ideal low-pass filter kernel.

Name	Shape function $S(x)$	Number of cells involved	Properties of grid-wise quantities
NGP	$\delta(x)$	$1^3 = 1$	Piecewise constant in cells
CiC	$\frac{1}{\Delta x}\Pi\left(\frac{x}{\Delta x}\right)$	$2^3 = 8$	C^0, piecewise linear
TSC	$\frac{1}{\Delta x}\Pi\left(\frac{x}{\Delta x}\right) * \frac{1}{\Delta x}\Pi\left(\frac{x}{\Delta x}\right)$	$3^3 = 27$	C^1, differentiable with piecewise linear derivative

Table B.1: Summary of the properties of commonly used particle shape functions.

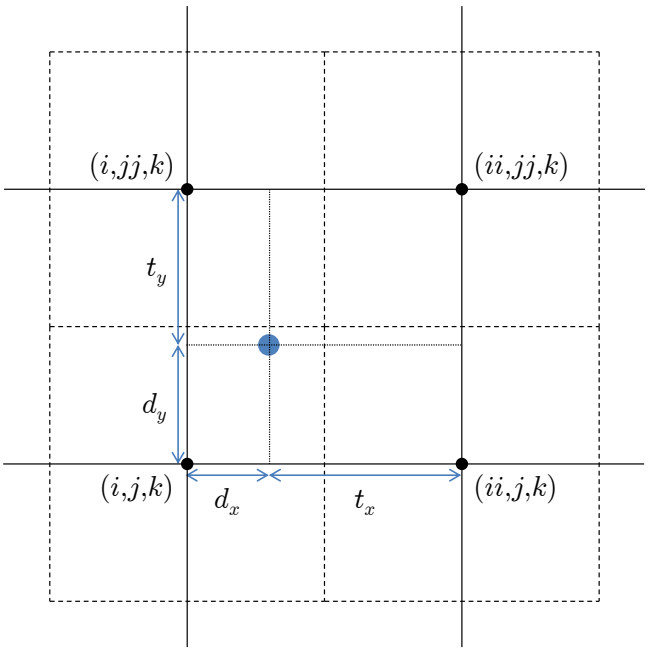

Figure B.2: Two-dimensional illustration of the Cloud-in-Cell assignment scheme. Evertyhing is expressed in units of the cell size, Δx along the x direction and Δy along the y direction. In three dimensions, the particle is assigned to the eight neighboring cells with different weights given by equations (B.33)–(B.40).

As noted before the particle may contribute to densities in the parent cell $(x_{\rm c}, y_{\rm c}, z_{\rm c})$ and the seven neighboring cells. Let us define

$$ii = \mathrm{mod}(i+1, N_{\rm g0}); \quad jj = \mathrm{mod}(j+1, N_{\rm g1}); \quad kk = \mathrm{mod}(k+1, N_{\rm g2}). \tag{B.30}$$

The modulo function enforces periodic boundary conditions. The particle contributes to the eight cells indexed by (i,j,k), (ii,j,k), (i,jj,k), (i,j,kk), (ii,jj,k), (ii,j,kk), (i,jj,kk) and (ii,jj,kk). Let us define

$$d_x = \frac{x_{\rm p}-x_{\rm c}}{\Delta x} = \frac{x_{\rm p}}{\Delta x} - i; \quad d_y = \frac{y_{\rm p}-y_{\rm c}}{\Delta y} = \frac{y_{\rm p}}{\Delta y} - j; \quad d_z = \frac{z_{\rm p}-z_{\rm c}}{\Delta z} = \frac{z_{\rm p}}{\Delta z} - k; \tag{B.31}$$

$$t_x = 1 - d_x; \quad t_y = 1 - d_y; \quad t_z = 1 - d_z. \tag{B.32}$$

Contributions to the eight cells are given by the formulae below, which also correspond to linear interpolations in 3D:

$$W\big(\mathbf{x}_{\rm p} - \mathbf{x}_{(i,j,k)}\big) = t_x t_y t_z, \tag{B.33}$$
$$W\big(\mathbf{x}_{\rm p} - \mathbf{x}_{(ii,j,k)}\big) = d_x t_y t_z, \tag{B.34}$$
$$W\big(\mathbf{x}_{\rm p} - \mathbf{x}_{(i,jj,k)}\big) = t_x d_y t_z, \tag{B.35}$$
$$W\big(\mathbf{x}_{\rm p} - \mathbf{x}_{(i,j,kk)}\big) = t_x t_y d_z, \tag{B.36}$$
$$W\big(\mathbf{x}_{\rm p} - \mathbf{x}_{(ii,jj,k)}\big) = d_x d_y t_z, \tag{B.37}$$
$$W\big(\mathbf{x}_{\rm p} - \mathbf{x}_{(ii,j,kk)}\big) = d_x t_y d_z, \tag{B.38}$$
$$W\big(\mathbf{x}_{\rm p} - \mathbf{x}_{(i,jj,kk)}\big) = t_x d_y d_z, \tag{B.39}$$
$$W\big(\mathbf{x}_{\rm p} - \mathbf{x}_{(ii,jj,kk)}\big) = d_x d_y d_z. \tag{B.40}$$

Summing over all particles will result in the calculation of any quantity A on the grid (equation (B.21)), in particular the density contrast (equation (B.23)).

In figure B.2, we illustrate the CiC scheme in two dimensions. The first step is to identify the cell indexes i, ii, j, jj, k, kk. Then, one computes the weight coefficients $d_x, t_x, d_y, t_y, d_z, t_z$ as shown on the figure, and assigns the particle to the neighboring cells using the formulae above.

B.3.4 Interpolation

Interpolation is used to distribute a grid-wise quantity to particles. For example, for PM codes, accelerations are computed on the grid (see section B.4), then interpolated back to each particle's position.

Using the same notations as before, for some quantity A, the problem is to compute A_p given the values of $A(\mathbf{x}_\mathrm{c})$ in all the cells. This can be written in a similar fashion as equation (B.21), but summing on grid cells instead of particles:

$$A_\mathrm{p} = A(\mathbf{x}_\mathrm{p}) = \sum_{\{\mathbf{x}_\mathrm{c}\}} A(\mathbf{x}_\mathrm{c}) W(\mathbf{x}_\mathrm{p} - \mathbf{x}_\mathrm{c}). \tag{B.41}$$

W is the assignment function defined in section B.3.1, which involves a shape function S as previously (NGP, CiC, TSC, etc.). It is generally important to be consistent between the mesh assignment scheme and the interpolation scheme. In particular, for PM codes, the same prescription should be used for density assignment and for interpolating accelerations at particles' positions. This ensures the absence of artificial self-forces (forces exerted by a particle on itself) and momentum conservation (Hockney & Eastwood, 1981).

For the NGP scheme, the value of A_p for a particle is just the value of $A(\mathbf{x}_\mathrm{c})$ in its parent cell (i, j, k). For the CiC scheme, using equations (B.41) and (B.33)–(B.40), we find:

$$\begin{aligned} A_\mathrm{p} = \quad & A_{(i,j,k)} t_x t_y t_z + A_{(ii,j,k)} d_x t_y t_z + A_{(i,jj,k)} t_x d_y t_z + A_{(i,j,kk)} t_x t_y d_z \\ & + A_{(ii,jj,k)} d_x d_y t_z + A_{(ii,j,kk)} d_x t_y d_z + A_{(i,jj,kk)} t_x d_y d_z + A_{(ii,jj,kk)} d_x d_y d_z. \end{aligned} \tag{B.42}$$

This is identical to trilinear interpolation.

B.4 Poisson equation and accelerations

After density assignment, several steps are done on the mesh in PM codes: solving the Poisson equation to get the reduced gravitational potential $\widetilde{\Phi} \equiv \Delta^{-1}\delta$ (section B.4.1), and then differentiating to get the reduced accelerations $\tilde{g} \equiv \nabla\left(\Delta^{-1}\delta\right)$ (section B.4.2).

B.4.1 Solving the Poisson equation

It is customary to solve the Poisson equation in Fourier space:

1. the configuration-space density contrast $\delta(\mathbf{x})$ is Fourier-transformed to get $\delta(\mathbf{k})$;
2. the reduced gravitational potential is estimated by solving the Poisson equation in Fourier space, $\widetilde{\Phi}(\mathbf{k}) = G(\mathbf{k})\delta(\mathbf{k})$, where $G(\mathbf{k})$ is a Green's function for the Laplacian, discussed below;
3. the reduced gravitational potential $\widetilde{\Phi}(\mathbf{k})$ is transformed back to real space to get $\widetilde{\Phi}(\mathbf{x})$.

As noted in section B.2.2, the same array can be used to store δ and $\widetilde{\Phi}$, by doing in-place Fourier transforms.

Fourier transforms. Steps 1 and 3 involve forward and backward discrete Fourier transforms. In the codes implemented for this thesis, we use the Fast Fourier Transform approach for discrete data, provided by the FFTW software library,[7] defined and normalized as follows, for the forward and backward operations respectively:

$$\begin{aligned} \hat{f}_{\ell,m,n} &= \Delta x \Delta y \Delta z \sum_{i=0}^{N_{\mathrm{g}0}-1} \sum_{j=0}^{N_{\mathrm{g}1}-1} \sum_{k=0}^{N_{\mathrm{g}2}-1} f_{i,j,k}\, \mathrm{e}^{-2\mathrm{i}\pi(i\ell+jm+kn)/N_\mathrm{g}}, \\ f_{i,j,k} &= \frac{1}{L_0 L_1 L_2} \sum_{i=0}^{N_{\mathrm{g}0}-1} \sum_{j=0}^{N_{\mathrm{g}1}-1} \sum_{k=0}^{N_{\mathrm{g}2}-1} \hat{f}_{\ell,m,n}\, \mathrm{e}^{2\mathrm{i}\pi(i\ell+jm+kn)/N_\mathrm{g}}. \end{aligned}$$

In the following, we note the components of a Fourier mode $\mathbf{k}$ as $k_x = \frac{2\pi}{L_0}\ell$, $k_y = \frac{2\pi}{L_1}m$, $k_z = \frac{2\pi}{L_2}n$.

[7] http://www.fftw.org/

Green's function. The choice for the Green's function $G(\mathbf{k})$ depends on how one wants to represent to Laplacian in configuration space. In Fourier space, the reduced potential obeys $-k^2\widetilde{\Phi}(\mathbf{k}) \equiv \delta(\mathbf{k})$ where $k^2 \equiv |\mathbf{k}|^2 = k_x^2 + k_y^2 + k_z^2$. It is therefore natural to simply use as Green's function for the Laplacian $G(\mathbf{k}) = -1/k^2$. This is the choice adopted in GADGET-2 (Springel, Yoshida & White, 2001; Springel, 2005) and in the codes used in this thesis. Care should be taken however, as this choice corresponds to a highly non-local function in configuration space (see e.g. the discussion in Birdsall & Langdon, 1985, appendix E). Alternatively, we can discretize the Laplacian operator using the so-called 7-point template,

$$(\Delta\Phi)_{i,j,k} = \Phi_{i-1,j,k} + \Phi_{i+1,j,k} + \Phi_{i,j-1,k} + \Phi_{i,j+1,k} + \Phi_{i,j,k-1} + \Phi_{i,j,k+1} - 6\Phi_{i,j,k}, \tag{B.43}$$

for which the Green's function is given by

$$G(\mathbf{k}) = -\frac{1}{4}\left[\sin^2\left(\frac{k_x\Delta x}{2}\right) + \sin^2\left(\frac{k_y\Delta y}{2}\right) + \sin^2\left(\frac{k_z\Delta z}{2}\right)\right]^{-1}. \tag{B.44}$$

Force smoothing. Due to the finite resolution of the PM grid, short-range forces cannot be accurately resolved, which can cause spurious effects in simulations (Hockney & Eastwood, 1981). For this reason, we smooth the short-range forces by multiplying by a Gaussian kernel in Fourier space,

$$K_{k_\mathrm{s}}(k) = \exp\left(-\frac{1}{2}\frac{k^2}{k_\mathrm{s}^2}\right), \quad \text{where} \quad k_\mathrm{s} \equiv \frac{2\pi}{L}A_\mathrm{s}. \tag{B.45}$$

A_s is a free parameter that defines the split between long-range and short-range forces, in units of mesh cells. In our codes, we adopted $A_\mathrm{s} = 1.25$, the default value used in GADGET-2.

Deconvolution of the CiC kernel. We also correct for the convolution with the CiC kernel, by dividing twice by (see section B.3.3)

$$K_\mathrm{CiC}(\mathbf{k}) = \mathrm{sinc}^2\left(\frac{k_x}{k_{\mathrm{Nyq},x}}\right)\mathrm{sinc}^2\left(\frac{k_y}{k_{\mathrm{Nyq},y}}\right)\mathrm{sinc}^2\left(\frac{k_z}{k_{\mathrm{Nyq},z}}\right). \tag{B.46}$$

One deconvolution corrects for the smoothing effect of the CiC in the density assignment, the other for the force interpolation (Springel, 2005).

Overall factor in Fourier space. Summing up our discussions in this section, the overall factor that we apply to δ in Fourier space (that we still note $G(\mathbf{k})$ for convenience) is

$$G(\mathbf{k}) = -\frac{1}{k^2} \times \frac{K_{k_\mathrm{s}}(k)}{K_\mathrm{CiC}(\mathbf{k})^2}. \tag{B.47}$$

After performing an inverse Fourier transform, we obtain the reduced gravitational potential on the mesh.

B.4.2 Computation of the accelerations

We get the reduced accelerations on the mesh by finite differencing the reduced potential. It would also be possible to take the gradient in Fourier space, by multiplying the potential by a factor $-\mathrm{i}\mathbf{k}$ and obtaining directly the accelerations. However, this would require an inverse Fourier transform for each coordinate (i.e. three instead of one), with little gain in accuracy compared to finite differences (Springel, 2005).

We adopt central finite differences. Several schemes are possible depending on the desired accuracy. The two-point finite difference approximation (FDA2) is

$$\tilde{g}x_{(i,j,k)} \equiv \left.\frac{\partial\widetilde{\Phi}}{\partial x}\right|_{(i,j,k)} \approx \frac{1}{\Delta x}\left[\frac{1}{2}\widetilde{\Phi}_{(i+1,j,k)} - \frac{1}{2}\widetilde{\Phi}_{(i-1,j,k)}\right] \tag{B.48}$$

and similar formulae for the other coordinates $\tilde{g}y$ and $\tilde{g}z$. The accuracy is of order $\mathcal{O}(\Delta x^2)$.

In the codes implemented for this thesis, we adopted the four-point finite difference approximation (FDA4), as in GADGET-2,

$$\tilde{g}x_{(i,j,k)} \equiv \left.\frac{\partial\widetilde{\Phi}}{\partial x}\right|_{(i,j,k)} \approx \frac{1}{\Delta x}\left[\frac{2}{3}\left(\widetilde{\Phi}_{(i+1,j,k)} - \widetilde{\Phi}_{(i-1,j,k)}\right) - \frac{1}{12}\left(\widetilde{\Phi}_{(i+2,j,k)} - \widetilde{\Phi}_{(i-2,j,k)}\right)\right] \tag{B.49}$$

which offers order $\mathcal{O}(\Delta x^4)$ accuracy. In the two equations above, periodic boundary conditions should always be enforced: $i+1$ is actually $\mathrm{mod}(i+1, N_{\mathrm{g}0})$, etc.

After having computed the three components of the accelerations on the grid, $\tilde{g}x(\mathbf{x}_{\mathrm{c}})$, $\tilde{g}y(\mathbf{x}_{\mathrm{c}})$, $\tilde{g}z(\mathbf{x}_{\mathrm{c}})$, we interpolate with the CiC scheme (see section B.3.4) to get the accelerations at particles' positions, $\tilde{g}x(\mathbf{x}_{\mathrm{p}})$, $\tilde{g}y(\mathbf{x}_{\mathrm{p}})$, $\tilde{g}z(\mathbf{x}_{\mathrm{p}})$.

B.5 Update of positions and momenta

Now that we have the accelerations for each particle from the grid-based Poisson solver (step 1 in section B.2.1), we are able to update their momenta ("kick") and their positions ("drift"). This corresponds to steps 2 and 3 in section B.2.1. At this point, we have to adopt a time integration scheme to update positions and momenta from a_i to a_f, and to define Kick and Drift operators. This is the object of sections B.5.1 and B.5.2, respectively.

B.5.1 Time integrators

Let us consider a Hamiltonian system, described in phase space by the canonical coordinates $\mathbf{z} = (q, p)$ and the Hamiltonian $\mathcal{H}(p, q) \equiv p^2/2 + \Phi(q)$. If we call $f(\mathbf{z}) = (p, -\partial\Phi/\partial q)$, then Hamilton's equations simply read $\dot{\mathbf{z}} = f(\mathbf{z})$. Hamilton's equations are a symplectic map, which means that the energy and the volume in phase space are time-invariants:

$$\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}t} = 0 \quad \text{and} \quad \nabla \cdot f = 0. \tag{B.50}$$

It is generally important to adopt a numerical integrator that respects these two conditions, at least approximately (see also the discussion in section 3.4.3). For a map $\mathbf{z}(t) = \mathcal{F}(\mathbf{z}_0)$, the volume in phase space is conserved if $\det \dfrac{\partial \mathcal{F}}{\partial \mathbf{z}} = 1$. Classical first order time integrators use Euler's method. In particular, the explicit Euler method,

$$\mathbf{z}_{n+1} = \mathbf{z}_n + f(\mathbf{z}_n)\Delta t; \quad \text{for which} \quad \det \frac{\partial \mathcal{F}}{\partial \mathbf{z}} = 1 + \Delta t^2 \frac{\partial^2 \Phi}{\partial q^2}, \tag{B.51}$$

and the implicit Euler method,

$$\mathbf{z}_{n+1} = \mathbf{z}_n + f(\mathbf{z}_{n+1})\Delta t; \quad \text{for which} \quad \det \frac{\partial \mathcal{F}}{\partial \mathbf{z}} = \frac{1}{1 + \Delta t^2 \dfrac{\partial^2 \Phi}{\partial q^2}}, \tag{B.52}$$

are only approximately symplectic. Using the particles' positions at time t_n and momenta at time t_{n+1} makes the Euler integrator symplectic:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + f(q_n, p_{n+1})\Delta t; \quad \det \frac{\partial \mathcal{F}}{\partial \mathbf{z}} = 1. \tag{B.53}$$

For this thesis, we adopted the second-order symplectic "kick-drift-kick" algorithm, also known as the leapfrog scheme (e.g. Birdsall & Langdon, 1985, see also section 4.3.4):

$$p_{n+1/2} = p_n - \left.\frac{\partial \Phi}{\partial q}\right|_n \frac{\Delta t}{2}, \tag{B.54}$$

$$q_{n+1} = q_n + p_{n+1/2}\,\Delta t, \tag{B.55}$$

$$p_{n+1} = p_{n+1/2} - \left.\frac{\partial \Phi}{\partial q}\right|_{n+1} \frac{\Delta t}{2}. \tag{B.56}$$

It is a straightforward exercise to check that this scheme exactly preserves volume in phase space.

For PM and COLA codes, we assume a constant integration step $\Delta a \equiv \frac{a_f - a_i}{n}$, in such a way that the initial scale factor is $a_i = a_0$ and the final scale factor is $a_f = a_{n+1} = a_i + n\Delta a$. A schematic view of the leapfrog integration scheme is show in figure B.3. Note that during the evolution, positions and momenta are not synchronized but displaced by half a timestep. For this reason during the first timestep, we give the particles only "half a kick" using the accelerations computed at a_i; and during the last timestep, we give the particles an additional "half a kick", to synchronize momenta with positions at a_f.

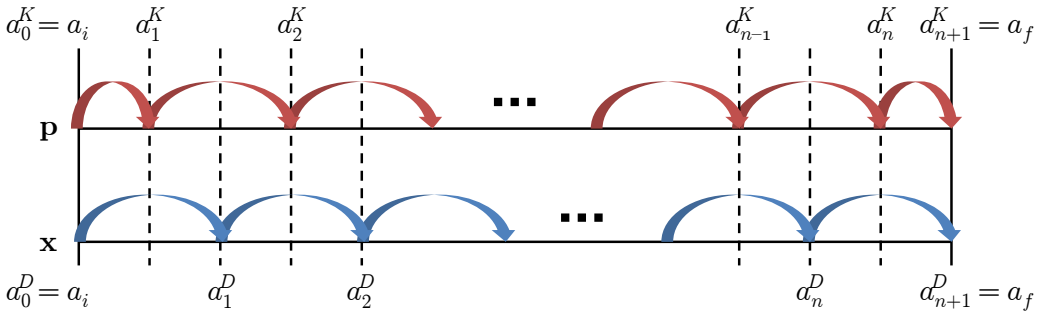

Figure B.3: Schematic illustration of the leapfrog integrator. Particles' momenta and positions are updated in turn, given the value of the other variable within the time interval.

B.5.2 Kick and Drift operators

In equations (B.7) and (B.8), all the explicit dependence on the scale factor is in the prefactors $\mathcal{D}(a)$ and $\mathcal{K}(a)$. The leapfrog scheme algorithm relies on integrating the equations on a small timestep and approximating the momenta or accelerations in the integrands by their value at some time within the interval. More precisely, for the "drift equation":

$$\mathbf{x}\big(a_f^{\mathrm{D}}\big) - \mathbf{x}\big(a_i^{\mathrm{D}}\big) = \int_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}} \mathcal{D}(\tilde{a})\mathbf{p}(\tilde{a})\,\mathrm{d}\tilde{a} \approx \left(\int_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}} \mathcal{D}(\tilde{a})\,\mathrm{d}\tilde{a}\right)\mathbf{p}\big(a^{\mathrm{K}}\big) \tag{B.57}$$

and similarly for the "kick equation":

$$\mathbf{p}\big(a_f^{\mathrm{K}}\big) - \mathbf{p}\big(a_i^{\mathrm{K}}\big) = \int_{a_i^{\mathrm{K}}}^{a_f^{\mathrm{K}}} \mathcal{K}(\tilde{a})\left[\nabla\left(\Delta^{-1}\delta\right)\right](\tilde{a})\,\mathrm{d}\tilde{a} \approx \left(\int_{a_i^{\mathrm{K}}}^{a_f^{\mathrm{K}}} \mathcal{K}(\tilde{a})\,\mathrm{d}\tilde{a}\right)\left[\nabla\left(\Delta^{-1}\delta\right)\right](a^{\mathrm{D}}) \tag{B.58}$$

This defines the Drift (D) and Kick (K) operators:

$$\mathrm{D}(a_i^{\mathrm{D}}, a_f^{\mathrm{D}}, a^{\mathrm{K}}): \quad \mathbf{x}(a_i^{\mathrm{D}}) \mapsto \mathbf{x}(a_f^{\mathrm{D}}) = \mathbf{x}(a_i^{\mathrm{D}}) + \left(\int_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}} \mathcal{D}(\tilde{a})\,\mathrm{d}\tilde{a}\right)\mathbf{p}\big(a^{\mathrm{K}}\big) \tag{B.59}$$

$$\mathrm{K}(a_i^{\mathrm{K}}, a_f^{\mathrm{K}}, a^{\mathrm{D}}): \quad \mathbf{p}(a_i^{\mathrm{D}}) \mapsto \mathbf{p}(a_f^{\mathrm{D}}) = \mathbf{p}(a_i^{\mathrm{D}}) + \left(\int_{a_i^{\mathrm{K}}}^{a_f^{\mathrm{K}}} \mathcal{K}(\tilde{a})\,\mathrm{d}\tilde{a}\right)\left[\nabla\left(\Delta^{-1}\delta\right)\right](a^{\mathrm{D}}) \tag{B.60}$$

Consistently with the scheme described in section B.5.1, the time evolution between a_0 and a_{n+1} is then achieved by applying the following operator, $\mathrm{E}(a_{n+1}, a_0)$, to the initial state $(\mathbf{x}(a_0), \mathbf{p}(a_0))$:

$$\mathrm{K}(a_{n+1/2}, a_{n+1}, a_{n+1})\mathrm{D}(a_n, a_{n+1}, a_{n+1/2})\left[\prod_{i=0}^{n}\mathrm{K}(a_{i+1/2}, a_{i+3/2}, a_{i+1})\mathrm{D}(a_i, a_{i+1}, a_{i+1/2})\right]\mathrm{K}(a_0, a_{1/2}, a_0). \tag{B.61}$$

If the COLA scheme is adopted, we obtain in a similar manner, from equations (B.16) and (B.17):

$$\begin{aligned}
\mathbf{x}\big(a_f^{\mathrm{D}}\big) - \mathbf{x}\big(a_i^{\mathrm{D}}\big) &\approx \left(\int_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}} \mathcal{D}(\tilde{a})\,\mathrm{d}\tilde{a}\right)\mathbf{p}_{\mathrm{MC}}\big(a^{\mathrm{K}}\big) - \left(\int_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}} \frac{\mathrm{d}D_1(\tilde{a})}{\mathrm{d}\tilde{a}}\,\mathrm{d}\tilde{a}\right)\boldsymbol{\Psi}_1 + \left(\int_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}} \frac{\mathrm{d}D_2(\tilde{a})}{\mathrm{d}\tilde{a}}\,\mathrm{d}\tilde{a}\right)\boldsymbol{\Psi}_2, \\
&= \left(\int_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}} \mathcal{D}(\tilde{a})\,\mathrm{d}\tilde{a}\right)\mathbf{p}_{\mathrm{MC}}\big(a^{\mathrm{K}}\big) - [D_1]_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}}\boldsymbol{\Psi}_1 + [D_2]_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}}\boldsymbol{\Psi}_2, \qquad \text{(B.62)} \\
\mathbf{p}_{\mathrm{MC}}\big(a_f^{\mathrm{K}}\big) - \mathbf{p}_{\mathrm{MC}}\big(a_i^{\mathrm{K}}\big) &\approx \left(\int_{a_i^{\mathrm{K}}}^{a_f^{\mathrm{K}}} \mathcal{K}(\tilde{a})\,\mathrm{d}\tilde{a}\right)\left(\left[\nabla\left(\Delta^{-1}\delta\right)\right](a^{\mathrm{D}}) - \mathcal{V}[D_1](a^{\mathrm{D}})\boldsymbol{\Psi}_1 - \mathcal{V}[D_2](a^{\mathrm{D}})\boldsymbol{\Psi}_2\right) \\
&= \left(\int_{a_i^{\mathrm{K}}}^{a_f^{\mathrm{K}}} \mathcal{K}(\tilde{a})\,\mathrm{d}\tilde{a}\right)\left(\left[\nabla\left(\Delta^{-1}\delta\right)\right](a^{\mathrm{D}}) - D_1(a^{\mathrm{D}})\boldsymbol{\Psi}_1 + \left(D_2(a^{\mathrm{D}}) - D_1^2(a^{\mathrm{D}})\right)\boldsymbol{\Psi}_2\right). \qquad \text{(B.63)}
\end{aligned}$$

In the last line we used equations (B.14) and (B.15). This defines new Drift ($\widetilde{\mathrm{D}}$) and Kick ($\widetilde{\mathrm{K}}$) operators:

$$\widetilde{\mathrm{D}}(a_i^{\mathrm{D}}, a_f^{\mathrm{D}}, a^{\mathrm{K}}): \quad \mathbf{x}(a_i^{\mathrm{D}}) \mapsto \mathbf{x}(a_f^{\mathrm{D}}) = \mathbf{x}(a_i^{\mathrm{D}}) + \left(\int_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}} \mathcal{D}(\tilde{a})\,\mathrm{d}\tilde{a}\right) \mathbf{p}_{\mathrm{MC}}(a^{\mathrm{K}}) - [D_1]_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}} \boldsymbol{\Psi}_1 + [D_2]_{a_i^{\mathrm{D}}}^{a_f^{\mathrm{D}}} \boldsymbol{\Psi}_2 \quad \text{(B.64)}$$

$$\widetilde{\mathrm{K}}(a_i^{\mathrm{K}}, a_f^{\mathrm{K}}, a^{\mathrm{D}}): \quad \mathbf{p}_{\mathrm{MC}}(a_i^{\mathrm{D}}) \mapsto \mathbf{p}_{\mathrm{MC}}(a_f^{\mathrm{D}}) = \mathbf{p}_{\mathrm{MC}}(a_i^{\mathrm{D}}) + \left(\int_{a_i^{\mathrm{K}}}^{a_f^{\mathrm{K}}} \mathcal{K}(\tilde{a})\,\mathrm{d}\tilde{a}\right) \times \\ \left(\left[\nabla\left(\Delta^{-1}\delta\right)\right](a^{\mathrm{D}}) - D_1(a^{\mathrm{D}})\boldsymbol{\Psi}_1 + \left(D_2(a^{\mathrm{D}}) - D_1^2(a^{\mathrm{D}})\right)\boldsymbol{\Psi}_2\right). \quad \text{(B.65)}$$

With COLA, the time evolution between a_0 and a_{n+1} is achieved by applying the following operator to the initial state $(\mathbf{x}(a_0), \mathbf{p}(a_0))$:

$$\mathrm{L}_+(a_{n+1})\widetilde{\mathrm{E}}(a_{n+1}, a_0)\mathrm{L}_-(a_0), \quad \text{(B.66)}$$

where $\widetilde{\mathrm{E}}(a_{n+1}, a_0)$ is the operator given by equation (B.61), replacing D by $\widetilde{\mathrm{D}}$ and K by $\widetilde{\mathrm{K}}$, and we where we use (see Tassev, Zaldarriaga & Eisenstein, 2013, appendix A):

$$\mathrm{L}_\pm(a): \quad \mathbf{p}(a) \mapsto \mathbf{p}(a) \pm \mathbf{p}_{\mathrm{LPT}}(a) = \mathbf{p}(a) \pm \frac{1}{\mathcal{D}(a)}\left(-\frac{\mathrm{d}D_1}{\mathrm{d}a}\boldsymbol{\Psi}_1 + \frac{\mathrm{d}D_2}{\mathrm{d}a}\boldsymbol{\Psi}_2\right). \quad \text{(B.67)}$$

L_- transforms the initial conditions to the rest frame of LPT observers (this is the same as initializing $\mathbf{p}_{\mathrm{MC}}$ to zero), and L_+ adds back the LPT momenta to $\mathbf{p}_{\mathrm{MC}}$ at the end.

In the codes implemented for this thesis, the integrals appearing in the Kick and Drift operators (equations (B.59), (B.60), (B.64), (B.65)) are explicitly computed numerically. Another approach for the discretization of time operators is proposed by Tassev, Zaldarriaga & Eisenstein (2013, section A.3.2.). When needed, the first order growth factor D_1 and its logarithmic derivative f_1 are also computed numerically by explicit integration. For the second-order growth factor and its logarithmic derivative, we use the fitting functions given by equations (1.119) and (1.138) (Bouchet *et al.*, 1995),

$$D_2(\tau) \approx -\frac{3}{7}D_1(\tau)\Omega_{\mathrm{m}}^{-1/143} \quad \text{and} \quad f_2(\tau) \approx 2f_1(\tau)^{54/55}. \quad \text{(B.68)}$$

B.6 Setting up initial conditions

The last missing part for a full cosmological pipeline including the PM/COLA codes described in previous sections is a way to set up initial conditions at $a = a_i$. The first step (section B.6.1) is to generate a realization of the random density field describing the early Universe. As argued in chapter 1, it is physically relevant to describe this field as a Gaussian random field.

The second step (section B.6.2) is to produce a high-redshift particle realization from this initial density field, to be given to the PM code. The common approach is to use Lagrangian perturbation theory (the ZA or 2LPT). Several existing codes perform this task: among others, GRAFIC (Bertschinger, 2001), N-GenIC (Springel, Yoshida & White, 2001; Springel, 2005, using the ZA) and its 2LPT extension, 2LPTIC (Crocce, Pueblas & Scoccimarro, 2006b; Pueblas & Scoccimarro, 2009), MPGRAFIC (Prunet *et al.*, 2008), MUSIC (Hahn & Abel, 2011). However, for the purpose of this thesis, we implemented an independent ZA/2LPT initial conditions generator. It is especially designed for full consistency with the BORG algorithm (see chapter 4); in particular, it uses the same routine as BORG for the generation of LPT displacement fields.

B.6.1 The initial Gaussian random field

There exists many software packages that allow generating normal random variates (i.e. single Gaussian random variates with mean 0 and variance 1), for example using the well-known Box-Müller method. We choose the routines provided by the GNU scientific library (Galassi *et al.*, 2003). We generate one such normal random variate in each cell of the initial grid, and call the resulting vector the "initial white noise field" ξ. It is a random signal with constant power spectrum ($\langle \xi\xi^\intercal = \mathbf{1}\rangle$). Alternatively, we can choose to import "constrained white noise" that comes, for example, of large-scale structure inferences performed with BORG.

Generally, using a vector of normal variates ξ, one can generate a realization of a grf with mean μ and covariance matrix C by simply taking any matrix $\sqrt{C}$ that satisfies $\sqrt{C}\sqrt{C}^\intercal = C$ and computing $x = \sqrt{C}\xi + \mu$.

One general way to generate $\sqrt{C}$ under the condition that C has only positive definite eigenvalues is to use the so-called Cholesky decomposition, implemented in many numerical packages.

For cosmological initial conditions, however, the problem is generally much simpler. As we are generating a random realization of the density contrast δ, the mean is $\mu = 0$ and, from statistical homogeneity and isotropy, the covariance matrix C should be diagonal in Fourier space and contain the power spectrum coefficients $P(k)/(2\pi)^{3/2}$ (see section 1.2.4.1). Hence, an obvious choice for the matrix $\sqrt{C}$ is the diagonal matrix containing the coefficients $\sqrt{P(k)/(2\pi)^{3/2}}$. Therefore, the procedure is to Fourier-transform ξ, to multiply each of its Fourier modes of norm k by $\sqrt{P(k)/(2\pi)^{3/2}}$, and to perform an inverse Fourier transform to get δ in configuration space.

Physical assumptions are needed for the power spectrum coefficients $P(k)$. One possible approach is to use the outputs of Boltzmann codes that describe the early Universe (e.g. CMBFAST – Seljak & Zaldarriaga, 1996, CAMB – Lewis & Challinor, 2002, or CLASS – Lesgourgues, 2011; Blas, Lesgourgues & Tram, 2011). However, in our implementation, we choose (as in BORG) to use the analytical power spectrum from Eisenstein & Hu (1998, 1999) for the baryon-CDM fluid (including baryonic wiggles). It depends on the following cosmological parameters, which have to be specified: Ω_Λ, Ω_m, Ω_b, n_s and σ_8.

When performing constrained simulations (see section 7.1.3), all the steps described in this section are bypassed, and we directly make use of the initial density contrast field inferred with BORG.

B.6.2 The high-redshift particle realization

We start from "grid-like" initial conditions, i.e. a realization of N_p dark matter particles, placed on a regular lattice. More precisely, for $0 \leq i < N_{\mathrm{p}0}$, $0 \leq j < N_{\mathrm{p}1}$, $0 \leq \mathit{k} < N_{\mathrm{p}2}$, we place a particle at Lagrangian coordinates $\mathbf{q} = (iL_0/N_{\mathrm{p}0}, jL_1/N_{\mathrm{p}1}, \mathit{k}L_2/N_{\mathrm{p}2})$. All the masses are set to the constant value given in footnote 1, and at this point all the velocities are zero. Finally, each particle's id is set to $\mathtt{mp} = \mathit{k} + N_{\mathrm{p}2} \times \big(j + N_{\mathrm{p}1} \times i\big)$. This allows to keep a memory of the initial position of particles at any later time, even in the PM code.

The following step is to compute the ZA and 2LPT displacements for each particle, given the initial density contrast field $\delta(\mathbf{q})$ generated in section B.6.1. We proceed as follows. The first-order potential field, $\phi^{(1)}(\mathbf{q})$, is evaluated on the Lagrangian grid by solving equation (1.134) in Fourier space,[8]

$$\phi^{(1)}(\boldsymbol{\kappa}) = -\delta(\boldsymbol{\kappa})/\kappa^2. \tag{B.69}$$

Each of its second order derivatives are also evaluated in Fourier space, using

$$\phi^{(1)}_{,ab}(\boldsymbol{\kappa}) = -\phi^{(1)}(\boldsymbol{\kappa})\boldsymbol{\kappa}_a \cdot \boldsymbol{\kappa}_b. \tag{B.70}$$

and inverse Fourier-transformed. From the configuration-space quantity

$$\boldsymbol{\phi}(\mathbf{q}) \equiv \phi^{(1)}_{,xx}(\mathbf{q})\phi^{(1)}_{,yy}(\mathbf{q}) + \phi^{(1)}_{,xx}(\mathbf{q})\phi^{(1)}_{,zz}(\mathbf{q}) + \phi^{(1)}_{,yy}(\mathbf{q})\phi^{(1)}_{,zz}(\mathbf{q}) - \phi^{(1)}_{,xy}(\mathbf{q})^2 - \phi^{(1)}_{,xz}(\mathbf{q})^2 - \phi^{(1)}_{,yz}(\mathbf{q})^2, \tag{B.71}$$

we compute the second-order potential field, $\phi^{(2)}(\mathbf{q})$, again in Fourier space, using (see equation (1.135))

$$\phi^{(2)}(\boldsymbol{\kappa}) = -\boldsymbol{\phi}(\boldsymbol{\kappa})/\kappa^2. \tag{B.72}$$

Once $\phi^{(1)}(\mathbf{q})$ and $\phi^{(2)}(\mathbf{q})$ are known, we evaluate the first and second order displacements $\boldsymbol{\Psi}^{(1)}(\mathbf{q}) \equiv \nabla_\mathbf{q}\phi^{(1)}(\mathbf{q})$ and $\boldsymbol{\Psi}^{(2)}(\mathbf{q}) \equiv \nabla_\mathbf{q}\phi^{(2)}(\mathbf{q})$ on the initial grid in configuration space, by using a finite difference approximation scheme at order 2 (see section B.4.2). Then, we interpolate from the grid to particles' Lagrangian positions using a CiC scheme (see section B.3.4).

Finally, particles are displaced from their Lagrangian positions and their velocities are modified as prescribed by LPT (equations (1.136) and (1.137)). More precisely, particles are given a zeroth "kick",

$$\mathrm{K}_0(a_i): \quad \mathbf{u} = 0 \mapsto \mathbf{u}(a_i) = -f_1(a_i)D_1(a_i)\mathcal{H}(a_i)\boldsymbol{\Psi}_1(\mathbf{q}) + f_2(a_i)D_2(a_i)\mathcal{H}(a_i)\boldsymbol{\Psi}_2(\mathbf{q}), \tag{B.73}$$

where $\mathbf{u} \equiv \mathrm{d}\mathbf{x}/\mathrm{d}\tau = a\mathcal{H}\mathrm{d}\mathbf{x}/\mathrm{d}a$. From this we deduce the initial momenta in code units,

$$\mathbf{p}(a_i) = \frac{1}{a_i\mathcal{H}(a_i)\mathcal{D}(a_i)}\mathbf{u}(a_i). \tag{B.74}$$

They also follow a zeroth "drift":

$$\mathrm{D}_0(a_i): \quad \mathbf{q} \mapsto \mathbf{x}(a_i) = \mathbf{q} - D_1(a_i)\boldsymbol{\Psi}_1(\mathbf{q}) + D_2(a_i)\boldsymbol{\Psi}_2(\mathbf{q}). \tag{B.75}$$

The required numerical prefactors are computed as described at the end of section B.5.2.

[8] We denote by $\boldsymbol{\kappa}$ a Fourier mode on the Lagrangian grid, κ its norm.