Differentiable Cosmological Simulation with Adjoint Method

Yin Li, Chirag Modi, Drew Jamieson, Yucheng Zhang, Libin Lu, Yu Feng, François Lanusse, Leslie Greengard

The Astrophysical Journal Supplement Series, 270:36 (13pp), 2024 February https://doi.org/10.3847/1538-4365/ad0ce7

Differentiable Cosmological Simulation with the Adjoint Method

Yin Li (李寅)[1,2,3], Chirag Modi[2,3], Drew Jamieson[4], Yucheng Zhang (张宇澄)[1,5], Libin Lu (陆利彬)[2], Yu Feng (冯雨)[6], François Lanusse[7], and Leslie Greengard[2,8]
[1] Department of Mathematics and Theory, Peng Cheng Laboratory, Shenzhen, Guangdong 518066, People's Republic of China; eelregit@gmail.com
[2] Center for Computational Mathematics, Flatiron Institute, New York, NY 10010, USA
[3] Center for Computational Astrophysics, Flatiron Institute, New York, NY 10010, USA
[4] Max Planck Institute for Astrophysics, 85748 Garching bei München, Germany
[5] Center for Cosmology and Particle Physics, Department of Physics, New York University, New York, NY 10003, USA
[6] Berkeley Center for Cosmological Physics, University of California, Berkeley, CA 94720, USA
[7] AIM, CEA, CNRS, Université Paris-Saclay, Université Paris Diderot, Sorbonne Paris Cité, F-91191 Gif-sur-Yvette, France
[8] Courant Institute, New York University, New York, NY 10012, USA
Received 2023 March 5; revised 2023 November 2; accepted 2023 November 6; published 2024 February 6

Abstract

Rapid advances in deep learning have brought not only a myriad of powerful neural networks, but also breakthroughs that benefit established scientific research. In particular, automatic differentiation (AD) tools and computational accelerators like GPUs have facilitated forward modeling of the Universe with differentiable simulations. Based on analytic or automatic backpropagation, current differentiable cosmological simulations are limited by memory, and thus are subject to a trade-off between time and space/mass resolution, usually sacrificing both. We present a new approach free of such constraints, using the adjoint method and reverse time integration. It enables larger and more accurate forward modeling at the field level, and will improve gradient-based optimization and inference. We implement it in an open-source particle-mesh (PM) *N*-body library `pmwd` (PM with derivatives). Based on the powerful AD system `JAX`, `pmwd` is fully differentiable, and is highly performant on GPUs.

Unified Astronomy Thesaurus concepts: Cosmology (343); Large-scale structure of the universe (902); N-body simulations (1083); Astronomy software (1855); Computational methods (1965); Algorithms (1883)

1. Introduction

Current established workflows of statistical inference from cosmological data sets involve reducing cleaned data to summary statistics, such as the power spectrum, and predicting these statistics using perturbation theories, semianalytic models, or simulation-calibrated emulators. These can be suboptimal due to the limited model fidelity and the risk of information loss in data compression. Cosmological simulations (Hockney & Eastwood 1988; Angulo & Hahn 2022) can accurately predict structure formation even in the nonlinear regime at the level of the fields. Using simulations as forward models also naturally accounts for the cross-correlation of different observables, and can easily incorporate systematic errors. This approach has been intractable due to the large computational costs of conventional CPU clusters, but rapid advances in accelerator technology such as GPUs open the possibility of simulation-based modeling and inference (Cranmer et al. 2020). Furthermore, model differentiability enabled by automatic differentiation (AD) libraries can accelerate parameter constraints with gradient-based optimization and inference. A differentiable field-level forward model combining these two features is able to constrain physical parameters together with the initial conditions of the Universe.

The first differentiable cosmological simulations, such as BORG, ELUCID, and BORG-PM (Jasche & Wandelt 2013; Wang et al. 2014; Jasche & Lavaux 2019), were developed before the advent of modern AD systems, and were based on the analytic derivatives, which involve first convolution derivation by hand using the chain rule (see, e.g., Seljak et al. 2017, Appendix D) before implementing them in code. Later codes including `FastPM` and `FlowPM` (Feng et al. 2016; Seljak et al. 2017; Modi et al. 2021) compute gradients using the AD engines, namely `vmad`[9] (written by the same authors) and `TensorFlow`, respectively. The AD frameworks automatically apply the chain rule to the primitive operations that comprise the whole simulation, relieving the burden of derivation and implementation of the derivatives. Both analytic differentiation and AD backpropagate the gradients through the whole history, which requires saving the states at all time steps in memory. Therefore, they are subject to a trade-off between time and space/mass resolution, usually sacrificing both. As a result, they lose accuracy on small scales and in dense regions where the time resolution is important, e.g., in weak lensing (Böhm et al. 2021).

Alternatively, the adjoint method provides systematic ways of deriving the gradients of an objective function $\mathcal{J}$ under constraints (Pontryagin 1962), such as those imposed by the *N*-body equations of motion in a simulated Universe. It identifies a set of *adjoint variables* $\boldsymbol{\lambda}$, dual to the state variables $\boldsymbol{z}$ of the model, and carrying the gradient information of the objective function with respect to the model state $\partial\mathcal{J}/\partial\boldsymbol{z}$. For time-dependent problems, the adjoint variables evolve *backward* in time by a set of equations dual to that of the forward evolution, known as the *adjoint equations*. For continuous time, the adjoint equations are a set of differential equations, while in the discrete case, they become difference equations which are practically a systematic way to structure the chain rule or backpropagation. Their initial conditions are set by

[9] https://github.com/rainwoodman/vmad

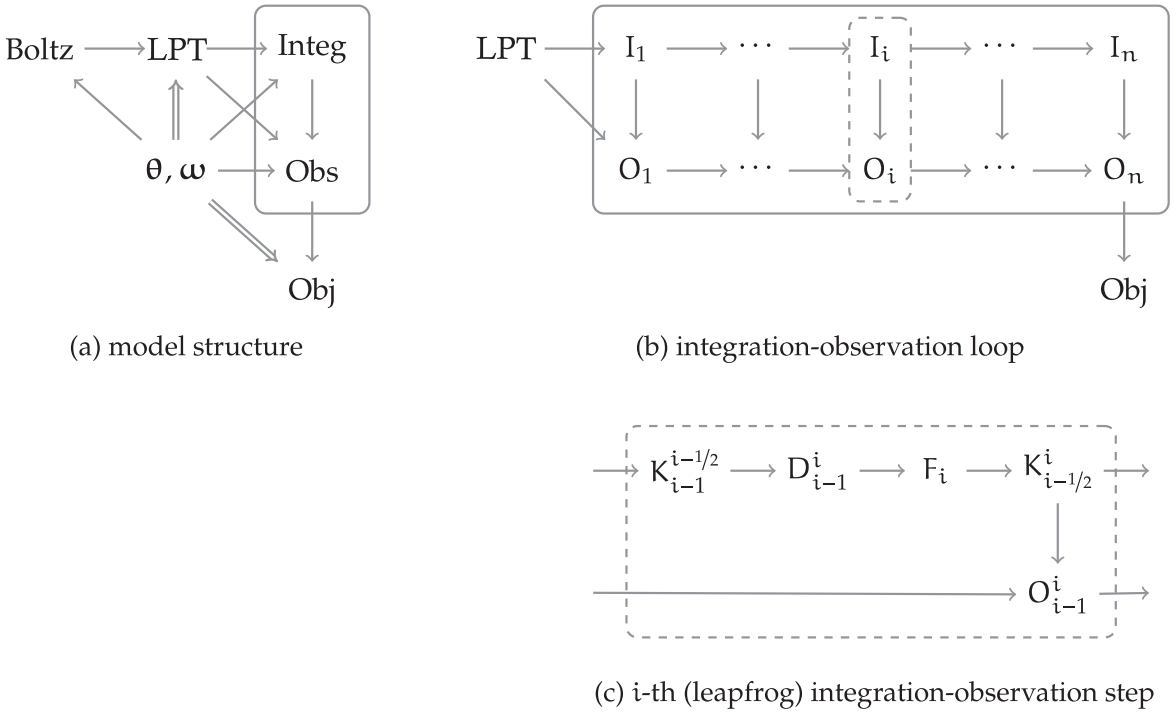


Figure 1. Simulation-based forward model of the Universe. (a) Shows the overall model structure. Single arrows from the cosmological parameters $\boldsymbol{\theta}$ and white noise modes $\boldsymbol{\omega}$ indicate dependence on $\boldsymbol{\theta}$ only, while double arrows imply dependence on both. The time integration loop in (b) expands the solid box in (a), and a single time step in (c) further expands the dashed box in (b). We describe different operators in Section 2: Boltzmann solver ("Boltz") and initial condition generator by the Lagrangian perturbation theory ("LPT") in Section 2.1; force solver (F) in Section 2.2; time integration ("Integ"), kick (K), and drift (D) in Section 2.3; observation ("Obs" and O) and objective ("Obj") in Section 2.4. Gradients flow backward with all arrows reversed (Section 3.6).

the explicit dependence of the objective on the simulation state, e.g., $\boldsymbol{\lambda}_n = \partial\mathcal{J}/\partial\boldsymbol{z}_n$ if $\mathcal{J}$ is a function of the final state $\boldsymbol{z}_n$. Solving the adjoint equations can help us propagate the gradient information via the adjoint variables to the input parameters $\boldsymbol{\theta}$, to compute the objective gradients $d\mathcal{J}/d\boldsymbol{\theta}$. And we will see later that the propagated and accumulated gradients on parameters come naturally from multiple origins, each reflecting a $\boldsymbol{\theta}$-dependence at one stage of the modeling in Figure 1(a).

The backward adjoint evolutions depend on the states in the forward run, which we can resimulate with reverse time integration if the dynamics are reversible, thereby dramatically reducing the memory cost (Chen et al. 2018). Furthermore, we derive the discrete adjoint equations dual to the discrete forward time integration, known as the discretize-then-optimize approach (e.g., Gholaminejad et al. 2019), to ensure gradients propagate backward along the *same* trajectory as taken by the forward time integration. This is in contrast with the optimize-then-discretize approach, which numerically integrates the continuous adjoint equations, and is prone to larger deviations between the forward and the backward trajectories, due to different discretizations (Lanzieri et al. 2022). In brief, to compute the gradient we only need to evolve a simulation forward, and then backward jointly with its dual adjoint equations. We introduce the adjoint method first for generic time-dependent problems in both the continuous and discrete cases in Section 3, and then present its application on cosmological simulation in Section 3.6.

We implement the adjoint method with reverse time integration in a new differentiable particle-mesh (PM) library pmwd using JAX (Li et al. 2022). pmwd is memory efficient at gradient computation with a space complexity independent of the number of time steps, and is computation efficient when running on GPUs.

2. Forward Simulation

We first review and formulate all components of the N-body simulation-based forward model of the cosmological structure formation.

2.1. Initial Conditions and Perturbation Theories

N-body particles discretize the uniform distribution of matter at the beginning of cosmic history (scale factor $a(t) \to 0$) at their Lagrangian positions $\boldsymbol{q}$, typically on a Cartesian grid, from which they then evolve by displacements $\boldsymbol{s}$ to their later positions, i.e.,

$$\boldsymbol{x} = \boldsymbol{q} + \boldsymbol{s}(\boldsymbol{q}). \tag{1}$$

To account for the cosmic background that expands globally, while $\boldsymbol{x}$ is the comoving position relative to this background, the physical position grows with the expansion scale factor, i.e., $\boldsymbol{ax}$. The expansion rate is described by the Hubble parameter $H \triangleq d \ln a/dt$.

The initial conditions of particles can be set perturbatively when $|\boldsymbol{\nabla} \cdot \boldsymbol{s}|$, the linear approximation of the density fluctuation, is much less than 1. We compute the initial displacements and momenta using the second-order Lagrangian perturbation theory (2LPT; Bouchet et al. 1995):

$$\begin{aligned} \boldsymbol{s} &= D_1\boldsymbol{s}^{(1)} + D_2\boldsymbol{s}^{(2)}, \\ \boldsymbol{p} &= a^2\dot{\boldsymbol{x}} = a^2\dot{\boldsymbol{s}} = a^2H(D_1'\boldsymbol{s}^{(1)} + D_2'\boldsymbol{s}^{(2)}), \end{aligned} \tag{2}$$

where $\boldsymbol{p}$ is the canonical momentum[10] for canonical coordinate $\boldsymbol{x}$. The temporal and spatial dependences separate at each order:

[10] We omit the particle mass m in the canonical momentum $\boldsymbol{p} = ma^2\dot{\boldsymbol{x}}$ throughout for brevity.

the ith order growth factor D_i is only a function of the scale factor a (or time t), and the ith order displacement field $\boldsymbol{s}^{(i)}$ depends only on $\boldsymbol{q}$. We have also used two types of time derivatives, $\dot{\square} \triangleq d\square/dt$ and $\square' \triangleq d\square/d\ln a$, related by $\dot{\square} = H\square'$.

Both the first- and second-order displacements are potential flows,

$$\boldsymbol{s}^{(1)} = -\boldsymbol{\nabla}_{\boldsymbol{q}}\phi_{\boldsymbol{s}}^{(1)}, \quad \boldsymbol{s}^{(2)} = -\boldsymbol{\nabla}_{\boldsymbol{q}}\phi_{\boldsymbol{s}}^{(2)}, \tag{3}$$

with the scalar potentials sourced by

$$\begin{aligned} \nabla_{\boldsymbol{q}}^2\phi_{\boldsymbol{s}}^{(1)} &= \delta^{(1)}(\boldsymbol{q}), \\ \nabla_{\boldsymbol{q}}^2\phi_{\boldsymbol{s}}^{(2)} &= \textstyle\sum_{i<j}(\phi_{\boldsymbol{s},ii}^{(1)}\phi_{\boldsymbol{s},jj}^{(1)} - \phi_{\boldsymbol{s},ij}^{(1)}\phi_{\boldsymbol{s},ji}^{(1)}), \end{aligned} \tag{4}$$

where $\phi_{\boldsymbol{s},ij} \triangleq \partial^2\phi_{\boldsymbol{s}}/\partial q_i\partial q_j$, and $\delta^{(1)}$ is the linear order of the overdensity field, δ, related to the density field ρ and mean matter density $\bar{\rho}$ by $\delta \triangleq \rho/\bar{\rho} - 1$.

The linear overdensity $\delta^{(1)}$, which sources the 2LPT particle initial conditions, is a homogeneous and isotropic Gaussian random field in the consensus cosmology, with its Fourier transform $\delta^{(1)}(\boldsymbol{k}) = \int d\boldsymbol{q}\delta^{(1)}(\boldsymbol{q})e^{-i\boldsymbol{k}\cdot\boldsymbol{q}}$ characterized by the linear matter power spectrum $P_{\rm lin}$,

$$\begin{aligned} \langle\delta^{(1)}(\boldsymbol{k})\delta^{(1)}(\boldsymbol{k}')\rangle &= (2\pi)^3\delta^{\rm D}(\boldsymbol{k}+\boldsymbol{k}')P_{\rm lin}(k) \\ &\simeq \delta^{\rm K}(\boldsymbol{k}+\boldsymbol{k}')VP_{\rm lin}(k). \end{aligned} \tag{5}$$

The angle bracket takes the ensemble average of all possible realizations. Homogeneity demands that different wavevectors are uncorrelated, thus the Dirac delta $\delta^{\rm D}$ in the first equality. And with isotropy, $P_{\rm lin}$ does not depend on the direction of the wavevector $\boldsymbol{k}$, but only on its magnitude, the wavenumber $k \triangleq |\boldsymbol{k}|$. In a periodic box of volume V, where $\boldsymbol{k}$ is discrete, $\delta^{\rm D}$ is replaced by the Kronecker delta $\delta^{\rm K}$ in the second equality. Numerically, we can easily generate a $\delta^{(1)}$ realization by sampling each Fourier mode independently,

$$\delta^{(1)}(\boldsymbol{k}) = \sqrt{VP_{\rm lin}(k)}\ \omega(\boldsymbol{k}), \tag{6}$$

with $\omega(\boldsymbol{k})$ being any Hermitian white noise, i.e., Fourier transform of a real white noise field $\omega(\boldsymbol{q})$.

Cosmological perturbation theory gives the linear power spectrum as[11]

$$\frac{k^3}{2\pi^2}P_{\rm lin}(k) = \frac{4}{25}A_{\rm s}\left(\frac{k}{k_{\rm pivot}}\right)^{n_{\rm s}-1}T^2(k)\frac{c^4k^4}{\Omega_m^2H_0^4}, \tag{7}$$

where the shape of $P_{\rm lin}$ is determined by the transfer function T, solution to the linearized Einstein–Boltzmann equations (Lewis & Challinor 2011; Blas et al. 2011). T depends on the cosmological parameters

$$\boldsymbol{\theta} = (A_{\rm s}, n_{\rm s}, \Omega_m, \Omega_b, h, \cdots),$$

some of which already appear in Equation (7): $A_{\rm s}$ is the amplitude of the primordial power spectrum defined at some fixed scale $k_{\rm pivot}$; $n_{\rm s}$ describes the shape of the primordial power spectrum; Ω_m is the total matter density parameter; Ω_b is the baryonic matter density parameter; and H_0 is the Hubble constant, often parameterized by the dimensionless h as $H_0 = 100\,h\,{\rm km\,s^{-1}\,Mpc^{-1}}$. Other parameters may enter in extensions of the standard Λ cold dark matter (ΛCDM) cosmology.

In summary, other than the discretized white noise modes $\boldsymbol{\omega}$, to generate initial conditions we need the growth functions D and the transfer function T, both of which depend on the cosmological parameters $\boldsymbol{\theta}$. We compute D by solving the ordinary differential equations (ODEs) given in Appendix, and employ the fitting formula for T from Eisenstein & Hu (1998). We illustrate these dependencies in the upper left triangle of Figure 1(a).

At early times and/or lower space/mass resolution, LPT can be accurate enough to directly compare to the observational data. However, a more expensive integration of the N-body dynamics is necessary in the nonlinear regime. During LPT and the time integration we can *observe* the simulation predictions by interpolating on the past light cone of a chosen observer. These form the upper right square of Figure 1(a).

2.2. Force Evaluation

The core of gravitational N-body simulation is the gravity solver. The gravitational potential sourced by matter density fluctuation satisfies the Poisson equation

$$\nabla^2\phi(\boldsymbol{x}) = \frac{3}{2}\frac{\Omega_mH_0^2}{a}\delta(\boldsymbol{x}), \tag{8}$$

where ∇^2 is the Laplacian with respect to $\boldsymbol{x}$. We separate the time dependence by defining $\varphi \triangleq a\phi$, so φ satisfies

$$\nabla^2\varphi(\boldsymbol{x}) = \frac{3}{2}\Omega_mH_0^2\ \delta(\boldsymbol{x}), \tag{9}$$

which only depends on the matter overdensity δ.

While our adjoint method is general, we employ the PM solver in pmwd for efficiency, and leave the implementation of short-range forces to future development. With the PM method, we evaluate $\delta(\boldsymbol{x})$ on an auxiliary mesh by *scattering* particle masses to the nearest grid points. We use the usual cloud-in-cell, or trilinear, interpolation (Hockney & Eastwood 1988), to compute the fractions of a particle at $\boldsymbol{x}'$ going to a grid point at $\boldsymbol{x}$,

$$W(\boldsymbol{x}, \boldsymbol{x}') = \prod_{i=1}^{3}\ \max\left(1 - \frac{|x_i - x_i'|}{l}, 0\right), \tag{10}$$

where l is the mesh cell size.

The gravitational field, $-\boldsymbol{\nabla}\varphi$, can then be readily computed on the mesh with the fast Fourier transform (FFT), as the above partial differential equation becomes an algebraic one in Fourier space:

$$-k^2\varphi(\boldsymbol{k}) = \frac{3}{2}\Omega_mH_0^2\ \delta(\boldsymbol{k}). \tag{11}$$

In Fourier space, $-\boldsymbol{\nabla}\varphi(\boldsymbol{x})$ is just $-i\boldsymbol{k}\varphi(\boldsymbol{k})$, each component of which can be transformed back to obtain the force field. With four (one forward and three inverse) FFTs, we can obtain $-\boldsymbol{\nabla}\varphi(\boldsymbol{x})$ from $\delta(\boldsymbol{x})$, with both on the mesh, efficiently.

Finally, we interpolate the particle accelerations by *gathering* $-\boldsymbol{\nabla}\varphi$ from the same grid points with the same weights, as given in Equation (10).

[11] Here the time dependence of the linear power, $P_{\rm lin}(k, a) = P_{\rm lin}(k)D_1^2(a)$, has been left to Equation (2).

2.3. Time Integration

N-body particles move by the following equations of motion:

$$\dot{\boldsymbol{x}} = \frac{\boldsymbol{p}}{a^2},$$
$$\dot{\boldsymbol{p}} = -\boldsymbol{\nabla}\phi = -\frac{\boldsymbol{\nabla}\varphi}{a}. \tag{12}$$

We use the `FastPM` time stepping (Feng et al. 2016), designed to reproduce in the linear regime the linear Lagrangian perturbation theory, i.e., the 1LPT as the first order in Equation (2), also known as the Zel'dovich approximation (hereafter ZA). We present a simplified derivation below.

N-body simulations integrate Equation (12) in discrete steps (Figure 1(b)), typically with a symplectic integrator that updates $\boldsymbol{x}$ and $\boldsymbol{p}$ alternately. From t_a to t_b,

$$\begin{aligned}\boldsymbol{x}_b &= \boldsymbol{x}_a + \int_{t_a}^{t_b} \frac{\boldsymbol{p}}{a^2}\, dt \approx \boldsymbol{x}_a \\ &\quad + \frac{\boldsymbol{p}(t_c)}{G_D(t_c)} \int_{t_a}^{t_b} \frac{G_D}{a^2}\, dt \triangleq \boldsymbol{x}_a + \boldsymbol{p}_c\, D(t_c, t_a, t_b), \\ \boldsymbol{p}_b &= \boldsymbol{p}_a - \int_{t_a}^{t_b} \frac{\boldsymbol{\nabla}\varphi}{a}\, dt \approx \boldsymbol{p}_a \\ &\quad - \frac{\boldsymbol{\nabla}\varphi(t_c)}{G_K(t_c)} \int_{t_a}^{t_b} \frac{G_K}{a}\, dt \triangleq \boldsymbol{p}_a - \boldsymbol{\nabla}\varphi_c K(t_c, t_a, t_b),\end{aligned} \tag{13}$$

which are the drift and kick operators, respectively. In the second equalities of each equation, we have introduced two time-dependent functions G_D and G_K. As approximations, they have been taken out of the integrals together with $\boldsymbol{p}$ and $\boldsymbol{\nabla}\varphi$ at some intermediate representative time t_c. We can make the approximation more accurate by choosing G_D to have a time dependence closer to that of $\boldsymbol{p}$, likewise for $\boldsymbol{\nabla}\varphi$ and G_K. However, in most codes G_D and G_K are simply set to 1 (Quinn et al. 1997), lowering the accuracy when the number of time steps is limited.

`FastPM` chooses G_D and G_K according to the ZA growth history, thereby improving the accuracy on large scales and at early times. In ZA, the displacements are proportional to the linear growth factor, $\boldsymbol{s} \propto D_1$, which determines the time dependences of the momenta and the accelerations by (12). Therefore, we can set G_D and G_K in Equation (13) to

$$G_D \coloneqq a^2 \dot{D}_1,$$
$$G_K \coloneqq a \dot{G}_D. \tag{14}$$

These are functions of D_1 and its derivatives, as given by Equation (A7). With these choices, the drift and kick factors, defined in Equation (13), then become

$$D(t_c, t_a, t_b) = \frac{D_1(t_b) - D_1(t_a)}{G_D(t_c)},$$
$$K(t_c, t_a, t_b) = \frac{G_D(t_b) - G_D(t_a)}{G_K(t_c)}. \tag{15}$$

While these operators are generally applicable in any symplectic integrator, we use them in the second-order kick-drift-kick leapfrog, or velocity Verlet, integration scheme (Quinn et al. 1997). From t_{i-1} to t_i, the particles' state $(\boldsymbol{x}, \boldsymbol{p})$ is updated in the following order:

$$\begin{aligned} K_{i-1}^{i-1/2}&: \quad \boldsymbol{p}_{i-1/2} = \boldsymbol{p}_{i-1} + \boldsymbol{a}_{i-1} K(t_{i-1}, t_{i-1}, t_{i-1/2}), \\ D_{i-1}^{i}&: \quad \boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \boldsymbol{p}_{i-1/2}\, D(t_{i-1/2}, t_{i-1}, t_i), \\ F_i&: \quad \boldsymbol{a}_i \triangleq -\boldsymbol{\nabla}\varphi(\boldsymbol{x}_i), \\ K_{i-1/2}^{i}&: \quad \boldsymbol{p}_i = \boldsymbol{p}_{i-1/2} + \boldsymbol{a}_i K(t_i, t_{i-1/2}, t_i). \end{aligned} \tag{16}$$

The left column names the operators as shown in Figure 1(c). The force operator F on the third line computes the accelerations as described in Section 2.2. It caches the results in $\boldsymbol{a}$, so that they can be used again by the first K in the next step. Note that we need to initialize $\boldsymbol{a}_0$ with F_0 before the first time step.

2.4. Observation and Objective

Because all observables live on our past light cone, we model observations on the fly by interpolating the jth particle's state $\hat{\boldsymbol{z}}_j = (\hat{\boldsymbol{x}}_j, \hat{\boldsymbol{p}}_j)$ when they cross the light cone at $\hat{t}_j$. Given $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{p})$ at t_{i-1} and t_i, we can parameterize intermediate particle trajectories with cubic Hermite splines. Combined with the $\boldsymbol{\theta}$-dependent propagation of the light front, we can solve for the intersections, at which we can record the observed $\hat{\boldsymbol{z}}$. The solution can even be analytic if the light propagation is also approximated cubically. Note that we only *observe* the dark matter phase space here, and leave more realistic observables to future works, including the forward modeling of real observational effects. Figure 1(c) illustrates the observation operator O and its dependencies on the previous and the current time step.

We can compare the simulated observables to the observational data, either at the level of the fields or the summary statistics, by some objective function in case of optimization, or by a posterior probability for Bayesian inference. We refer to both cases by objective and denote it by $\mathcal{J}$ throughout. Note that in general, it can also depend on $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$, in the form of regularization or prior probability.

Formally, we can combine the observation and objective operators as

$$\mathcal{J}(\hat{\boldsymbol{z}}, \boldsymbol{\theta}, \boldsymbol{\omega}) = \mathcal{J}(\hat{\boldsymbol{z}}(z_0, \cdots, z_n, \boldsymbol{\theta}), \boldsymbol{\theta}, \boldsymbol{\omega}), \tag{17}$$

as illustrated in the lower right triangle in Figure 1(a). Note this form also captures the conventional simulation snapshots at the end of a time step or those interpolated between two consecutive steps, so we model all these cases as observations in `pmwd`.

3. Backward Differentiation—the Adjoint Method

We first introduce the adjoint method for generic time-dependent ODEs, and derive the adjoint equations following a pedagogical tutorial by Andrew Bradley.[12] We then adopt the discretize-then-optimize approach, and derive the discrete adjoint equations, which is more suitable for the N-body symplectic time integration. Finally, we apply them to derive the adjoint equations and the gradients for cosmological simulations described in Section 2, and couple them with the reverse time integration to reduce the space complexity.

[12] https://cs.stanford.edu/~ambrad/adjoint_tutorial.pdf

3.1. Variational (Tangent) and Adjoint Equations

Consider a vector state $\boldsymbol{z}(t)$ subject to the following ODEs and initial conditions

$$\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z}, t, \boldsymbol{\theta}), \qquad \boldsymbol{z}(t_0) = \boldsymbol{z}_0(\boldsymbol{\theta}), \tag{18}$$

for $t \in [t_0, t_1]$. Here the initial conditions can depend on the parameters $\boldsymbol{\theta}$.

A perturbation in the initial conditions propagates forward in time. For $\boldsymbol{z}(t, t_0, \boldsymbol{z}_0)$, the Jacobian of state variables describing this,

$$\boldsymbol{\Delta} = \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}_0}, \tag{19}$$

evolves from identity $\boldsymbol{\Delta}_0 = \boldsymbol{I}$ by

$$\dot{\boldsymbol{\Delta}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} \cdot \boldsymbol{\Delta}, \tag{20}$$

following from Equation (18). Equation (20) is known as the *variational* or *tangent equation*.

The backward version of $\boldsymbol{\Delta}$,

$$\boldsymbol{\Lambda} = \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{z}}, \tag{21}$$

evolves backward in time from identity $\boldsymbol{\Lambda}_1 = \boldsymbol{I}$ by

$$\dot{\boldsymbol{\Lambda}} = -\boldsymbol{\Lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}. \tag{22}$$

Equation (22) is called the *adjoint equation*, for the right-hand side is $(\partial \boldsymbol{f}/\partial \boldsymbol{z})^\top \boldsymbol{\Lambda}$. It can be derived from the time invariance of $\boldsymbol{\Lambda} \cdot \boldsymbol{\Delta}$:

$$\boldsymbol{\Lambda} \cdot \dot{\boldsymbol{\Delta}} + \dot{\boldsymbol{\Lambda}} \cdot \boldsymbol{\Delta} = \frac{d}{dt}(\boldsymbol{\Lambda} \cdot \boldsymbol{\Delta}) = \frac{d}{dt}\left(\frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}_0}\right) = \frac{d}{dt}\frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{z}_0} = 0. \tag{23}$$

Alternatively, the adjoint equation can be derived from the variational equation, using the facts that $d\boldsymbol{M}^{-1}/dt = -\boldsymbol{M}^{-1}\dot{\boldsymbol{M}}\,\boldsymbol{M}^{-1}$ for any invertible matrix $\boldsymbol{M}$, and $(\partial \boldsymbol{z}/\partial \tilde{\boldsymbol{z}})^{-1} = \partial \tilde{\boldsymbol{z}}/\partial \boldsymbol{z}$. Like Equations (20) and (22)

$$\frac{d}{dt}\frac{\partial \boldsymbol{z}}{\partial \tilde{\boldsymbol{z}}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{z}}{\partial \tilde{\boldsymbol{z}}}, \qquad \frac{d}{dt}\frac{\partial \tilde{\boldsymbol{z}}}{\partial \boldsymbol{z}} = -\frac{\partial \tilde{\boldsymbol{z}}}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}. \tag{24}$$

As we can see next, the adjoint equation takes a similar form when one optimizes an objective function of the state.

3.2. Objective on the Final State

In the simplest case, the objective function depends only on the final state, e.g., the last snapshot of a simulation, and possibly the parameters too in the form of regularization or prior information, i.e., $\mathcal{J}(\boldsymbol{z}_1, \boldsymbol{\theta})$. To optimize the objective under the constraint given by the ODEs, we can introduce a time-dependent function $\boldsymbol{\lambda}(t)$ as the Lagrange multiplier:

$$\mathcal{L} = \mathcal{J}(\boldsymbol{z}_1, \boldsymbol{\theta}) - \int_{t_0}^{t_1} \boldsymbol{\lambda}(t) \cdot [\dot{\boldsymbol{z}} - \boldsymbol{f}(\boldsymbol{z}, t, \boldsymbol{\theta})]\, dt. \tag{25}$$

Note the minus sign we have introduced in front of $\boldsymbol{\lambda}$ for later convenience.

Its total derivative with respect to $\boldsymbol{\theta}$ is

$$\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}_1} \cdot \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{\theta}} - \int_{t_0}^{t_1} \boldsymbol{\lambda} \cdot \left(\frac{\partial \dot{\boldsymbol{z}}}{\partial \boldsymbol{\theta}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}\right) dt. \tag{26}$$

Integrating the first term of the integrand by parts

$$\int_{t_0}^{t_1} \boldsymbol{\lambda} \cdot \frac{\partial \dot{\boldsymbol{z}}}{\partial \boldsymbol{\theta}}\, dt = \int_{t_0}^{t_1} \boldsymbol{\lambda} \cdot \frac{\partial^2 \boldsymbol{z}}{\partial \boldsymbol{\theta} \partial t}\, dt = \left[\boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}}\right]_{t_0}^{t_1} - \int_{t_0}^{t_1} \dot{\boldsymbol{\lambda}} \cdot \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}}\, dt, \tag{27}$$

and plugging it back

$$\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} + \left(\frac{\partial \mathcal{J}}{\partial \boldsymbol{z}_1} - \boldsymbol{\lambda}_1\right) \cdot \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{\theta}} + \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{\theta}} + \int_{t_0}^{t_1} \left[\left(\boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} + \dot{\boldsymbol{\lambda}}\right) \cdot \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}} + \boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}\right] dt. \tag{28}$$

Now we are free to choose

$$\dot{\boldsymbol{\lambda}} = -\boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}, \qquad \boldsymbol{\lambda}_1 = \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}_1}, \tag{29}$$

which allows us to avoid all $\partial \boldsymbol{z}/\partial \boldsymbol{\theta}$ terms in the final objective gradient:

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} + \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{\theta}} + \int_{t_0}^{t_1} \boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}\, dt, \tag{30}$$

in which the first two terms come from the regularization and initial conditions, respectively. Equation (29) is the adjoint equation for the objective $\mathcal{J}(\boldsymbol{z}_1, \boldsymbol{\theta})$. With the initial conditions set at the final time, we can integrate it backward in time to obtain $\boldsymbol{\lambda}(t)$, which enters the above equation and yields $d\mathcal{J}/d\boldsymbol{\theta}$.

Note that Equation (29) has the same form as Equation (22), and their solutions are related by

$$\boldsymbol{\lambda} = \boldsymbol{\lambda}_1 \cdot \boldsymbol{\Lambda} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}_1} \cdot \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{z}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}}. \tag{31}$$

And like $\boldsymbol{\Lambda} \cdot \boldsymbol{\Delta}$, $\boldsymbol{\lambda} \cdot \boldsymbol{\Delta}$ is time invariant:

$$\boldsymbol{\lambda}_0 \cdot \boldsymbol{\Delta}_0 = \boldsymbol{\lambda}_1 \cdot \boldsymbol{\Delta}_1 = \boldsymbol{\lambda} \cdot \boldsymbol{\Delta} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{z}_0} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}_0} = \boldsymbol{\lambda}_0. \tag{32}$$

Computing $\boldsymbol{\lambda}_1 \cdot \boldsymbol{\Delta}_1$ directly is expensive, but solving Equation (29) backward for $\boldsymbol{\lambda}_0$ is cheap. This is related to the fact that the reverse-mode AD or backpropagation is cheaper than the forward mode for optimization.

3.3. Objective on the State History

The adjoint method applies to more complex cases too. Let us consider an objective as a functional of the evolution history

with some regularization $\mathcal{R}$ on $\boldsymbol{\theta}$

$$\mathcal{J} = \mathcal{R}(\boldsymbol{\theta}) + \int_{t_0}^{t_1} g(\boldsymbol{z}, t, \boldsymbol{\theta})\, dt. \tag{33}$$

The derivation is similar:

$$\mathcal{L} = \mathcal{R}(\boldsymbol{\theta}) + \int_{t_0}^{t_1} \{g(\boldsymbol{z}, t, \boldsymbol{\theta}) - \boldsymbol{\lambda}(t) \cdot [\dot{\boldsymbol{z}} - \boldsymbol{f}(\boldsymbol{z}, t, \boldsymbol{\theta})]\}\, dt \tag{34}$$

has the gradient

$$\begin{aligned}\frac{d\mathcal{L}}{d\boldsymbol{\theta}} &= \frac{d\mathcal{R}}{d\boldsymbol{\theta}} + \int_{t_0}^{t_1} \left[\frac{\partial g}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}} + \frac{\partial g}{\partial \boldsymbol{\theta}} - \boldsymbol{\lambda} \cdot \left(\frac{\partial \dot{\boldsymbol{z}}}{\partial \boldsymbol{\theta}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} \cdot \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}\right)\right] dt \\ &= \frac{d\mathcal{R}}{d\boldsymbol{\theta}} - \boldsymbol{\lambda}_1 \cdot \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{\theta}} + \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{\theta}} + \int_{t_0}^{t_1} \left[\left(\frac{\partial g}{\partial \boldsymbol{z}} + \boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} + \dot{\boldsymbol{\lambda}}\right) \cdot \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}} + \frac{\partial g}{\partial \boldsymbol{\theta}} + \boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}\right] dt. \end{aligned} \tag{35}$$

The adjoint equation becomes

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial g}{\partial \boldsymbol{z}} - \boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}, \qquad \boldsymbol{\lambda}_1 = 0, \tag{36}$$

with the objective gradient

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \frac{d\mathcal{R}}{d\boldsymbol{\theta}} + \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{\theta}} + \int_{t_0}^{t_1} \left(\frac{\partial g}{\partial \boldsymbol{\theta}} + \boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}\right) dt. \tag{37}$$

The time invariant is now

$$\begin{aligned}\boldsymbol{\lambda}_1 \cdot \boldsymbol{\Delta}_1 &= \boldsymbol{\lambda}_0 \cdot \boldsymbol{\Delta}_0 + \int_{t_0}^{t_1} \frac{\partial g}{\partial \boldsymbol{z}} \cdot \boldsymbol{\Delta}\, dt \\ &= \boldsymbol{\lambda} \cdot \boldsymbol{\Delta} + \int_{t}^{t_1} \frac{\partial g}{\partial \boldsymbol{z}} \cdot \boldsymbol{\Delta}\, dt' = 0, \end{aligned} \tag{38}$$

so

$$\boldsymbol{\lambda} \cdot \boldsymbol{\Delta} = -\int_{t}^{t_1} \frac{\partial g}{\partial \boldsymbol{z}} \cdot \boldsymbol{\Delta}\, dt' = -\int_{t}^{t_1} \frac{\partial g}{\partial \boldsymbol{z}_0}\, dt'. \tag{39}$$

3.4. Objective on the Observables

Now let us consider an objective that depends on different state components at different times, e.g., in the Universe where further objects intersected our past light cone earlier. It falls between the previous two scenarios, and we can derive its adjoint equation similarly.

We denote the observables by $\hat{\boldsymbol{z}}$, with different components $\hat{z}_j$ affecting the objective $\mathcal{J}(\hat{\boldsymbol{z}}, \boldsymbol{\theta})$ at different times $\hat{t}_j$, i.e., $\hat{z}_j \triangleq z_j(\hat{t}_j)$. The Lagrangian becomes

$$\mathcal{L} = \mathcal{J}(\hat{\boldsymbol{z}}, \boldsymbol{\theta}) - \sum_j \int_{t_0}^{\hat{t}_j} \lambda_j(t)[\dot{z}_j - f_j(\boldsymbol{z}, t, \boldsymbol{\theta})]\, dt, \tag{40}$$

constraining only the parts of trajectories inside the light cone. Its gradient is

$$\begin{aligned}\frac{d\mathcal{L}}{d\boldsymbol{\theta}} &= \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{J}}{\partial \hat{\boldsymbol{z}}} \cdot \frac{\partial \hat{\boldsymbol{z}}}{\partial \boldsymbol{\theta}} - \sum_j \int_{t_0}^{\hat{t}_j} \left(\lambda_j \frac{\partial \dot{z}_j}{\partial \boldsymbol{\theta}} - \boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial z_j} \frac{\partial z_j}{\partial \boldsymbol{\theta}} - \lambda_j \frac{\partial f_j}{\partial \boldsymbol{\theta}}\right) dt \\ &= \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} + \left(\frac{\partial \mathcal{J}}{\partial \hat{\boldsymbol{z}}} - \hat{\boldsymbol{\lambda}}\right) \cdot \frac{\partial \hat{\boldsymbol{z}}}{\partial \boldsymbol{\theta}} + \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{\theta}} + \sum_j \int_{t_0}^{\hat{t}_j} \left[\left(\boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial z_j} + \dot{\lambda}_j\right) \frac{\partial z_j}{\partial \boldsymbol{\theta}} + \lambda_j \frac{\partial f_j}{\partial \boldsymbol{\theta}}\right] dt, \end{aligned} \tag{41}$$

where we have defined $\hat{\boldsymbol{\lambda}}$ similarly with components $\hat{\lambda}_j \triangleq \lambda_j(\hat{t}_j)$. In the first equality, we have also dropped a vanishing term $\sum_j \hat{\lambda}_j[\dot{z}_j - f_j(\hat{t}_j)]\partial \hat{t}_j / \partial \boldsymbol{\theta}$, i.e., $\partial \hat{t}_j / \partial \boldsymbol{\theta}$ does not directly enter the gradient.

Now we find the adjoint equation

$$\dot{\boldsymbol{\lambda}} = -\boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}, \qquad \hat{\boldsymbol{\lambda}} = \frac{\partial \mathcal{J}}{\partial \hat{\boldsymbol{z}}}, \tag{42}$$

which has the same form as Equation (29), with a slightly different initial condition given at the respective observation time of each component. The objective gradient is also similar to the previous cases:

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} + \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{\theta}} + \sum_j \int_{t_0}^{\hat{t}_j} \lambda_j \frac{\partial f_j}{\partial \boldsymbol{\theta}}\, dt. \tag{43}$$

Note that even though $\lambda_j(t)$ for $t > \hat{t}_j$ does not affect the final gradient, they can enter the right-hand side of the adjoint equation, and affect those λ_k with $t < \hat{t}_k$, i.e., inside the light cone. Physically, however, $\partial f_j / \partial z_k$ should vanish for spacelike separated pairs of z_j and z_k, even though the Newtonian approximation we adopt introduces some small deviation. Therefore, we can set $\lambda_j(t)$ to 0 for $t > \hat{t}_j$, and bump it to $\partial \mathcal{J} / \partial \hat{z}_j$ at $\hat{t}_j$.

3.5. Discretize Then Optimize

In practice, the time integration of Equation (18) is discrete. Consider the explicit methods,

$$\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + \boldsymbol{F}_i(\boldsymbol{z}_i, \boldsymbol{\theta}), \qquad i = 0, \cdots, n-1, \tag{44}$$

which include the leapfrog integrator commonly used for Hamiltonian dynamics. We want to propagate the gradients backward along the *same* discrete trajectory as taken by the forward integration. Therefore, instead of the continuous adjoint equations derived above, we need the adjoint method for the discrete integrator.

Without loss of generality, we derive the adjoint equation for an objective depending on the state at all time steps, which can be easily specialized to the three cases discussed above with

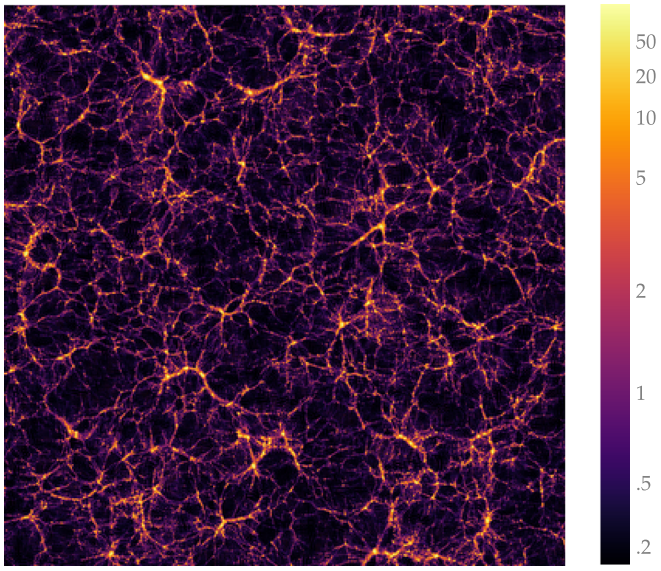


Figure 2. Relative matter density field, $1+\delta$, at $a=1$, projected from an 8 Mpc h^{-1} thick slab in a pmwd simulation, that has evolved 512^3 particles with single precision and a 1024^3 mesh in a (512 Mpc h^{-1})^3 box for 63 time steps. The simulation takes only 13 s to finish on an NVIDIA H100 PCIe GPU.

only slight modifications. The discretized Lagrangian is now

$$\mathcal{L} = \mathcal{J}(\boldsymbol{z}_0,\cdots,\boldsymbol{z}_n,\boldsymbol{\theta}) - \sum_{i=0}^{n-1} \boldsymbol{\lambda}_{i+1}\cdot[\boldsymbol{z}_{i+1} - \boldsymbol{z}_i - \boldsymbol{F}_i(\boldsymbol{z}_i,\boldsymbol{\theta})], \tag{45}$$

whose gradient is

$$\begin{aligned}
\frac{d\mathcal{L}}{d\boldsymbol{\theta}} &= \frac{\partial\mathcal{J}}{\partial\boldsymbol{\theta}} + \sum_{i=0}^{n}\frac{\partial\mathcal{J}}{\partial\boldsymbol{z}_i}\cdot\frac{\partial\boldsymbol{z}_i}{\partial\boldsymbol{\theta}} - \sum_{i=0}^{n-1}\boldsymbol{\lambda}_{i+1}\cdot\left(\frac{\partial\boldsymbol{z}_{i+1}}{\partial\boldsymbol{\theta}} - \frac{\partial\boldsymbol{z}_i}{\partial\boldsymbol{\theta}} - \frac{\partial\boldsymbol{F}_i}{\partial\boldsymbol{z}_i}\cdot\frac{\partial\boldsymbol{z}_i}{\partial\boldsymbol{\theta}} - \frac{\partial\boldsymbol{F}_i}{\partial\boldsymbol{\theta}}\right)\\
&= \frac{\partial\mathcal{J}}{\partial\boldsymbol{\theta}} + \sum_{i=0}^{n}\frac{\partial\mathcal{J}}{\partial\boldsymbol{z}_i}\cdot\frac{\partial\boldsymbol{z}_i}{\partial\boldsymbol{\theta}} - \sum_{i=1}^{n}\boldsymbol{\lambda}_i\cdot\frac{\partial\boldsymbol{z}_i}{\partial\boldsymbol{\theta}} + \sum_{i=0}^{n-1}\boldsymbol{\lambda}_{i+1}\cdot\left(\frac{\partial\boldsymbol{z}_i}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{F}_i}{\partial\boldsymbol{z}_i}\cdot\frac{\partial\boldsymbol{z}_i}{\partial\boldsymbol{\theta}} + \frac{\partial\boldsymbol{F}_i}{\partial\boldsymbol{\theta}}\right)\\
&= \frac{\partial\mathcal{J}}{\partial\boldsymbol{\theta}} + \left(\frac{\partial\mathcal{J}}{\partial\boldsymbol{z}_n} - \boldsymbol{\lambda}_n\right)\cdot\frac{\partial\boldsymbol{z}_n}{\partial\boldsymbol{\theta}} + \boldsymbol{\lambda}_0\cdot\frac{\partial\boldsymbol{z}_0}{\partial\boldsymbol{\theta}}\\
&\quad + \sum_{i=0}^{n-1}\left[\left(\frac{\partial\mathcal{J}}{\partial\boldsymbol{z}_i} - \boldsymbol{\lambda}_i + \boldsymbol{\lambda}_{i+1} + \boldsymbol{\lambda}_{i+1}\cdot\frac{\partial\boldsymbol{F}_i}{\partial\boldsymbol{z}_i}\right)\cdot\frac{\partial\boldsymbol{z}_i}{\partial\boldsymbol{\theta}} + \boldsymbol{\lambda}_{i+1}\cdot\frac{\partial\boldsymbol{F}_i}{\partial\boldsymbol{\theta}}\right].
\end{aligned} \tag{46}$$

So the discrete adjoint equation is

$$\boldsymbol{\lambda}_{i-1} = \boldsymbol{\lambda}_i + \boldsymbol{\lambda}_i\cdot\frac{\partial\boldsymbol{F}_{i-1}}{\partial\boldsymbol{z}_{i-1}} + \frac{\partial\mathcal{J}}{\partial\boldsymbol{z}_{i-1}}, \quad i=n,\cdots,1; \qquad \boldsymbol{\lambda}_n = \frac{\partial\mathcal{J}}{\partial\boldsymbol{z}_n}. \tag{47}$$

We can iterate it backward in time to compute the final objective gradient:

$$\frac{d\mathcal{J}}{d\boldsymbol{\theta}} = \frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \frac{\partial\mathcal{J}}{\partial\boldsymbol{\theta}} + \boldsymbol{\lambda}_0\cdot\frac{\partial\boldsymbol{z}_0}{\partial\boldsymbol{\theta}} + \sum_{i=1}^{n}\boldsymbol{\lambda}_i\cdot\frac{\partial\boldsymbol{F}_{i-1}}{\partial\boldsymbol{\theta}}. \tag{48}$$

These equations are readily adaptable to simulated observables. For snapshots at t_n or interpolated between t_{n-1} and t_n, all $\partial\mathcal{J}/\partial\boldsymbol{z}_i$ vanish except for the last one or two, respectively. For light cones, as discussed in Section 3.4, each component of $\hat{\boldsymbol{z}}$ is interpolated at different times; thus, all $\partial\mathcal{J}/\partial\boldsymbol{z}_i$ vanish except for those times relevant for its interpolation, and the corresponding $\boldsymbol{\lambda}_i$ can be set to zero for i greater than the intersection time.

At the ith iteration, the adjoint variable requires the vector-Jacobian product (VJP) $\boldsymbol{\lambda}_i\cdot\partial\boldsymbol{F}_{i-1}/\partial\boldsymbol{z}_{i-1}$ and the partial objective derivative $\partial\mathcal{J}/\partial\boldsymbol{z}_{i-1}$ at the next time step, which can be easily computed by AD if the whole forward history of Equation (44) has been saved. However, this can be extremely costly in memory, which can be alleviated by checkpointing algorithms such as `Revolve` and its successors (Griewank & Walther 2000). Alternatively, if a solution to Equation (18) is unique, we can integrate it backward and recover the history, which is easy for reversible Hamiltonian dynamics and with reversible integrators such as leapfrog. When the N-body dynamics become too chaotic, one can use more precise floating-point numbers and/or save multiple checkpoints[13] during the forward evolution, from which the backward evolution can be resumed piecewise.

3.6. Application to Simulation

The adjoint method provides systematic ways of deriving the objective gradient under constraints (Pontryagin 1962), here imposed by the N-body equations of motion. We have introduced above the adjoint method for generic time-dependent problems in both continuous and discrete cases. The continuous case is easier to understand and has pedagogical values, while the discrete case is the useful one in our application, for we want to propagate numerically the gradients backward along the *same* path as that of the forward time integration.

For the N-body particles, the state variable[14] is $\boldsymbol{z}=(\boldsymbol{x},\boldsymbol{p})^\top$. Their adjoint variables help to accumulate the objective gradient while evolving backward in time by the adjoint equation. Let us denote them by $\boldsymbol{\lambda}=(\boldsymbol{\xi},\boldsymbol{\pi})$. We can compare each step of Equations (16)–(44), and write down its adjoint equation following Equation (47). Taking D_{i-1}^i for example, we

[13] This is different from the checkpointing in the `Revolve` algorithm, which needs to rerun the forward iterations.

[14] State and adjoint vectors in the adjoint equations include enumeration of particles, e.g., $\nabla\boldsymbol{\varphi}$ includes the $\nabla\varphi$ of each particle.

can write it explicitly as

$$\begin{pmatrix} \boldsymbol{x}_i \\ \boldsymbol{p}_{i-1/2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_{i-1} \\ \boldsymbol{p}_{i-1/2} \end{pmatrix} + \begin{pmatrix} \boldsymbol{p}_{i-1/2} D(t_{i-1/2}, t_{i-1}, t_i) \\ 0 \end{pmatrix},$$

in the form of Equation (44). By Equation (47), its adjoint equation is

$$\begin{aligned} \begin{pmatrix} \boldsymbol{\xi}_{i-1/2} & \boldsymbol{\pi}_{i-1} \end{pmatrix} &= \begin{pmatrix} \boldsymbol{\xi}_{i-1/2} & \boldsymbol{\pi}_i \end{pmatrix} + \begin{pmatrix} \boldsymbol{\xi}_{i-1/2} & \boldsymbol{\pi}_i \end{pmatrix} \begin{pmatrix} 0 & \boldsymbol{I}\, D(t_{i-1/2}, t_{i-1}, t_i) \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\xi}_{i-1/2} & \boldsymbol{\pi}_i \end{pmatrix} - \begin{pmatrix} 0 & \boldsymbol{\xi}_{i-1/2} D(t_{i-1/2}, t_i, t_{i-1}) \end{pmatrix}, \end{aligned}$$

where we have used the fact that $D(t_c, t_a, t_b) = -D(t_c, t_b, t_a)$, and left the $\partial \mathcal{J}/\partial \boldsymbol{z}_i$ term, the explicit dependence of the objective on the intermediate states (from, e.g., observables on the light cone), to the observation operator O below. This also naturally determines the subscripts of ξ and π.

Repeating the derivation for K and O, and flipping the arrow of time, we present the adjoint equation time stepping for Equation (16) from t_i to t_{i-1}:

$$\begin{aligned} K_i^{i-1/2}\colon \quad & \boldsymbol{p}_{i-1/2} = \boldsymbol{p}_i + \boldsymbol{a}_i K(t_i, t_i, t_{i-1/2}), \\ & \boldsymbol{\xi}_{i-1/2} = \boldsymbol{\xi}_i - \boldsymbol{\alpha}_i K(t_i, t_i, t_{i-1/2}), \\ D_i^{i-1}\colon \quad & \boldsymbol{x}_{i-1} = \boldsymbol{x}_i + \boldsymbol{p}_{i-1/2} D(t_{i-1/2}, t_i, t_{i-1}), \\ & \boldsymbol{\pi}_{i-1} = \boldsymbol{\pi}_i - \boldsymbol{\xi}_{i-1/2} D(t_{i-1/2}, t_i, t_{i-1}), \\ F_{i-1}\colon \quad & \boldsymbol{a}_{i-1} = -\nabla \varphi(\boldsymbol{x}_{i-1}), \\ & \boldsymbol{\alpha}_{i-1} \triangleq -\boldsymbol{\pi}_{i-1} \cdot \frac{\partial \nabla \varphi_{i-1}}{\partial \boldsymbol{x}_{i-1}}, \\ & \boldsymbol{\zeta}_{i-1} \triangleq -\boldsymbol{\pi}_{i-1} \cdot \frac{\partial \nabla \varphi_{i-1}}{\partial \boldsymbol{\theta}}, \\ K_{i-1/2}^{i-1}\colon \quad & \boldsymbol{p}_{i-1} = \boldsymbol{p}_{i-1/2} + \boldsymbol{a}_{i-1} K(t_{i-1}, t_{i-1/2}, t_{i-1}), \\ & \boldsymbol{\xi}_{i-1} = \boldsymbol{\xi}_{i-1/2} - \boldsymbol{\alpha}_{i-1} K(t_{i-1}, t_{i-1/2}, t_{i-1}), \\ O_i^{i-1}\colon \quad & \boldsymbol{\xi}_{i-1} :{=} \boldsymbol{\xi}_{i-1} + \frac{\partial \mathcal{J}}{\partial \boldsymbol{x}_{i-1}}, \\ & \boldsymbol{\pi}_{i-1} :{=} \boldsymbol{\pi}_{i-1} + \frac{\partial \mathcal{J}}{\partial \boldsymbol{p}_{i-1}}. \end{aligned} \tag{49}$$

Like $\boldsymbol{a}$, we have introduced $\boldsymbol{\alpha}$ and $\boldsymbol{\zeta}$ to cache the vector-Jacobian products on their right-hand sides, for the next time step in the kick operator and the objective gradient (see below), respectively. Note that in the reverse order, the F operator is at t_{i-1} instead of t_i as in Equation (16), and we need to initialize $\boldsymbol{a}_n$, $\boldsymbol{\alpha}_n$, and $\boldsymbol{\zeta}_n$ with F_n before stepping from t_n to t_{n-1}. Likewise, the gradient of O_n^{n-1} at t_n is absent in Equation (49) but enters via the initial conditions following (47). Explicitly, the initial conditions of Equation (49) are

$$\begin{aligned} & \boldsymbol{\xi}_n = \frac{\partial \mathcal{J}}{\partial \boldsymbol{x}_n}, \ \boldsymbol{\pi}_n = \frac{\partial \mathcal{J}}{\partial \boldsymbol{p}_n}, \ \boldsymbol{a}_n = -\nabla \varphi(\boldsymbol{x}_n), \\ & \boldsymbol{\alpha}_n = -\boldsymbol{\pi}_n \cdot \frac{\partial \nabla \varphi_n}{\partial \boldsymbol{x}_n}, \ \boldsymbol{\zeta}_n = -\boldsymbol{\pi}_n \cdot \frac{\partial \nabla \varphi_n}{\partial \boldsymbol{\theta}}. \end{aligned} \tag{50}$$

The VJPs in F and the $\partial \mathcal{J}/\partial \boldsymbol{z}$'s in O can be computed by AD if the whole forward integration and observation history of Equations (16) and (17) has been saved. However, this can be too costly spatially for GPUs, whose memories are much smaller than those of CPUs. Alternatively, we take advantage of the reversibility of the N-body dynamics and the leapfrog integrator, and recover the history by reverse time integration, which we have already included on the first lines of the K and D operators in Equation (49). We can integrate the leapfrog and the adjoint equations jointly backward in time, and still benefit from the convenience of AD in computing VJPs and $\partial \mathcal{J}/\partial \boldsymbol{z}$'s. In practice, the numerical reversibility suffers from the finite precision and the chaotic N-body dynamics, which we find is generally not a concern for our applications in the results section.

Finally, during the reverse time integration, we can accumulate the objective gradient following Equation (48):

$$\begin{aligned} \frac{d\mathcal{J}}{d\boldsymbol{\theta}} = & \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{J}}{\partial \hat{\boldsymbol{x}}} \cdot \frac{\partial \hat{\boldsymbol{x}}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{J}}{\partial \hat{\boldsymbol{p}}} \cdot \frac{\partial \hat{\boldsymbol{p}}}{\partial \boldsymbol{\theta}} \\ & + \boldsymbol{\xi}_0 \cdot \frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{\theta}} + \boldsymbol{\pi}_0 \cdot \frac{\partial \boldsymbol{p}_0}{\partial \boldsymbol{\theta}} \\ & - \sum_{i=1}^{n} \Bigg[\left(\boldsymbol{\pi}_i \cdot \boldsymbol{a}_i \frac{\partial}{\partial \boldsymbol{\theta}} + \boldsymbol{\zeta}_i \right) K(t_i, t_i, t_{i-1/2}) \\ & + \boldsymbol{\xi}_{i-1/2} \cdot \boldsymbol{p}_{i-1/2} \frac{\partial D(t_{i-1/2}, t_i, t_{i-1})}{\partial \boldsymbol{\theta}} \\ & + \left(\boldsymbol{\pi}_{i-1} \cdot \boldsymbol{a}_{i-1} \frac{\partial}{\partial \boldsymbol{\theta}} + \boldsymbol{\zeta}_{i-1} \right) K(t_{i-1}, t_{i-1/2}, t_{i-1}) \Bigg], \\ \frac{d\mathcal{J}}{d\boldsymbol{\omega}} = & \frac{\partial \mathcal{J}}{\partial \boldsymbol{\omega}} + \boldsymbol{\xi}_0 \cdot \frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{\omega}} + \boldsymbol{\pi}_0 \cdot \frac{\partial \boldsymbol{p}_0}{\partial \boldsymbol{\omega}}, \end{aligned} \tag{51}$$

where the latter backpropagates from fewer sources than the former as shown in Figure 1(a). To implement Equations (49)–(51) in pmwd with `JAX`, we only need to write custom VJP rules for the high-level N-body integration-observation loop, while the derivatives and VJPs of the remaining parts, including regularization/prior, the observation, the initial conditions, the kick and drift factors, the growth and transfer functions, etc., can all be conveniently computed by AD.

Other than that, we also implement custom VJPs for the scatter and gather operations in Section 2.2 following https://web.archive.org/web/20230529030440/https://bids.berkeley.edu/news/automatic-differentiation-and-cosmology-simulation, and these further save memory in gradient computations of those nonlinear functions.

4. Results

We implement the forward simulation and backward adjoint method in the PM code pmwd. All results assume a simple ΛCDM cosmology: $\boldsymbol{\theta} = (A_s = 2 \times 10^{-9},\ n_s = 0.96,\ \Omega_m = 0.3,\ \Omega_b = 0.05,\ h = 0.7)$, and use an NVIDIA H100 PCIe GPU with 80 GB memory.

As in `FastPM`, the choice of time steps is flexible, and in fact, with pmwd it can even be optimized in non-parametric ways to improve simulation accuracy at fixed computation cost. Here we use time steps linearly spaced in scale factor a, and leave such optimization to follow-up work.

4.1. Simulation

We first test the forward simulations of pmwd. Figure 2 shows the cosmic web in the final snapshot of a fairly large simulation for the size of GPU memories.

Because GPUs are inherently parallel devices, they can output different results for identical inputs. To test the reproducibility, in Table 1 we compare the rms deviations (RMSDs) of particle displacements and velocities between two runs, relative to their respective standard deviations, with

Table 1
pmwd Reproducibility on a GPU

Precision	Cell/ptcl	ptcl Mass [$10^{10} M_\odot$]	Time Steps	Disp. Rel. Diff.	Vel. Rel. Diff.
Single	8	1	63	1.6×10^{-6}	4.8×10^{-5}
Single	8	1	126	5.1×10^{-7}	1.3×10^{-5}
Single	8	8	63	3.1×10^{-7}	2.4×10^{-6}
Single	8	8	126	3.1×10^{-7}	2.3×10^{-6}
Single	1	1	63	1.4×10^{-7}	1.5×10^{-6}
Single	1	1	126	1.3×10^{-7}	1.4×10^{-6}
Single	1	8	63	1.1×10^{-7}	4.5×10^{-7}
Single	1	8	126	9.9×10^{-8}	4.4×10^{-7}
Double	8	1	63	1.4×10^{-15}	3.6×10^{-14}
Double	8	1	126	8.3×10^{-16}	1.7×10^{-14}
Double	8	8	63	5.4×10^{-16}	4.1×10^{-15}
Double	8	8	126	5.9×10^{-16}	4.3×10^{-15}
Double	1	1	63	2.4×10^{-16}	2.3×10^{-15}
Double	1	1	126	2.3×10^{-16}	2.3×10^{-15}
Double	1	8	63	1.9×10^{-16}	7.9×10^{-16}
Double	1	8	126	1.8×10^{-16}	7.8×10^{-16}

Note. GPUs can output different results for identical inputs. We simulate 384^3 particles from $a = 1/64$ to $a = 1$, with two floating-point precisions, two mesh sizes, two particle masses (with box sizes of 192 and 384 Mpc h^{-1}), and two time step sizes. We take the RMSDs of particle displacements and velocities between two runs at $a = 1$, and quote their ratios to the respective standard deviations, about 6 Mpc h^{-1} and 3×10^2 km s^{-1}. In general, the factors on the left of the left four columns affect the reproducibility more than those on the right, and lower rows are more reproducible than the upper ones.

Table 2
pmwd Reversibility on a GPU

Precision	Cell/ptcl	ptcl Mass [$10^{10} M_\odot$]	Time Steps	Disp. Rel. Diff.	Vel. Rel. Diff.
Single	8	1	63	5.2×10^{-2}	7.1×10^{-2}
Single	8	1	126	2.1×10^{-2}	3.6×10^{-2}
Single	8	8	63	3.3×10^{-3}	7.6×10^{-3}
Single	8	8	126	3.7×10^{-3}	7.0×10^{-3}
Single	1	1	63	1.4×10^{-3}	2.2×10^{-3}
Single	1	1	126	1.3×10^{-3}	1.7×10^{-3}
Single	1	8	63	4.3×10^{-4}	7.3×10^{-4}
Single	1	8	126	4.4×10^{-4}	6.3×10^{-4}
Double	8	1	63	5.4×10^{-11}	1.3×10^{-10}
Double	8	1	126	3.5×10^{-11}	7.0×10^{-11}
Double	8	8	63	5.8×10^{-12}	1.4×10^{-11}
Double	8	8	126	6.4×10^{-12}	1.2×10^{-11}
Double	1	1	63	2.2×10^{-12}	3.4×10^{-12}
Double	1	1	126	2.1×10^{-12}	2.9×10^{-12}
Double	1	8	63	7.2×10^{-13}	1.2×10^{-12}
Double	1	8	126	6.9×10^{-13}	9.4×10^{-13}

Note. Our adjoint method reduces memory cost by reconstructing the forward evolution with reverse time integration. We test the numerical reversibility by comparing the displacements and velocities of particles that have evolved to $a = 1$ and then reversed to $a = 1/64$, to those of the LPT initial conditions at $a = 1/64$, in RMSDs. We take their ratios to the respective standard deviations, about 0.1 Mpc h^{-1} and 0.7 km s^{-1}. Their smallness is the main reason that the quoted relative differences here are orders of magnitude greater than those in Table 1 (see Figure 3 where the reversibility is a few times more important than the reproducibility in their impacts on the gradients). With the same setup as that in Table 1, we see the same general trend—the left factors are more important than the right ones, and the lower rows are more reversible than the upper ones.

different floating-point precisions, mesh sizes, particle masses, and numbers of time steps. Other than the precision, mesh size is the most important factor because a finer mesh can better resolve the most nonlinear and dense structures, which can affect reproducibility as the order of many operations can change easily. The particle mass plays a similar role and less massive particles generally take part in nonlinear motions at earlier times. The number of time steps has a very small impact except in the most nonlinear cases. And interestingly, more time steps improve the reproducibility most of the time.

4.2. Differentiation

Model differentiation evolves the adjoint equations backward in time. To save memory, the trajectory of the model state in the forward run is not saved, but resimulated together with the adjoint equations. Even though in principle the N-body systems are reversible, in practice, the reconstructed trajectory can differ from the forward one due to the finite numerical precision and exacerbated by the chaotic dynamics. Better reversibility means the gradients propagate backward along a

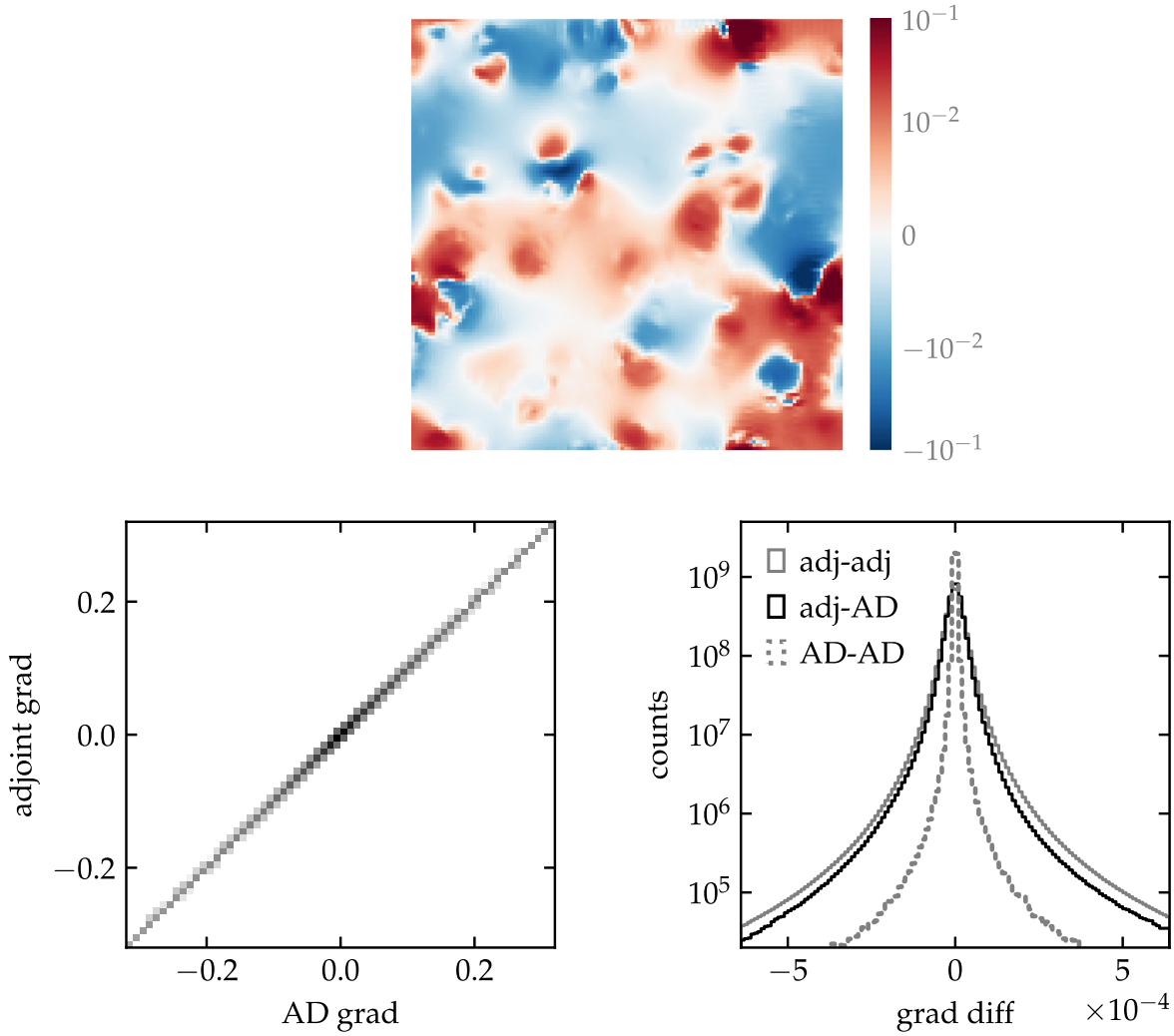
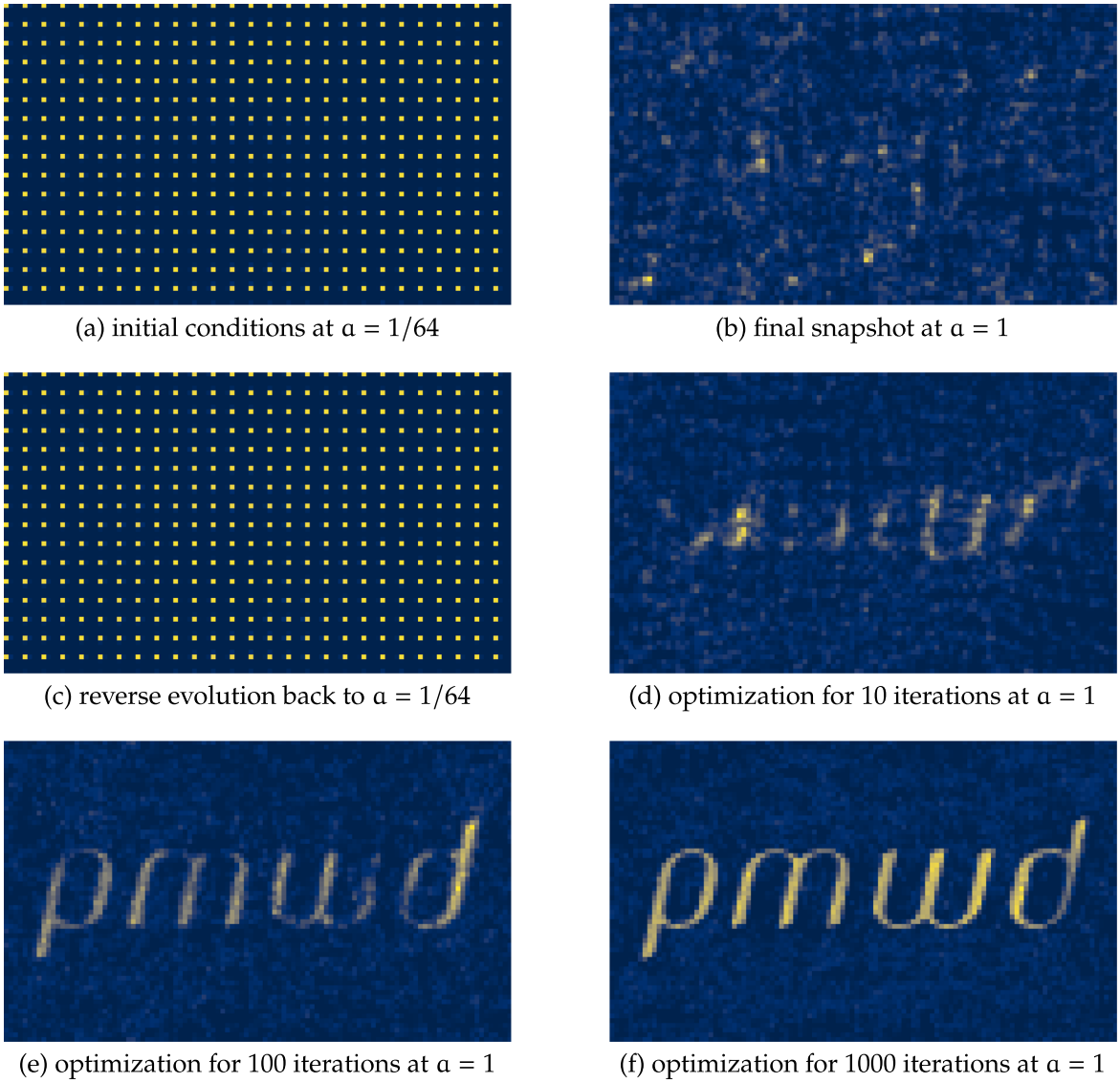


Figure 3. Adjoint gradients of a 128 × 128 slice of the real white noise field $d\mathcal{J}/d\boldsymbol{\omega}$ (top panel), in a pmwd simulation of 128^3 particles in a (128 Mpc h^{-1})^3 box, with a 256^3 mesh, 15 time steps, and single precision. We choose a mean squared error (MSE) objective between two realizations with the same cosmology but different initial modes, on their density fields on the 256^3 mesh at $a = 1$, and then compute the gradients with respect to one realization, while holding the other fixed. We compare the adjoint gradients to those by AD, for which we have disabled the custom gradient implementation on the scatter, gather, and N-body time stepping operators. The adjoint and AD gradients agree as expected, with a RMSD of $\approx 4 \times 10^{-5}$, 3 orders of magnitude smaller than the standard deviation of the gradients itself, ≈0.015. It is also comparable to the difference between two different adjoint gradients, with a RMSD $\approx 5 \times 10^{-5}$. Different AD gradients are more consistent, with a tighter RMSD of $\approx 1 \times 10^{-5}$, due to the absence of uncertainty from reverse time integration.

trajectory closer to the forward path, and thus would be more accurate. To test this, in Table 2 we compare the RMSDs of the forward-then-reverse particle displacements and velocities from the LPT initial conditions, relative to their respective standard deviations, which are very small at the *initial* time. As before, we vary the floating-point precision, the mesh size, the particle mass, and the number of time steps. The order of factor importance and their effects are the same as in the reproducibility test. This is because more nonlinear structures are more difficult to reverse. One way to improve reversibility is to use higher-order LPT to initialize the N-body simulations at later times (Michaux et al. 2021), when the displacements and velocities are not as small. We leave this for future development.

Next, we want to verify that our adjoint method yields the same gradients as those computed by AD. As explained in Section 3.6, pmwd already utilizes AD on most of the differentiation tasks. To get the AD gradients we disable our custom VJP implementations on the N-body time integration, and the scatter and gather operations. In Figure 3, we compare the adjoint and AD gradients on a smaller problem because AD already runs out of memory if we double the number of time steps or increase the space/mass resolution by $2^3\times$ from the listed specifications in the caption. For better statistics, we repeat both the adjoint and AD runs 64 times, with the same cosmology and white noise modes, and compare their results by an asymmetric difference of $X_i - Y_j$, where $1 \leqslant j < i \leqslant 64$. First, we set X and Y to the adjoint and AD gradients, respectively, and find they agree very well on the real white noise (so do their gradients on cosmological parameters not shown here; see https://github.com/eelregit/pmwd/tree/master/docs/papers/adjoint/grads.txt). In addition, we can set both X and Y to either adjoint or AD to check their respective *reproducibility*. We find both gradients are consistent among different runs of themselves, with AD being a lot more reproducible without uncertainty from the reverse time



(a) initial conditions at a = 1/64

(b) final snapshot at a = 1

(c) reverse evolution back to a = 1/64

(d) optimization for 10 iterations at a = 1

(e) optimization for 100 iterations at a = 1

(f) optimization for 1000 iterations at a = 1

Figure 4. A toy problem where we optimize the initial conditions by gradient descent to make some interesting patterns after projection. The particles originally fill a $16 \times 27 \times 16$ grid, and then evolve from $a = 1/64$ to $a = 1$ for 63 time steps with *single precision* and a $32 \times 54 \times 32$ mesh in a $160 \times 270 \times 160$ $\mathrm{Mpc}^3\ h^{-3}$ box. We compute their projected density in 64×108 pixels and compare that to the target image at the same resolution with an MSE objective. We use the adjoint method and reverse time integration, assuming the latter can reconstruct the forward evolution history accurately. We validate this by demonstrating that the particles evolve backward to align on the initial grid. The optimized initial conditions successfully evolve into the target pattern, which improves with more iterations. Also, see the animated reverse time evolution at https://youtu.be/Epsgh6vr0qs and initial condition optimization at https://youtu.be/vD6lbjHP3SY on YouTube.

integration but only that from the GPU reproducibility. This implies that we can ignore the reproducibility errors (Table 1) when the reversibility ones dominate (Table 2). Though this statement should be verified again in the future when we reduce the reversibility errors using, e.g., 3LPT.

Our last test on the adjoint gradients uses them in a toy optimization problem in Figure 4. We use the Adam optimizer (Kingma & Ba 2014) with a learning rate of 0.1 and the default values for other hyperparameters. Holding the cosmological parameters fixed, we can optimize the real white noise modes to source initial particles to evolve into our target pattern, in hundreds to thousands of iterations. Interestingly, we find that the variance of the modes becomes bigger than 1 (that of the standard normal white noise) after the optimization, and the optimized modes show some level of spatial correlation not present in the white noise fields, suggesting that the optimized initial conditions are probably no longer Gaussian.

4.3. Performance

pmwd benefits from GPU accelerations and efficient CUDA implementations of scatter and gather operations. In Figure 5, we present a performance test of pmwd, and find both the LPT and the *N*-body parts scale well except for very small problems. The growth function solution has a constant cost, and generally does not affect problems of moderate sizes. However, for a small number of particles and few time steps, it can dominate the computation, in which case one can accelerate the growth computation with an emulator (Kwan et al. 2022).

5. Conclusions

In this work, we develop the adjoint method for memory-efficient differentiable cosmological simulations, exploiting the reversible nature of *N*-body Hamiltonian dynamics, and

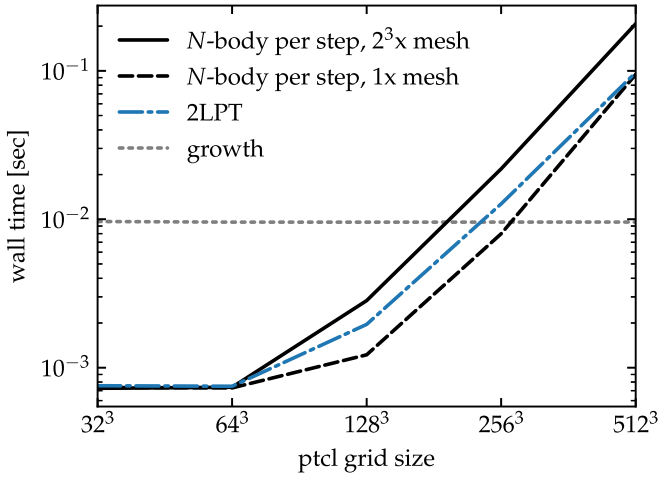


Figure 5. Performance of pmwd. Both the 2LPT and *N*-body components scale well from 512^3 particles to 64^3 particles, below which they become overhead dominated. Solving the growth ODEs takes a constant time, and can dominate the cost for small numbers of particles and few time steps, but generally does not affect problems with more than 128^3 particles.

implement it with `JAX` in a new PM library, pmwd. We have validated the numerical reversibility and the accuracy of the adjoint gradients.

pmwd is both computation and memory efficient, enabling larger and more accurate cosmological dark matter simulations. The next step involves modeling cosmological observables such as galaxies. One can achieve this with analytic, semianalytic, and deep learning components running based on or in parallel with pmwd. In the future, it can also facilitate the simultaneous modeling of multiple observables and the understanding of the astrophysics at play.

pmwd will benefit all the forward-modeling approaches in cosmology, and will improve gradient-based optimization and field-level inference to simultaneously constrain the cosmological parameters and the initial conditions of the Universe. The efficiency of pmwd also makes it a promising route to generate the large amount of training data needed by the likelihood-free inference frameworks (Alsing et al. 2019; Cranmer et al. 2020).

Currently, these applications require more development, including distributed parallel capability on multiple GPUs (Modi et al. 2021), more accurate time integration beyond `FastPM` (List & Hahn 2023), optimization of spatiotemporal resolution of the PM solvers (Dai & Seljak 2021; Lanzieri et al. 2022; Y. Zhang et al. 2023, in preparation), short-range supplement by direct summation of the particle–particle forces on GPUs (Habib et al. 2016; Potter et al. 2017; Garrison et al. 2021), differentiable models for observables, etc. We plan to pursue these in the future.

Acknowledgments

Y.L. is grateful to all organizers of the 2013 KICP Summer School on Computational Cosmology,[15] especially Andrey Kravtsov and Nick Gnedin. We thank Alex Barnett, Biwei Dai, Boris Leistedt, ChangHoon Hahn, Daisuke Nagai, Dan Foreman-Mackey, Dan Fortunato, David Duvenaud, David Spergel, Dhairya Malhotra, Erik Thiede, Francisco Villaescusa-Navarro, Fruzsina Agocs, Giulio Fabbian, Jens Stücker, Kaze Wong, Kazuyuki Akitsu, Lehman Garrison, Manas Rachh, Raúl Angulo, Shirley Ho, Simon White, Teppei Okumura, Tomomi Sunayama, Ulrich Steinwandel, Uroš Seljak, Wen Yan, Wenda Zhou, William Coulton, Xiangchong Li, and Yueying Ni for interesting discussions. Y.L. and Y.Z. were supported by The Major Key Project of PCL. Y.Z. was supported by the China Postdoctoral Science Foundation under award number 2023M731831. The Flatiron Institute is supported by the Simons Foundation.

Code Availability: pmwd is open-source on GitHub at https://github.com/eelregit/pmwd, including the source files and scripts of this paper at https://github.com/eelregit/pmwd/tree/master/docs/papers/adjoint. Also see the companion code paper (Li et al. 2022): https://github.com/eelregit/pmwd/tree/master/docs/papers/pmwd.

Software: `JAX` (https://github.com/google/jax), `NumPy` (Harris et al. 2020), `SciPy` (Virtanen et al. 2020), and `matplotlib` (Hunter 2007).

Appendix
Growth Equations

The 2LPT growth functions follow the following ODEs:

$$D_1'' + \left(2 + \frac{H'}{H}\right) D_1' - \frac{3}{2}\Omega_m(a) D_1 = 0,$$
$$D_2'' + \left(2 + \frac{H'}{H}\right) D_2' - \frac{3}{2}\Omega_m(a) D_2 = \frac{3}{2}\Omega_m(a) D_1^2, \quad \text{(A1)}$$

where

$$\Omega_m(a) = \frac{\Omega_m H_0^2}{a^3 H^2}. \quad \text{(A2)}$$

If the universe has been sufficiently matter dominated at a_i, with $\Omega_m(a_\mathrm{i}) \simeq 1$ and $H'/H \simeq -3/2$, the initial conditions of the ODEs can be set as the growing mode in this limiting case

$$D_1(a_\mathrm{i}) = a_\mathrm{i},$$
$$D_1'(a_\mathrm{i}) = a_\mathrm{i},$$
$$D_2(a_\mathrm{i}) = \frac{3}{7} a_\mathrm{i}^2,$$
$$D_2'(a_\mathrm{i}) = \frac{6}{7} a_\mathrm{i}^2. \quad \text{(A3)}$$

$D_1 \simeq a$ in the matter-dominated era, before being suppressed by dark energy.

The above growth equations can be written in such suppression factors,

$$G_m \triangleq D_m / a^m, \quad \text{(A4)}$$

for $m \in \{1, 2\}$, as

$$G_1'' + \left(4 + \frac{H'}{H}\right) G_1' + \left(3 + \frac{H'}{H} - \frac{3}{2}\Omega_m(a)\right) G_1 = 0,$$
$$G_2'' + \left(6 + \frac{H'}{H}\right) G_2' + \left(8 + 2\frac{H'}{H} - \frac{3}{2}\Omega_m(a)\right) G_2 = \frac{3}{2}\Omega_m(a) G_1^2, \quad \text{(A5)}$$

[15] https://kicp-workshops.uchicago.edu/compcosmology2013/

with initial conditions

$$G_1(a_i) = 1,$$
$$G_1'(a_i) = 0,$$
$$G_2(a_i) = \frac{3}{7},$$
$$G_2'(a_i) = 0. \tag{A6}$$

We solve the growth equations in G_m instead of D_m, with the JAX adaptive ODE integrator implementing the adjoint method in the optimize-then-discretize approach. This is because the former can be integrated backward in time more accurately for early times, which can improve the adjoint gradients.

We can then evaluate the FastPM time integration factors in Equation (14) by

$$G_D = a^2\dot{D}_1 = a^2 H D_1',$$
$$G_K = a\dot{G}_D = a^3 H^2\left[D_1'' + \left(2 + \frac{H'}{H}\right)D_1'\right], \tag{A7}$$

and

$$D_m = a^m G_m$$
$$D_m' = a^m(mG_m + G_1'),$$
$$D_m'' = a^m(m^2 G_1 + 2mG_1' + G_1''). \tag{A8}$$

ORCID iDs

Yin Li (李寅) https://orcid.org/0000-0002-0701-1410
Chirag Modi https://orcid.org/0000-0002-1670-2248
Drew Jamieson https://orcid.org/0000-0001-5044-7204
Yucheng Zhang (张宇澄) https://orcid.org/0000-0002-9300-2632
Libin Lu (陆利彬) https://orcid.org/0000-0003-0745-9431
Yu Feng (冯雨) https://orcid.org/0000-0001-5590-0581
François Lanusse https://orcid.org/0000-0001-7956-0542
Leslie Greengard https://orcid.org/0000-0003-2895-8715

References

Alsing, J., Charnock, T., Feeney, S., & Wandelt, B. 2019, MNRAS, 488, 4440
Angulo, R. E., & Hahn, O. 2022, LRCA, 8, 1
Blas, D., Lesgourgues, J., & Tram, T. 2011, JCAP, 2011, 034
Böhm, V., Feng, Y., Lee, M. E., et al. 2021, A&C, 36, 100490
Bouchet, F. R., Colombi, S., Hivon, E., et al. 1995, A&A, 296, 575
Chen, R. T. Q., Rubanova, Y., Bettencourt, J., et al. 2018, Advances in Neural Information Processing Systems 31, ed. S. Bengio et al. (NeurIPS), 6571, http://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf
Cranmer, K., Brehmer, J., & Louppe, G. 2020, PNAS, 117, 30055
Dai, B., & Seljak, U. 2021, PNAS, 118, e2020324118
Eisenstein, D. J., & Hu, W. 1998, ApJ, 496, 605
Feng, Y., Chu, M.-Y., Seljak, U., et al. 2016, MNRAS, 463, 2273
Garrison, L. H., Eisenstein, D. J., Ferrer, D., et al. 2021, MNRAS, 508, 575
Gholaminejad, A., Keutzer, K., & Biros, G. 2019, in Proc. of the 28th Int. Joint Conf. on Artificial Intelligence, IJCAI-19, ed. S. Kraus (Internatational Joint Conferences on Artificial Intelligence Organization), 730
Griewank, A., & Walther, A. 2000, ACM Trans. Math. Softw., 26, 19
Habib, S., Pope, A., Finkel, H., et al. 2016, NewA, 42, 49
Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Natur, 585, 357
Hockney, R. W., & Eastwood, J. W. 1988, Computer Simulation Using Particles (New York: Taylor & Francis)
Hunter, J. D. 2007, CSE, 9, 90
Jasche, J., & Lavaux, G. 2019, A&A, 625, A64
Jasche, J., & Wandelt, B. D. 2013, MNRAS, 432, 894
Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
Kwan, N. P., Modi, C., Li, Y., et al. 2022, arXiv:2211.06564
Lanzieri, D., Lanusse, F., & Starck, J.-L. 2022, Proc. 39th Int. Conf. on Machine Learning (ICML 2022), 60
Lewis, A., & Challinor, A., 2011 CAMB: Code for Anisotropies in the Microwave Background, Astrophysics Source Code Library, ascl:1102.026
Li, Y., Lu, L., Modi, C., et al. 2022, arXiv:2211.09958
List, F., & Hahn, O. 2023, arXiv:2301.09655
Michaux, M., Hahn, O., Rampf, C., et al. 2021, MNRAS, 500, 663
Modi, C., Lanusse, F., & Seljak, U. 2021, A&C, 37, 100505
Pontryagin, L. S. 1962, The Mathematical Theory of Optimal Processes (Boca Raton, FL: CRC Press)
Potter, D., Stadel, J., & Teyssier, R. 2017, ComAC, 4, 2
Quinn, T., Katz, N., Stadel, J., et al. 1997, arXiv:astro-ph/9710043
Seljak, U., Aslanyan, G., Feng, Y., et al. 2017, JCAP, 2017, 009
Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, NatMe, 17, 261
Wang, H., Mo, H. J., Yang, X., et al. 2014, ApJ, 794, 94