COmoving Computer Acceleration (COCA): N-body simulations in an emulated frame of reference

Deaglan J. Bartlett, Marco Chiarenza, Ludvig Doeser, Florent Leclercq

A&A, 694, A287 (2025)
https://doi.org/10.1051/0004-6361/202452217

Astronomy & Astrophysics

COmoving Computer Acceleration (COCA): *N*-body simulations in an emulated frame of reference

Deaglan J. Bartlett[1,⋆], Marco Chiarenza[1,2], Ludvig Doeser[3], and Florent Leclercq[1,⋆]

[1] CNRS & Sorbonne Université, UMR 7095, Institut d'Astrophysique de Paris, 98 bis boulevard Arago, F-75014 Paris, France
[2] Dipartimento di Fisica "Aldo Pontremoli", Università degli Studi di Milano, Via Celoria 16, 20133 Milano, Italy
[3] The Oskar Klein Centre, Department of Physics, Stockholm University, Albanova University Center, SE 106 91 Stockholm, Sweden

Received 12 September 2024 / Accepted 20 January 2025

ABSTRACT

Context. *N*-body simulations are computationally expensive and machine learning (ML) based emulation techniques have thus emerged as a way to increase their speed. Surrogate models are indeed fast, however, they are limited in terms of their trustworthiness due to potentially substantial emulation errors that current approaches are not equipped to correct.
Aims. To alleviate this problem, we have introduced COmoving Computer Acceleration (COCA), a hybrid framework interfacing ML algorithm with an *N*-body simulator. The correct physical equations of motion are solved in an emulated frame of reference, so that any emulation error is corrected by design. Thus, we are able to find a solution for the perturbation of particle trajectories around the ML solution. This approach is computationally cheaper than obtaining the full solution and it is guaranteed to converge to the truth as the number of force evaluations is increased.
Methods. Even though it is applicable to any ML algorithm and *N*-body simulator, we assessed this approach in the particular case of particle-mesh (PM) cosmological simulations in a frame of reference predicted by a convolutional neural network. In such cases, the time dependence is encoded as an additional input parameter to the network.
Results. We find that COCA efficiently reduces emulation errors in particle trajectories, requiring far fewer force evaluations than running the corresponding simulation without ML. As a consequence, we were able to obtain accurate final density and velocity fields for a reduced computational budget. We demonstrate that this method exhibits robustness when applied to examples outside the range of the training data. When compared to the direct emulation of the Lagrangian displacement field using the same training resources, COCA's ability to correct emulation errors results in more accurate predictions.
Conclusions. Therefore, COCA makes *N*-body simulations cheaper by skipping unnecessary force evaluations, while still solving the correct equations of motion and correcting for emulation errors made by ML.

Key words. gravitation – methods: numerical – cosmology: theory – large-scale structure of Universe

1. Introduction

N-body simulations represent a state-of-the-art numerical method for studying the dynamics of complex systems, including non-linear gravitational structure formation in the Universe (Vogelsberger et al. 2020; Angulo & Hahn 2022). Such simulations can be incredibly computationally expensive to run (e.g. Potter et al. 2017; Heitmann et al. 2019; Ishiyama et al. 2021; Frontiere et al. 2022; Wang et al. 2022); hence, various approaches based on ML have been proposed to either remove the requirement to run physical simulations or to reduce the complexity of the simulator used.

The most straightforward application of ML methods is as surrogate models, which take the initial conditions as inputs and emulate various features of the corresponding full *N*-body simulation as outputs. For example, Lucie-Smith et al. (2018) were able to predict halo properties from the initial conditions given certain environmental properties. Perraudin et al. (2019) used generative adversarial networks (GANs) to generate visually plausible three-dimensional (3D) cosmological density fields but encountered difficulties in reproducing the correct statistical distribution of physical density fields. Going further, Conceição et al. (2024) built an emulator of cosmological density fields based on a combination of dimensionality reduction via principal component analysis and supervised ML. He et al. (2019) demonstrated that it is possible to replicate the full result of particle-mesh (PM) simulations (i.e. a Lagrangian displacement field) using a deep neural network, which takes the Zel'dovich approximation (Zel'dovich 1970) displacement field as input. This work was extended to tree-based *N*-body simulations by Alves de Oliveira et al. (2020). Such emulators can replicate the power spectrum to the percent level up to $k \approx 1\,h\,\mathrm{Mpc}^{-1}$. Jamieson et al. (2023b) further extended these works by predicting both the displacement and velocity fields through two separate neural networks and by incorporating the cosmological matter density information through the addition of a 'style' parameter (Karras et al. 2020, see Sect. 3.3). The resulting emulator can reproduce power spectra and bispectra to within a few percent and achieves a similar level of cross-correlation with the true simulation run with the same initial conditions. By adding a time variable as an additional style parameter, Jamieson et al. (2024) were able to eliminate the need for two separate networks and produce an emulator capable of predicting *N*-body outputs as a function of redshift. The speed and differentiable nature of particle-based emulators enable them to be integrated within field-based inference of initial conditions (Doeser et al. 2024), with posterior re-simulations indicating a faithful reconstruction of the initial conditions. Simulations

⋆ Corresponding authors; deaglan.bartlett@iap.fr; florent.leclercq@iap.fr

including the effects of massive neutrinos (Giusarma et al. 2023) or modified gravity (Saadeh et al. 2025) can also be emulated using similar neural network techniques.

Instead of completely bypassing the N-body simulation, we can include ML corrections that capture unresolved physics in low-resolution, cheaper simulations. For example, Lanzieri et al. (2022) introduced an additional effective force to PM simulations to capture unresolved forces between particles. Their ML isotropic Fourier filter was extended by Payot et al. (2023) to depend not only on time and wavenumber but also on cosmological parameters. Super-resolution techniques based on GANs (Kodi Ramanah et al. 2020; Li et al. 2021) and U-nets (Zhang et al. 2024) have also been proposed, achieving power spectra correct within a few percent, as well as reasonable bispectra, void size functions, and halo mass functions correct to within 10% for halos down to $\approx 10^{11}\,\mathrm{M}_\odot$ in mass. Given the computationally demanding nature of hydrodynamical simulations, Dai & Seljak (2021) introduced a light model (with only $O(10)$ learnable parameters) to transform the output of a dark matter-only simulation to one that resembles the hydrodynamical simulation run with the same initial conditions. Ding et al. (2024) also presented a light and interpretable neural network to produce halo catalogues from dark matter density fields.

Accuracy and interpretability are pivotal challenges in the application of ML to N-body simulations. Despite the high reported accuracy of the methods reviewed above on various tests (mainly using summary statistics), none of these models can be expected to perfectly recover the truth. Are ML-accelerated simulation algorithms sufficiently accurate to be used in real-world applications? Without a ground-truth model to compare against during actual use (since such algorithms are designed to eliminate the need for it), current approaches have limited means of identifying the emulation error and cannot correct for it. Since typical simulations usually also involve simplifying assumptions and approximations, perfectly accurate ML-based models may not be required for many purposes. The question that arises is then that of the interpretability of ML, in order to control the approximation made with respect to a physical simulator. Unfortunately, many ML algorithms, including (deep) neural networks, lack interpretability. If machines predict something humans do not understand, it is essential to be able to check (and trust) the results.

In this paper, we contend that addressing the lack of interpretability of ML is not always necessary to use an emulator of an expensive model while maintaining control over the degree of accuracy. We elucidated this argument by constructing a framework in which emulation of N-body simulations is made an ML-safe task by physically rectifying emulation inaccuracies. By 'ML-safe', we mean systems that are reliable, robust, and trustworthy by construction. The key idea is to find a mathematically equivalent form of the system's equations of motion, where we solve for the (not necessarily small) perturbation around the approximate solution provided by ML. From a physical point of view, in N-body simulations obeying Newtonian dynamics, this is equivalent to solving the equation of motion in an emulated frame of reference. Since the ML solution is designed to be approximately correct, computing any corrections is numerically easier than evolving the full system, thus requiring fewer evaluations of the forces. Through the number and the temporal positions of force evaluations, the user controls the trade-off between speed and accuracy, ranging from fully trusting the ML solution by never correcting particle trajectories to correcting for ML emulation errors at any time step of the simulation. The system has the theoretical guarantee of asymptotically converging to the physical solution as the number of force evaluations increases.

For gravitational N-body simulations of dark matter particles, we introduce the COmoving Computer Acceleration (COCA) approach to running cosmological simulations within an emulated frame of reference. While traditional emulators aim to translate initial conditions into final particle positions, directly representing the non-linear dark matter distribution, COCA aims to emulate a frame of reference in which to run a physical simulation with lower computational cost. The approach can be seen as a generalisation and improvement of the idea behind COmoving Lagrangian Acceleration (COLA) (Tassev et al. 2013). As an illustration, we compare the results of COLA and COCA simulations when forces are evaluated through a particle-mesh (PM) scheme. The current implementation can be interpreted either as a means of making COLA more computationally efficient (from a simulator's perspective) or as a method for correcting emulators of PM-based simulations (from an emulator builder's perspective). We find that our ML-enhanced approach requires very few force evaluations to correct for emulation errors: it achieves percent-level accurate power spectra, bispectra, and cross-correlation to a reference simulation, with approximately 8 force evaluations, compared to 20 for COLA. Importantly, the COCA formalism is not limited to PM codes. Future iterations of this work will extend the framework to other gravity solvers (e.g. $\mathrm{P^3M}$ or tree-based methods), with the ultimate goal of improving emulators of these more precise codes.

This paper is organised as follows. In Sect. 2, we review the COLA approach to N-body simulations, extend it to yield COCA, and describe the benefits of COCA in terms of computational efficiency. A more thorough description is provided in Appendix A. We introduce our emulator for the frame of reference in Sect. 3 and describe the training procedure and validation metrics for the COCA simulations. In Sect. 4, we present our results: the performance of the emulator, the accuracy of COCA simulations as a function of the number of force evaluations, the generalisation to an example known to be outside the range of the training set, the comparison to a Lagrangian displacement field emulator, and a discussion of the computational performance. We present our conclusions in Sect. 5, along with a discussion of potential future extensions and applications of this study.

2. Theory

For simplicity, some of the equations in this section are abridged. We reintroduce the omitted constants, temporal prefactors, and Hubble expansion in Appendix A.

2.1. Review of COLA

In a cosmological dark matter-only N-body code, we compute the final Eulerian positions of particles, $\mathbf{x}$, as a function of scale factor, a, as they interact under gravity. If the initial comoving particle positions are $\mathbf{q}$, then the Lagrangian displacement field is given by (e.g. Bernardeau et al. 2002, for a review):

$$\mathbf{\Psi}(\mathbf{q}, a) \equiv \mathbf{x}(a) - \mathbf{q}. \tag{1}$$

We must then solve the equation of motion, which is schematically expressed as

$$\partial_a^2 \mathbf{\Psi}(\mathbf{q}, a) = -\boldsymbol{\nabla}\Phi(\mathbf{x}, a), \tag{2}$$

where the gravitational potential Φ satisfies the Poisson equation sourced by the density contrast field $\delta(\mathbf{x}, a)$,

$$\Delta\Phi(\mathbf{x}, a) = \delta(\mathbf{x}, a). \tag{3}$$

In the perturbative regime, analytic solutions for $\boldsymbol{\Psi}(\mathbf{q}, a)$ can be derived, which are known as Lagrangian perturbation theory (LPT, Zel'dovich 1970; Buchert 1989; Bouchet et al. 1995; Bernardeau et al. 2002). These solutions are valid on large scales but become inaccurate once shell crossing occurs, making the approximation more reliable at early times. This behaviour is illustrated in Fig. 1a, where initially the LPT trajectories and the true trajectories are indistinguishable, but the discrepancy increases over time.

The temporal COmoving Lagrangian Acceleration (Tassev et al. 2013, tCOLA or simply COLA in the following) algorithm aims to separate the temporal evolution of large and small scales by evolving large scales using analytic LPT results and small scales using a numerical solver. This is accomplished by decomposing the Lagrangian displacement field into two components (Tassev & Zaldarriaga 2012):

$$\boldsymbol{\Psi}(\mathbf{q}, a) \equiv \boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a) + \boldsymbol{\Psi}^{\rm COLA}_{\rm res}(\mathbf{q}, a), \tag{4}$$

where $\boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a)$ represents the LPT displacement field, and $\boldsymbol{\Psi}^{\rm COLA}_{\rm res}(\mathbf{q}, a)$ denotes the residual displacement of each particle as observed from a frame comoving with an 'LPT observer', whose trajectory is defined by $\boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a)$. Knowing $\boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a)$, we do not need to solve for the full trajectory of the particle, but just the residual between the approximation and the truth (the blue arrows in Fig. 1a).

Using Eq. (4), it is possible to rewrite Eq. (2) as

$$\partial_a^2 \boldsymbol{\Psi}^{\rm COLA}_{\rm res}(\mathbf{q}, a) = -\boldsymbol{\nabla}\Phi(\mathbf{x}, a) - \partial_a^2 \boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a). \tag{5}$$

Therefore, we can view LPT as providing a new frame of reference within which we solve the equations of motion. The term $\partial_a^2 \boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a)$ can be thought of as a fictitious force acting on particles, caused by our use of a non-inertial frame of reference.

Since particles experience lower typical accelerations in the LPT frame compared to the natural cosmological frame, solving the equation of motion numerically becomes a comparatively simpler task that requires fewer time steps to achieve equivalent accuracy (Tassev et al. 2013; Howlett et al. 2015; Leclercq et al. 2015; Koda et al. 2016; Izard et al. 2016). In particular, COLA has been demonstrated to always yield correct results at large scales, even with a small number of time steps ($\leq$10), unlike a basic particle-mesh (PM) code. Given that Eq. (5) is mathematically equivalent to Eq. (2), COLA is asymptotically equivalent to the corresponding standard N-body code (e.g. a PM code if forces $-\boldsymbol{\nabla}\left(\Delta^{-1}\delta\right)$ are evaluated via a standard PM technique), in the limit of an infinite number of time steps.

2.2. COCA formalism

While the COLA formalism has proven effective in solving the equation of motion within the frame of reference defined by LPT, there is no requirement to adhere to LPT or any other analytic approximation in the decomposition given by Eq. (4). According to the principle of Galilean invariance, the equation of motion can be solved in any frame of reference, provided appropriate fictitious forces are introduced for non-inertial frames. Considering that the simplest scenario is the one where no motion occurs, we aim at finding a frame of reference where particles are nearly stationary. In such a frame of reference, solving the equation of motion numerically to reach a given level of accuracy becomes an easier problem than in COLA. This is the key insight that underpins the formalism proposed in this paper. We have named this approach COmoving Computer Acceleration (COCA).

We propose utilising a ML algorithm, such as a neural network, as an emulator to learn and predict the trajectories of particles in N-body simulations. Since the LPT frame already provides a good approximation of the trajectories, particularly on large scales, we opt to learn displacements relative to the LPT frame. Therefore, the emulator outputs a displacement field $\boldsymbol{\Psi}_{\rm ML}(\mathbf{q}, a)$ that approximates $\boldsymbol{\Psi}(\mathbf{q}, a) - \boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a)$.

Rather than directly employing the emulator as a surrogate for simulation results, we use the frame of reference corresponding to the emulated trajectories in order to run a simulation. Following the same spirit as COLA, we split the Lagrangian displacement field into three contributions,

$$\boldsymbol{\Psi}(\mathbf{q}, a) \equiv \boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a) + \boldsymbol{\Psi}_{\rm ML}(\mathbf{q}, a) + \boldsymbol{\Psi}^{\rm COCA}_{\rm res}(\mathbf{q}, a), \tag{6}$$

where $\boldsymbol{\Psi}_{\rm ML}(\mathbf{q}, a)$ is the ML contribution to the Lagrangian displacement field, and the residual displacement $\boldsymbol{\Psi}^{\rm COCA}_{\rm res}(\mathbf{q}, a)$ represents the emulation error. Different contributions are shown schematically in Fig. 1b.

Reframing Eq. (2) using Eq. (6), the equation of motion for COCA contains an extra fictitious force with respect to COLA:

$$\partial_a^2 \boldsymbol{\Psi}^{\rm COCA}_{\rm res}(\mathbf{q}, a) = -\boldsymbol{\nabla}\Phi(\mathbf{x}, a) - \partial_a^2 \boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a) - \partial_a^2 \boldsymbol{\Psi}_{\rm ML}(\mathbf{q}, a). \tag{7}$$

In COCA, the predicted displacement $\boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a) + \boldsymbol{\Psi}_{\rm ML}(\mathbf{q}, a)$ approximates the optimal frame of reference in which to solve the simulation (the one where all particles are at rest). Ideally, in case of perfect emulation, solving the equation of motion would result in no trajectory adjustment ($\boldsymbol{\Psi}^{\rm COCA}_{\rm res}(\mathbf{q}, a) = 0$ for any a). Otherwise, numerically solving Eq. (7) corrects the trajectories of particles to produce a more accurate solution.

We describe the COCA formalism in greater detail in Appendix A. Notably, although we describe the framework in terms of an emulated displacement field, $\boldsymbol{\Psi}_{\rm ML}(\mathbf{q}, a)$, we also show that we can equivalently define the new frame of reference by the momentum $\mathbf{p} \equiv \mathrm{d}\mathbf{x}/\mathrm{d}a$ of particles, namely,

$$\mathbf{p}(a) \equiv \mathbf{p}_{\rm LPT}(a) + \mathbf{p}_{\rm ML}(a) + \mathbf{p}_{\rm res}(a), \tag{8}$$

where $\mathbf{p}_{\rm LPT}(a)$ and $\mathbf{p}_{\rm ML}(a)$ denote momenta predicted by LPT and ML, respectively, and the residual momentum $\mathbf{p}_{\rm res}(a)$ is determined by solving the equations of motion.

2.3. Reducing the number of force evaluations

To integrate the equations of motion in the new frame of reference, we utilise a symplectic 'kick-drift-kick' (leapfrog) algorithm (e.g. Birdsall & Langdon 1985). With this method, the positions, $\mathbf{x}$, and momenta, $\mathbf{p}$, of the particles are updated at different times, typically with one momentum update between every two position updates. A schematic illustration of the technique is given in Fig. 2, with the full details provided in Appendix A.

At each momentum update ('kick'), we face two choices. We can assume that the emulated frame of reference is sufficiently accurate and thus update the particle momenta by simply evaluating the emulator, corresponding to following the 'emulated' (purple) trajectory in Fig. 1b (equivalent to assuming that $\mathbf{g}_\delta(t^{\rm D}) = \mathbf{0}$ in Eq. (A.25), using the notations of Appendix A). Alternatively, we may deem the emulation error significant and opt to correct the trajectory, aiming to bring the particles back to the "N-body" (black) trajectory in Fig. 1b. This

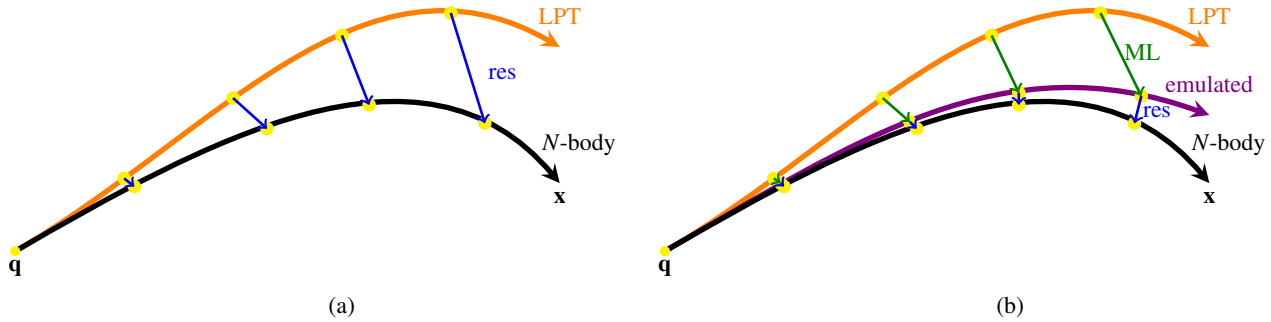

Fig. 1. Schematic illustration of the (a) COLA and (b) COCA formalism for cosmological simulations. In COLA, we solve the equations of motion in the frame of reference given by LPT, so we compute the residual ('res') between the LPT trajectory and the true position, **x**, of particles. In COCA, we emulate a frame of reference closer to the true trajectory by adding a ML contribution to LPT. Thus, we can solve for the (smaller) residuals between **x** and the emulated frame.

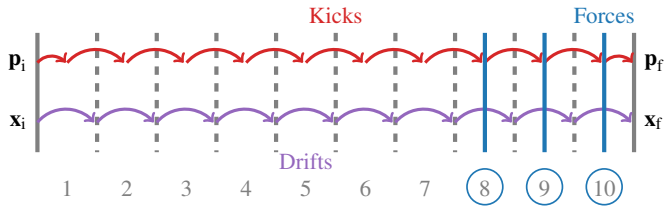

Fig. 2. Schematic illustration of the kick-drift-kick integration scheme employed in this study. The initial positions $\mathbf{x}_i$ and momenta $\mathbf{p}_i$ are evolved to their final values, $\mathbf{x}_f$ and $\mathbf{p}_f$, with updates to these quantities occurring at different times. Unlike traditional kick-drift-kick integration, we choose not to evaluate forces that appear in the equations of motion at all time steps, but only at a subset (steps 8, 9 and 10 in this example). At all other kick steps, the momenta are updated according to the emulated frame of reference only.

correction involves evaluating gravitational forces between particles[1] ($\mathbf{g}_\delta(t^D)$ in Appendix A) and using the complete form of the kick operator. Correcting trajectories is more computationally expensive than simply following the emulated trajectories, so the number of force evaluations, n_f, should be as small as possible, but large enough to correct for emulation errors. During time steps without force evaluations, particles move according to trajectories defined by their respective frames of reference ($\mathbf{\Psi}_{\rm LPT}$ for COLA and $\mathbf{\Psi}_{\rm LPT}+\mathbf{\Psi}_{\rm ML}$ for COCA). Hence, $n_f = 0$ corresponds to the LPT solution in COLA simulations and a purely emulated one in COCA simulations.

In COCA, the ability to reduce the number of force evaluations introduces an additional degree of freedom compared to PM/COLA simulations, where forces are evaluated at every time step. Force evaluations can, in principle, be placed at any of the time steps, however, we have found that concentrating all evaluations towards the end of the simulation, when structure formation is non-linear, typically yields the most accurate results. Up until the first force evaluation, the COCA framework consists of predicting particle positions, **x**, and momenta, **p**, at specific times, functioning in a similar way as more traditional emulators (e.g. Jamieson et al. 2023b). In Fig. 2, we show an example of a kick-drift-kick scheme with ten time steps, with three force evaluations at time steps 8, 9, and 10. At all other time steps, momentum updates (kicks) rely solely on the chosen frame of reference.

3. Emulation

We recall that the field to be emulated, $\mathbf{p}_{\rm ML}(\mathbf{q}, a)$, is the residual momentum field with respect to the LPT momentum field, namely,

$$\mathbf{p}(\mathbf{q}, a) - \mathbf{p}_{\rm LPT}(\mathbf{q}, a), \tag{9}$$

at any value of a corresponding to a kick time step (see Appendix A).

3.1. Training data

For the application of COCA described in this work, we chose to emulate the frame of reference in a cubic box of length $128\,h^{-1}$ Mpc with $N^3 = 64^3$ dark matter particles, resulting in a final density field at $a = 1$ on a grid with a resolution of $\Delta x = 2\,h^{-1}$ Mpc. This resolution is approximately the same as that used by Jamieson et al. (2023b)[2]. Since the focus of this paper is the time evolution of the fields, we adopt fixed cosmological parameters equal to the best-fit values (TT,TE,EE+lowE+lensing+BAO) from Planck 2018 (Planck Collaboration VI 2020): $\Omega_b = 0.0490$, $\Omega_m = 0.3111$, $h = 0.6766$, $\tau = 0.0561$, $n_s = 0.9665$, and $\sigma_8 = 0.8102$. We assume a flat Universe and a non-evolving equation of state for dark energy.

Although the COCA formalism can be applied to any method of computing the forces between particles (PM, P^{3}M, tree-based, etc.), for this paper, we chose to work with a PM force solver, utilising a modified version of the publicly available SIMBELMYNË code[3] (Leclercq et al. 2015, 2020). For our simulations, we generated initial conditions at a scale factor of $a = 0.05$ using second-order LPT and solved the equations of motion using COLA with 20 time steps equally spaced in a and a PM grid of size 64^3 (see Izard et al. 2016; Koda et al. 2016, for investigations on the effect of these choices in COLA simulations). Although we have verified that this initial scale factor and number of time steps are appropriate to give converged results for all $k \leq 1\,h\,{\rm Mpc}^{-1}$, the 'reference' against which we

[1] In a PM scheme, which we employ in this paper, forces are computed by deriving the density field from particle positions through cloud-in-cell binning, solving the Poisson equation in Fourier space to obtain the gravitational potential, and then finite differencing the potential in configuration space to get the forces.

[2] The number of force evaluations needed in COCA to achieve a given accuracy likely depends on the accuracy of the frame of reference emulator and, therefore, on the resolution. We leave the investigation of such effects to future work.

[3] https://simbelmyne.florent-leclercq.eu/

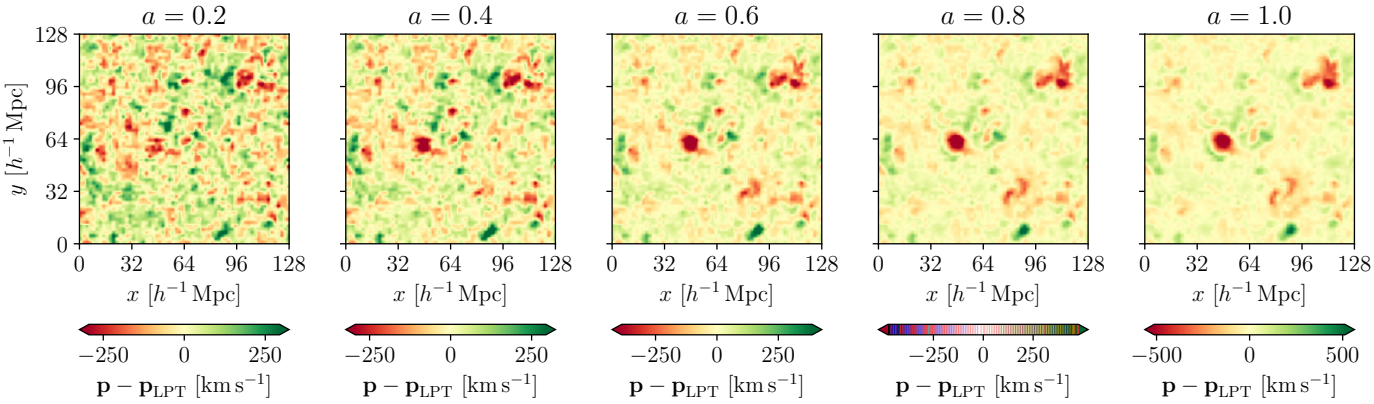

Fig. 3. Slice of the difference between the true momenta of particles $\mathbf{p}$ and the LPT prediction $\mathbf{p}_{\mathrm{LPT}}$, for a test simulation, as a function of scale factor a. We plot the component orthogonal to the chosen slice. At late times, the spatial structure of the field $\mathbf{p} - \mathbf{p}_{\mathrm{LPT}}$ remains relatively constant, with most of the time dependence being a simple multiplicative scaling.

made comparisons during testing refers to a COLA simulation with the same setup, except with 100 time steps equally spaced in a. At each time step of the simulations, we output the difference between the computed momentum of the particles $\mathbf{p}$ and the LPT momentum $\mathbf{p}_{\mathrm{LPT}}$, which is the quantity we must emulate.

We produce 100 simulations for training, 50 for validation, and a further 50 for testing. This is a sufficiently small number of training simulations that re-training with a different resolution or specifications does not require significant computational resources. While we could potentially achieve a higher accuracy for the emulator with more training simulations, the aim of this paper is primarily to demonstrate how to correct for emulation errors rather than to produce the optimal emulator. Therefore, we found 100 training simulations to be sufficient for our purposes. For each simulation, we use all 20 output snapshots, resulting in a total of 2000 fields for training. In addition, we used 1000 fields for validation and 1000 for testing.

3.2. Scaling of momenta

In Fig. 3, we plot one component of the field $\mathbf{p} - \mathbf{p}_{\mathrm{LPT}}$ as a function of scale factor, for a slice of one of our test simulations. From visual inspection, we find that the large-scale spatial structure of the field to be emulated does not change significantly as a function of time, particularly at late times, but its magnitude does. We therefore chose to rescale the momenta to be emulated by defining

$$\mathbf{p}_{\mathrm{ML}}(\mathbf{q}, a) \equiv D(a)\mathcal{H}(a)\varpi(a)\tilde{\mathbf{p}}_{\mathrm{ML}}(\mathbf{q}, a)\,, \tag{10}$$

where $\tilde{\mathbf{p}}_{\mathrm{ML}}$ is defined to have a standard deviation of unity, $D(a)$ is the linear growth factor, $\mathcal{H}(a)$ is the conformal Hubble parameter in units of h, and $\varpi(a)$ is a time-dependent function which we wish to approximate. Our emulator is designed to directly predict $\tilde{\mathbf{p}}_{\mathrm{ML}}$, and thus this scaling has the benefit of standardising the output, since $\tilde{\mathbf{p}}_{\mathrm{ML}}$ has zero mean and standard deviation unity.

To find an approximation for $\varpi(a)$, we compute the standard deviation of the 2000 training $\mathbf{p} - \mathbf{p}_{\mathrm{LPT}}$ fields and fit these as a function of a using the ESR (Bartlett et al. 2022) symbolic regression code. We use a mean squared error loss function and allow functions to be comprised of addition, multiplication, subtraction, division, the power operator, as well as free constants, θ, and the scale factor a. Upon inspecting the fitted equations, we find that a power law provides a sufficiently simple yet accurate

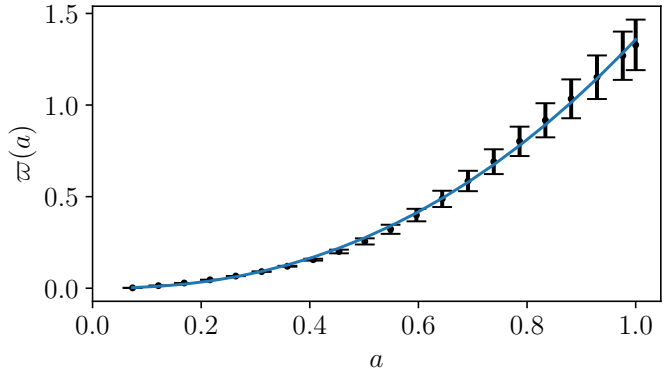

Fig. 4. Scaling of the residual momentum as a function of scale factor, as defined in Eq. (10). The points and error bars represent the mean and standard deviation, respectively, across the 100 training simulations. The curve represents the best fit, as given by Eq. (11).

approximation for our purposes:

$$\varpi(a) \approx (\theta_0 a)^{\theta_1}\,, \tag{11}$$

with parameters $\theta_0 = 1.1415174$ and $\theta_1 = 2.3103984$, which yields a root mean squared error of $1.5{\times}10^{-3}$. We compare this fit to the training data in Fig. 4, from which we see that it accurately reproduces $\varpi(a)$ at all scale factors. Note that any error in this fit can be compensated for by the emulator, so a perfect fit is not required.

3.3. Neural network architecture

To emulate $\tilde{\mathbf{p}}_{\mathrm{ML}}$, we utilised a U-net/V-net architecture (Ronneberger et al. 2015; Milletari et al. 2016), with a similar implementation to Alves de Oliveira et al. (2020), Jamieson et al. (2023b, 2024). Our model consists of three resolution levels connected in a 'V' shape, using two downsampling (by stride-2 2^3 convolutions) and two up-sampling (by stride-$^1/_2$ 2^3 convolutions) layers. At each level, we apply a 3^3 convolution and, as in a V-Net, we apply a 1^3 convolution as a residual connection (ResNet, He et al. 2016) within each block. A batch normalisation is applied after each convolution, which is followed by a leaky ReLU activation function with a negative slope of 0.01. Each layer has 64 channels, except the input (1), output (3), and those after concatenations (128).

For every convolutional layer, we introduce a "style" parameter[4] (borrowing the nomenclature from StyleGAN2, Karras et al. 2020), where each convolutional kernel is multiplied by an array of the same dimension as the layer's input and with values equal to the style parameter. Since we are producing a time-dependent emulator, we use the scale factor a as a style parameter (similar to Jamieson et al. 2024, who also include a redshift-dependent style parameter). Our network is implemented and trained using a modified version of the MAP2MAP[5] package and PYTORCH (Paszke et al. 2019).

The input to the emulator is the redshift-zero linear density field (at $a = 1$). This contrasts with He et al. (2019), Alves de Oliveira et al. (2020), Jamieson et al. (2023b), and Jamieson et al. (2024), who use the three displacement fields predicted by first-order LPT as input. Given that the latter is computed deterministically from the former, both fields contain the same amount of information. However, the linear density field requires three times less memory to store, making our approach more memory-efficient.

Compared to previous N-body emulators, our architecture is most similar to that of Jamieson et al. (2023b), although we used only $N^3 = 64^3$ voxels in the input field, whereas Jamieson et al. (2023b) employed $N^3 = 128^3$ voxels with a similar voxel size. This reduction in voxel count decreases memory requirements by an additional factor of 8 compared to Jamieson et al. (2023b). Furthermore, we require less padding of the input field than Jamieson et al. (2023b): our approach employs periodic padding of 24 voxels, compared to 48 in their implementation. The smaller input size (resulting from using a scalar field rather than a three-vector field) necessitates one less resolution layer in our neural network architecture, thus reducing the number of trainable parameters in our model to 2.4×10^6, compared to 3.4×10^6 in Jamieson et al. (2023b). The study of He et al. (2019) emulated the displacement field of particle-mesh simulations, using $N^3 = 32^3$ voxels with the same box size as in our work, which resulted in a voxel size that is twice as large. Despite their lower resolution, their architecture employs 8.4×10^6 parameters. It is worth noting that while our model has fewer parameters than Jamieson et al. (2023b), unlike their emulator, our network does not currently encode dependence on cosmological parameters. A similar number of parameters may be necessary once this feature is incorporated.

Regarding the dependence of the emulation on cosmology, we expect the sensitivity of $\mathbf{p} - \mathbf{p}_{\rm LPT}$ to cosmological parameters to be relatively small, since long-range features should be captured in $\mathbf{p}_{\rm LPT}$. Moreover, we choose to use the linear density field as input instead of the white noise field from which it is produced. This way, our emulator of $\mathbf{p} - \mathbf{p}_{\rm LPT}$ only depends on $\Omega_{\rm m}$, as the equations of motion depend solely on this parameter. The dependence on all other cosmological parameters is contained in the linear power spectrum, which is used to transform the white noise field into the linear density field. Thus, adding only $\Omega_{\rm m}$ as a second style parameter to the network would be sufficient to capture the dependence of the framework on cosmological parameters. For simplicity, we fix $\Omega_{\rm m}$ and save this extension for future work. Omitting $\Omega_{\rm m}$ as a second style parameter also enabled us to test the robustness of the COCA framework in the case of cosmological parameter mis-specification, hence, we were able to check for ML safety. We discuss this aspect in Sect. 4.3.

To summarise, our architecture is similar to that of Jamieson et al. (2023b), with three main differences: (i) we use a single channel (linear density) input rather than three channels (LPT displacements or velocities); (ii) we have three resolution levels instead of four (since we work with $N^3 = 64^3$ grids as opposed to $N^3 = 128^3$); and (iii) we include a as a style parameter (as in Jamieson et al. 2024) and fix $\Omega_{\rm m}$.

While it is possible to discuss similarities and differences in the neural network architectures, we cannot directly compare the output of our emulator with that of He et al. (2019), Jamieson et al. (2023b) or Jamieson et al. (2024). This is because these works emulate the Lagrangian displacement field $\mathbf{\Psi}$, whereas our emulator predicts the residual momentum $\tilde{\mathbf{p}}_{\rm ML}$, the quantity required by the COCA framework. We note that Jamieson et al. (2023b, 2024) employ a more accurate gravity solver in their training simulations compared to the PM model used by He et al. (2019) and in this work. However, this difference does not affect the respective neural network architectures.

3.4. Training

As our loss function, we chose

$$\text{Loss} \equiv \log L_1 + \log L_2, \tag{12}$$

where

$$L_n \equiv \sum_{\mathbf{q}} \sum_i \left\{ \left[(\mathbf{p}_{\rm LPT} + \mathbf{p}_{\rm ML})_i \right]^n - \left[(\mathbf{p}_{\rm true})_i \right]^n \right\}^2, \tag{13}$$

and the sum runs over the Lagrangian coordinates of the particles, $\mathbf{q}$, and the three Cartesian components, $i \in x, y, z$.

This functional form is partially inspired by Jamieson et al. (2023b). The L_1 term matches $\mathbf{p}_{\rm ML}$ to the residual momenta $\mathbf{p}_{\rm true} - \mathbf{p}_{\rm LPT}$, whereas the L_2 term ensures that the full momentum field (including the LPT contribution) matches $\mathbf{p}_{\rm true}$. Jamieson et al. (2023b) found that terms similar to L_2 are required to correctly predict redshift-space distortions. We leave the investigation of redshift-space distortions in COCA for future work. Both terms of our loss function use the mean square error between the fields in Lagrangian coordinates. Unlike Jamieson et al. (2023b), we do not include any term in Eulerian coordinates. Given the computational and memory requirements to use the displacement fields in Eulerian coordinates, and the good performance already achieved with our choice, we decided to omit such additional terms.

We use the Adam optimiser (Kingma & Ba 2014) with decoupled weight decay (AdamW) (Loshchilov & Hutter 2017), an initial learning rate of 1.5×10^{-4}, a weight decay coefficient of 8×10^{-3}, and parameters $\beta_1 = 0.85$, $\beta_2 = 0.994$, and $\epsilon = 3 \times 10^{-9}$. The learning rate is reduced on a plateau by a factor of 0.35 when the loss does not improve by more than 10^{-3} over 50 epochs. After a change in learning rate, we apply a cooldown of 30 epochs before the scheduler resumes normal operation. We use a batch size of 5 and train on a single V100 GPU, which has 32 GB of RAM. The entire time for generating the training, validation, and test simulations (for which we use 40 Intel Xeon Gold 6230 cores) and training was 120 hours, corresponding to 277 epochs, by which time the training and validation losses had plateaued.

[4] A style parameter in a neural network is an additional input at each layer that encodes dependence on an important feature. In our case, the scale factor a encodes the dependence on clustering at various cosmological times. For more details, we refer the interested reader to Eqs. (1)–(3) of Karras et al. (2020).

[5] https://github.com/eelregit/map2map/

3.5. Validation metrics

To quantitatively determine the accuracy of the COCA simulations, we compute the dark matter density field δ using a cloud-in-cell estimator (Hockney & Eastwood 1981) and the velocity field $\mathbf{v}$ using a simplex-in-cell estimator (Hahn et al. 2015; Leclercq et al. 2017). To work with a scalar field rather than a vector field, we compute the divergence of the velocity field in Fourier space, $\nabla \cdot \mathbf{v}$[6].

For both fields $\varphi \in \{\delta, \nabla \cdot \mathbf{v}\}$, we compute the (auto) power spectrum $P_\varphi(k)$ defined by

$$\langle \varphi(\mathbf{k})\, \varphi(\mathbf{k}') \rangle \equiv (2\pi)^3\, \delta_{\mathrm{D}}(\mathbf{k} + \mathbf{k}')\, P_\varphi(k), \tag{14}$$

where δ_{D} is a Dirac delta distribution. For all simulations, we compute the ratio of power spectra between the simulation of interest and the reference. We also compute the cross spectrum $P_{\varphi_a\varphi_b}(k)$ between the test simulation and the reference simulation, defined by

$$\langle \varphi_a(\mathbf{k})\, \varphi_b(\mathbf{k}') \rangle \equiv (2\pi)^3\, \delta_{\mathrm{D}}(\mathbf{k} + \mathbf{k}')\, P_{\varphi_a\varphi_b}(k). \tag{15}$$

Thus, we obtain the cross-correlation coefficient

$$r_{\varphi_a\varphi_b}(k) = \frac{P_{\varphi_a\varphi_b}(k)}{\sqrt{P_{\varphi_a}(k) P_{\varphi_b}(k)}}. \tag{16}$$

We could interpret $1 - r^2$ as the fraction of the variance in the prediction that is not explained by the reference. In schemes such as CARPOOL (Chartier et al. 2021), where exact and approximate simulations are combined, $1 - r^2$ is proportional to the required number of simulations. Hence, improving r^2 can dramatically reduce the required computational resources. Just comparing r can hide the importance of improving the cross-correlation: for example, improving r from 0.9 to 0.99 (a change of 0.09) corresponds to explaining an additional 17% of the variance at that scale. For these reasons, in all figures, we plot r^2 rather than r, since it is more meaningful. All two-point statistics are computed using SIMBELMYNË.

To assess higher-order statistics, we also compute the bispectrum $B(k_1, k_2, k_3)$ of the density field, defined by

$$\langle \delta(\mathbf{k}_1)\, \delta(\mathbf{k}_2)\, \delta(\mathbf{k}_3) \rangle \equiv (2\pi)^3\, \delta_{\mathrm{D}}\left(\sum_{i=1}^{3} \mathbf{k}_i \right) B(k_1, k_2, k_3), \tag{17}$$

and to factor out the dependence on scale and cosmological parameters, we have the reduced bispectrum,

$$Q(k_1, k_2, k_3) \equiv \frac{B(k_1, k_2, k_3)}{P_1 P_2 + P_2 P_3 + P_3 P_1}, \tag{18}$$

with $P_i \equiv P_\delta(k_i)$ for $i \in \{1, 2, 3\}$. We consider two different configurations in this work, which are designed to be approximately the same as those used in Jamieson et al. (2023b), Doeser et al. (2024). First, we consider a "squeezed" bispectrum, consisting of an isosceles triangle configuration with one small wavenumber, $k_\ell = 9.8 \times 10^{-2}\, h\,\mathrm{Mpc}^{-1}$, and two sides of equal but varying size, $k_1 = k_2 = k_{\mathrm{s}}$. For our second configuration, we fix two of the wavenumbers, $k_1 = 0.1\, h\,\mathrm{Mpc}^{-1}$ and $k_2 = 1.0\, h\,\mathrm{Mpc}^{-1}$, and vary the angle θ between them. All bispectrum calculations are performed using PYLIANS (Villaescusa-Navarro 2018).

[6] The velocity potential is usually of greater physical interest than the divergence of the velocity field. However, in Fourier space, they are related by a factor of $1/k^2$, and since we only compare the ratio of auto and cross spectra at a given k, all quantities shown are identical for both. Thus, we computed only the divergence.

4. Results

4.1. Emulator performance

In Fig. 5, we plot slices of the input δ_{linear}, output $\mathbf{p}_{\mathrm{ML}}$, target $\mathbf{p} - \mathbf{p}_{\mathrm{LPT}}$, and emulation error $\mathbf{p}_{\mathrm{res}} \equiv \mathbf{p} - \mathbf{p}_{\mathrm{LPT}} - \mathbf{p}_{\mathrm{ML}}$ for one of our test simulations, where all fields are evaluated at $a = 1$. The target fields are obtained by running COLA simulations with 20 time steps, using initial conditions matching those of the test simulations, and saving the residuals between the calculated and LPT momenta. As described in Sect. 3.3, the input is the linear density field, comprising a single channel, whereas the output prediction is a three-component vector for each Lagrangian grid point. Since we are learning the residuals between the true momentum and the LPT prediction, correlations observed in $\mathbf{p} - \mathbf{p}_{\mathrm{LPT}}$ are highly localised, reflecting the accurate capture of large-scale modes by LPT.

Visually, there is a notable correlation between $\mathbf{p} - \mathbf{p}_{\mathrm{LPT}}$ (second column of Fig. 5) and $\mathbf{p}_{\mathrm{ML}}$ (third column of Fig. 5). Leveraging the linear density field and scale factor information, the emulator accurately identifies the spatial structure of the $\mathbf{p}_{\mathrm{ML}}$ field. The small emulation errors indicate its capability to predict magnitudes as well. We observe that the emulation error is signal-dependent, resulting in larger values of $\mathbf{p}_{\mathrm{res}}$ in the regions where $|\mathbf{p}_{\mathrm{ML}}|$ is large. These regions are highly nonlinear and appear as the simulation progresses. It is noteworthy that the emulation errors become particularly visible when visualising the final quantities in Fig. 5, given their lesser prevalence at earlier times.

To quantify the magnitude and time-dependence of the emulation error, we plot the root mean squared error (RMSE) between the true $\mathbf{p} - \mathbf{p}_{\mathrm{LPT}}$ and $\mathbf{p}_{\mathrm{ML}}$, as predicted by the emulator, as a function of the scale factor in Fig. 6. We present the mean and standard deviation of the RMSE across 50 test simulations. At early times, the trajectories of the particles are well described by perturbation theory. Thus, even though LPT is not a perfect description of the dynamics, the emulator can easily correct for the error, maintaining a relatively constant RMSE of less than $20\,\mathrm{km\,s^{-1}}$ for $a < 0.5$. We observe a slight decrease in RMSE between $a = 0.2$ and $a = 0.4$, which is understandable given Fig. 3: initially, the field exhibits a high degree of small-scale structure, which becomes less significant and approximately constant over time, making $\mathbf{p} - \mathbf{p}_{\mathrm{LPT}}$ easier to predict during this period. Beyond $a \approx 0.5$, the small-scale dynamics become highly non-linear, making it more challenging for the emulator to predict the correct frame of reference. Consequently, we observe the behaviour schematically illustrated in Fig. 1: the emulation error grows at late times, approximately doubling between $a = 0.5$ and $a = 1$. Jamieson et al. (2023b) and Jamieson et al. (2024) found similar issues with predicting virialised motions within collapsed regions due to their chaotic and random nature. It is precisely these emulation errors that we aim to correct using the COCA framework.

4.2. COCA performance

We now turn to testing the use of our frame of reference emulator within a cosmological simulation. To do this, for each realisation of initial conditions in our test set, we run a reference simulation (see Sect. 3.1) as well as COCA and COLA simulations with varying specifications. For these runs, we use 20 time steps between $a = 0.05$ and $a = 1$, spaced linearly in scale factor, but we vary the number of force evaluations n_{f}. After some experimentation, we found that the best strategy to maximise the statistics described in Sect. 3.5 is to place all force

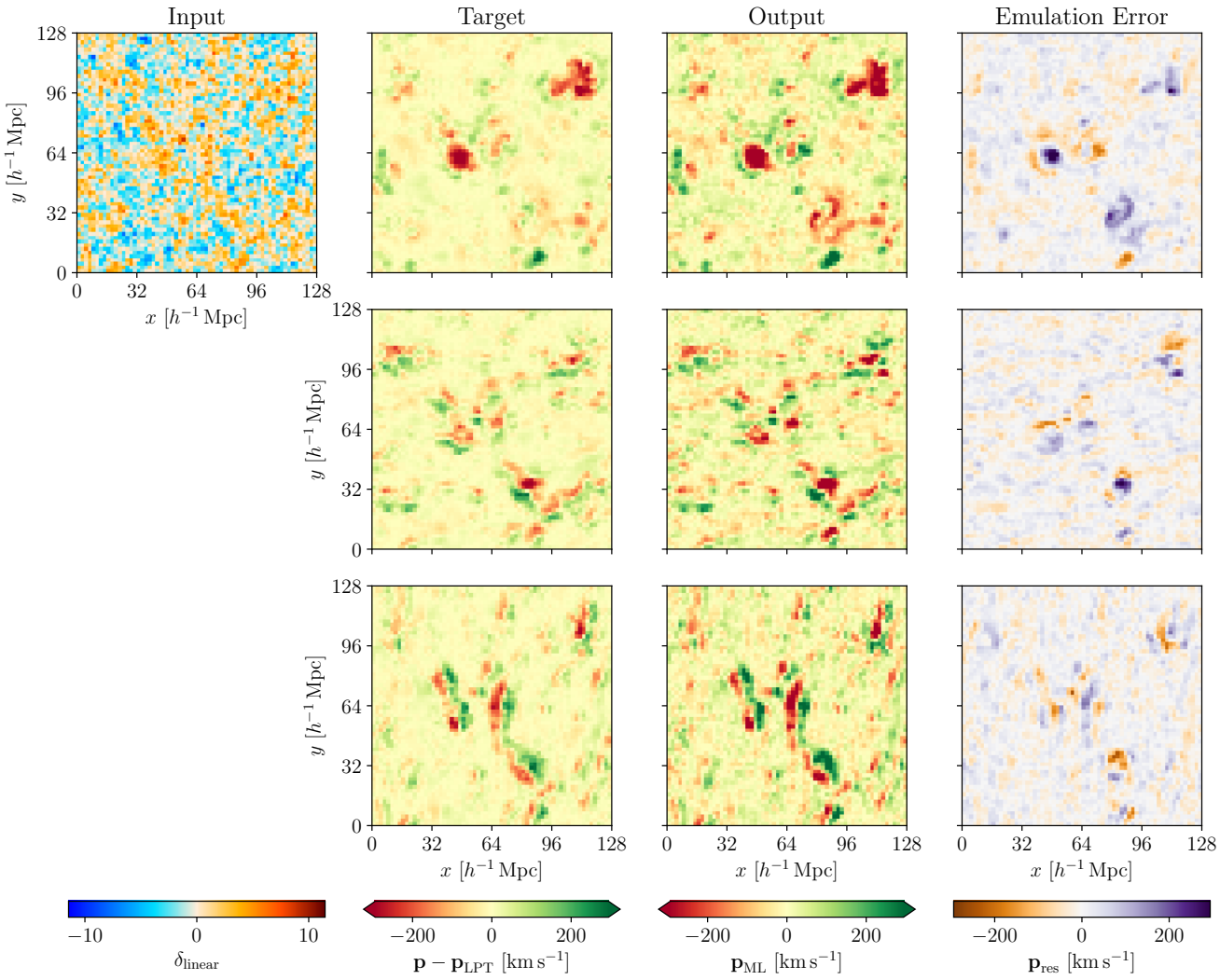

Fig. 5. Slices of the input, target, output, and error of the frame of reference emulator at the final time step (i.e. with style parameter $a = 1$). The input is the (scalar) linear density field (first column). The emulator aims to predict the three components (one per row) of its target $\mathbf{p} - \mathbf{p}_{\mathrm{LPT}}$ (second column). The emulator's predictions are shown in the third column, and the emulation error $\mathbf{p}_{\mathrm{res}} = \mathbf{p} - \mathbf{p}_{\mathrm{LPT}} - \mathbf{p}_{\mathrm{ML}}$, is shown in the final column.

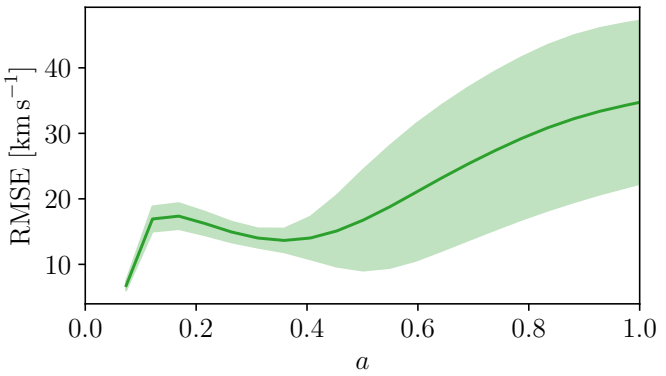

Fig. 6. Root mean squared error (RMSE) of the emulated frame of reference, $\mathbf{p}_{\mathrm{ML}}$, compared to $\mathbf{p} - \mathbf{p}_{\mathrm{LPT}}$, as a function of the scale factor a. The solid line indicates the mean across the test simulations, while the shaded band represents the standard deviation.

evaluations at the end of the simulation. This is expected, as the dynamics become more non-linear at later times, making it crucial to accurately resolve particle trajectories during these periods, especially since the emulation error is also largest at these times (see Fig. 6).

In Fig. 7, we plot a slice of the final density field for one of the reference simulations in our test set, as well as the corresponding COCA simulations with $n_{\mathrm{f}} = 10$ and $n_{\mathrm{f}} = 20$, and their respective residuals relative to the reference. Both COCA simulations accurately recover the overall structure of the cosmic web, with correctly positioned filaments and nodes. With the smaller number of force evaluations, there is a small residual in the final density, but this has almost completely disappeared when $n_{\mathrm{f}} = 20$.

To assess the relative performance of COCA and COLA and to determine the optimal number of force evaluations, in Fig. 8 we plot the fractional error on the matter power spectra and the cross-correlation coefficient for the $a = 1$ density field as a function of wavenumber for both simulation frameworks on the test set. As a sanity check, we verify that both COLA and COCA achieve similar performance when performing a force evaluation at each of the 20 time steps. Our first observation is that COCA performs dramatically better than COLA, even when using few force evaluations. It is unsurprising that with $n_{\mathrm{f}} = 0$ COLA performs poorly, as this is merely the LPT

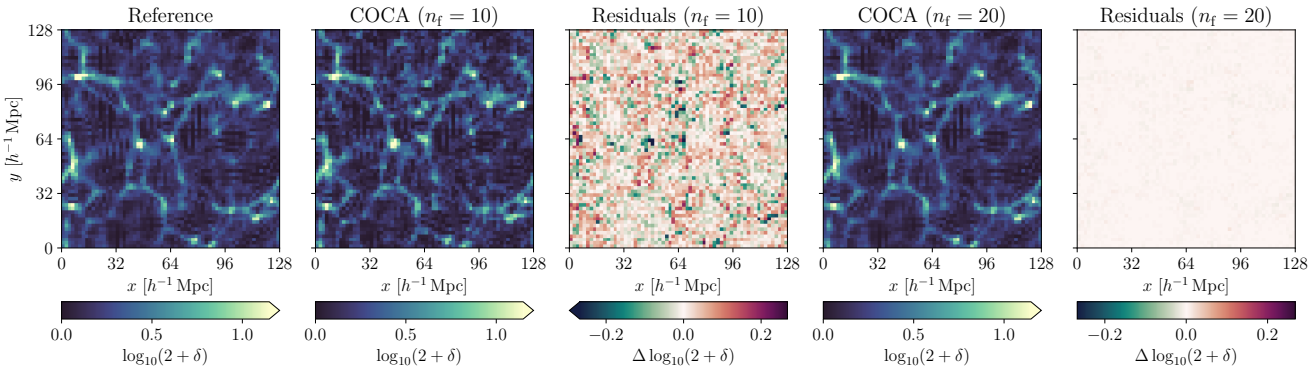

Fig. 7. Slices of the final ($a = 1$) matter density field for a reference simulation (first column) compared to the corresponding COCA simulations using either 10 (second column) or 20 (fourth column) force evaluations. The residuals relative to the reference simulation are shown in the third and fifth columns.

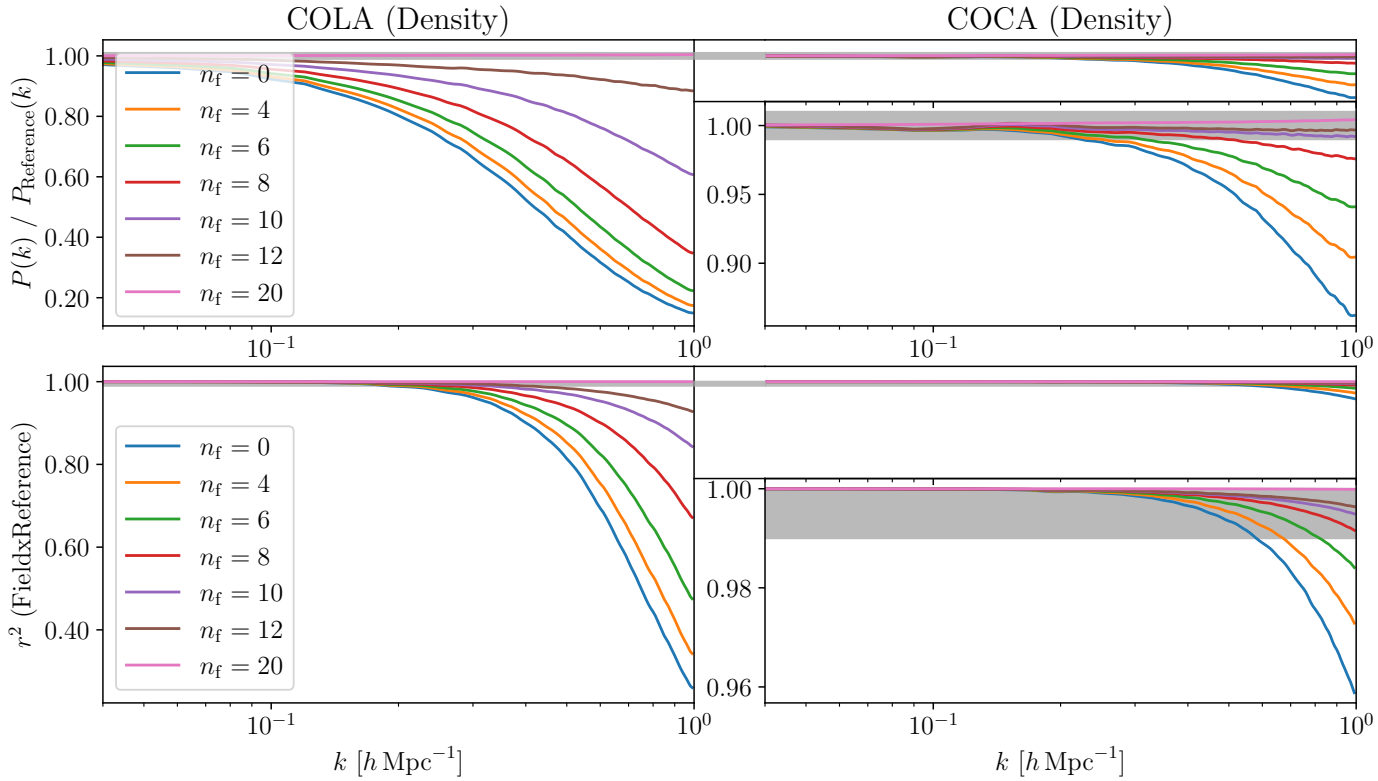

Fig. 8. Ratio of the matter power spectrum (top row) and the cross-correlation (bottom row) with respect to the reference simulation, for COLA (left column) and COCA (right column) simulations with varying numbers of force evaluations, n_f. The coloured lines represent the mean over the test set. In the COCA column, the top panel of each row is plotted on the same scale as COLA, while the lower panel provides a zoomed-in version. The grey band indicates 1% agreement with the reference. COCA simulations are much closer to the reference even when using far fewer force evaluations, and the agreement improves as n_f increases.

prediction, which is known to be a poor description at this redshift and on these scales. In contrast, we find that COCA with $n_f = 0$ is already extremely accurate: purely following the trajectories of the emulated frame of reference ($n_f = 0$) produces a behaviour practically identical to running a COLA simulation with $n_f = 12$ force evaluations. The matter power spectrum of the emulated field is 99% accurate up to $k \approx 0.3\,h\,\mathrm{Mpc}^{-1}$, with $r^2(k) > 0.99$ up to $k \approx 0.6\,h\,\mathrm{Mpc}^{-1}$. We would expect that if the training simulations and evolution used a higher accuracy gravity solver (e.g. P^{3}M or a tree-based approach), COCA would outperform COLA. However, it is not possible to check this conjecture in this example, since both the frame of reference emulator and COCA solver are based on PM forces. Despite the good predictions of the emulator, we see that the relative error on the power spectrum increases to more than 10% at $k = 1\,h\,\mathrm{Mpc}^{-1}$ when $n_f = 0$. However, the error is reduced to less than 1% up to $k = 0.5\,h\,\mathrm{Mpc}^{-1}$ by adding just 8 force evaluations, and less than 1% up to $k = 1\,h\,\mathrm{Mpc}^{-1}$ with 10 force evaluations, both in terms of the power spectrum and phase accuracy. This feature highlights the benefit of the COCA framework: we can use ML to provide good approximations to the true solution and run a physical simulation to correct for any errors made, using far fewer force evaluations than would ordinarily be required.

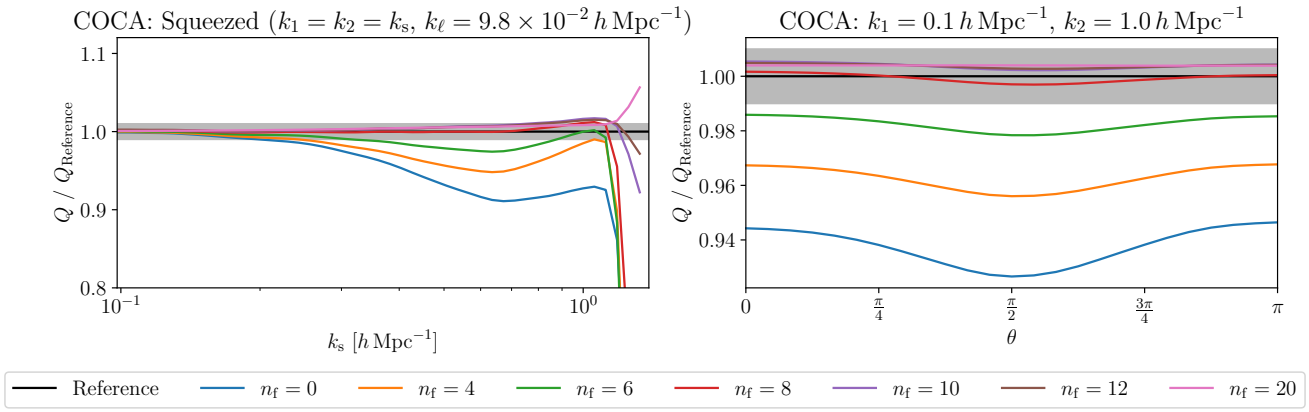

Fig. 9. Ratio of the reduced bispectrum (Eq. (18)) between COCA simulations and the reference, as a function of the number of force evaluations, n_f. In the left panel, we plot a squeezed configuration, where the third wavenumber is $k_3 = k_\ell = 9.8 \times 10^{-2}\,h\,\mathrm{Mpc}^{-1}$ and we vary $k_1 = k_2 = k_s$. In the right panel, we choose $k_1 = 0.1\,h\,\mathrm{Mpc}^{-1}$ and $k_2 = 1.0\,h\,\mathrm{Mpc}^{-1}$, and plot Q as a function of the angle between these two vectors, θ. The coloured lines represent the mean over the test set. The grey band indicates 1% agreement.

The same behaviour is observed when considering the three-point statistics. In Fig. 9 we plot the bispectrum for the COCA simulations in the configurations outlined in Sect. 3.5. As with the power spectrum, reasonable agreement with the reference is achieved without any force corrections, with errors of the order of 5-10%. However, with just 8 force evaluations, we achieve nearly a perfect agreement with the reference for almost all configurations considered, with the only discrepancy occurring for $k_s > 1\,h\,\mathrm{Mpc}^{-1}$.

We now evaluate the accuracy of the simulated velocity fields by plotting the error on the power spectrum and cross-correlation coefficient for the final velocity potential in Fig. 10. Velocity fields are very poorly predicted for all COLA simulations that skip force evaluations, with an under-prediction of power beyond $k \approx 0.1\,h\,\mathrm{Mpc}^{-1}$, and an over-prediction as we approach the Nyquist frequency of our simulations. The cross-correlation between the COLA velocities and the reference is also very low, with practically zero correlation at $k = 1\,h\,\mathrm{Mpc}^{-1}$ when no force evaluations are used, and with $r^2(k) \approx 0.5$ at this scale for $n_f = 12$. This is unsurprising, since this latter case is equivalent to initialising a COLA simulation with an LPT prediction at a redshift of $z = 1.5$ and using 12 time steps; we would not expect the initial conditions of such a simulation to be reasonable, as this is well beyond the validity of LPT. However, this problem is alleviated if we use an emulated frame of reference. Using only the emulator ($n_f = 0$) reduces the error on the velocity field power spectrum to approximately 5% at a $a = 1$, which, although still reasonably large, is much smaller than what is found with COLA with up to $n_f = 12$. The advantage of the COCA framework is particularly evident when varying n_f, as the addition of just 6 force evaluations practically eliminates this error, reducing it to 1%. Similarly, we find that the COCA fields are much more correlated with the reference, even when using far fewer force evaluations, with $r^2(k) > 0.8$ for all $k \lesssim 1\,h\,\mathrm{Mpc}^{-1}$ and for any number of force evaluations. The degree of correlation improves as n_f is increased.

In summary, we find that our emulator can reasonably recover the density and velocity fields even without any correction. However, emulation errors of up to $O(10\%)$ remain, but these can be reduced to the sub-percent level with just 8 force evaluations. Thus, COCA is able to correct for mistakes made in the emulation of particle trajectories by running a simulation in the corresponding frame of reference.

4.3. COCA with mis-specified cosmological parameters

One of the key motivations behind the COCA framework is the concept of ML safety. Although emulation techniques have previously been applied to predict the results of dark matter simulations (Perraudin et al. 2019; He et al. 2019; Alves de Oliveira et al. 2020; Jamieson et al. 2023b, 2024; Giusarma et al. 2023; Conceição et al. 2024; Saadeh et al. 2025), there may be concerns that the emulated solutions might not match the truth if the initial conditions or cosmological parameters are 'unusual', that is, dissimilar to the training data. The capacity of emulators to extrapolate was tested by Jamieson et al. (2023a) in the context of well-understood simple matter distributions that had not been seen during training. Furthermore, Doeser et al. (2024) found that their emulator performed well with initial conditions containing significantly less power than their training examples. However, with regular emulators, it is not possible to test all possible configurations, and thus, in general, we can only hope that the model extrapolates well to the application of interest. In contrast, COCA uses a frame of reference emulator but solves the fundamental equations of motion. Therefore, any extrapolation mistake made in the emulation should be automatically corrected, unlike with the use of an emulator alone.

Our frame of reference emulator was trained using simulations run at a single cosmology. To test its out-of-distribution behaviour and its use in the COCA formalism, we ran 50 additional test simulations with a different set of cosmological parameters: $\Omega_b = 0.03$, $\Omega_m = 0.35$, $h = 0.7$, $n_s = 0.99$, and $\sigma_8 = 0.9$. These parameters are chosen to be relatively extreme, yet still within the support of a moderately wide prior that could be used for a cosmological analysis. We note that we use the correct cosmological parameters for producing the initial density field, obtaining the LPT displacement fields, and solving the equations of motion; the only place where cosmological parameters are mis-specified is in the prediction of $\mathbf{p}_{\mathrm{ML}}$.

In Fig. 11, we plot the fractional error on the power spectra and the cross-correlation coefficients for the density and velocity fields in COCA simulations. The reference is the COLA simulations run with the same initial conditions, which are not subject to model mis-specification in this scenario. Despite the relatively extreme cosmological parameters, the uncorrected fields ($n_f = 0$)

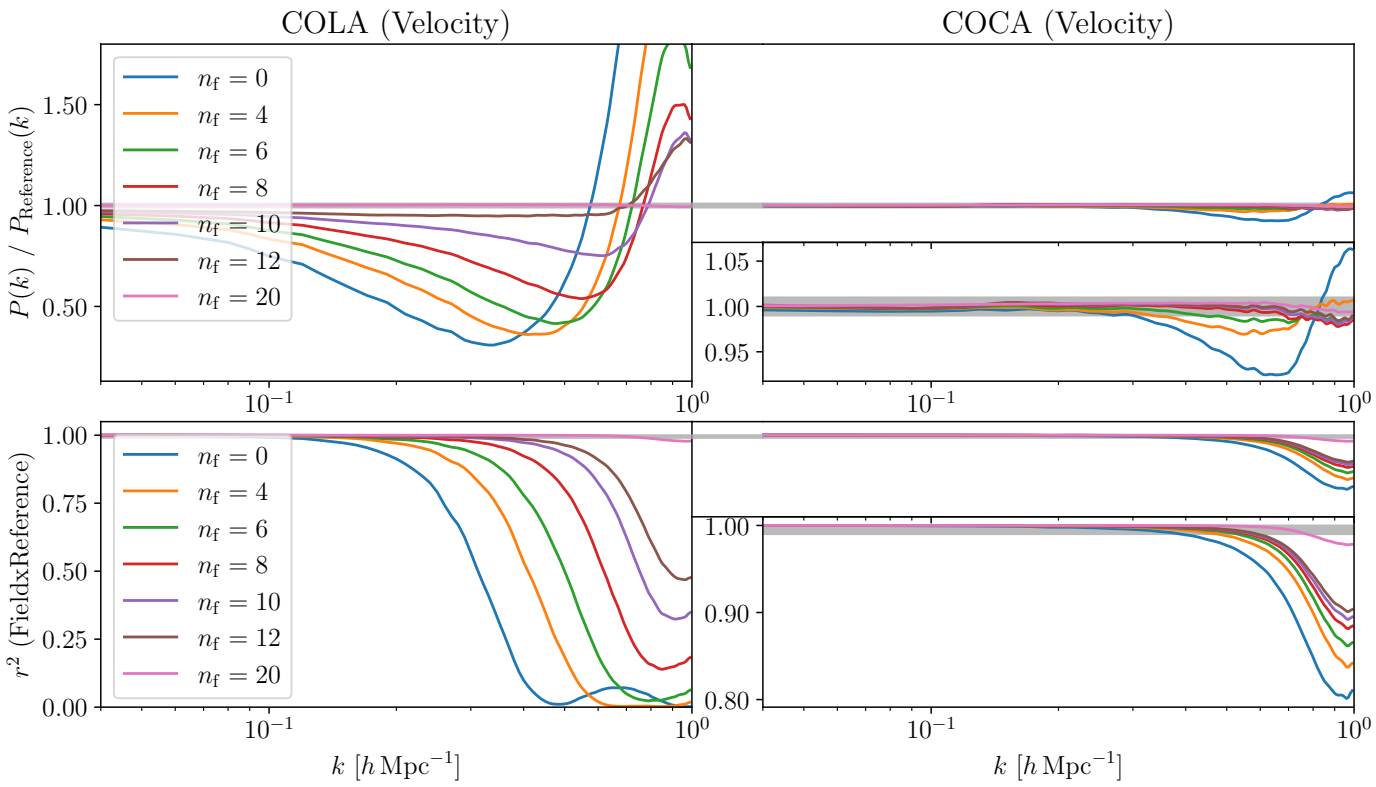

Fig. 10. Same as Fig. 8, but for the velocity-field two-point statistics. Again, we see that COCA performs much better than COLA with fewer force evaluations, and that the result converges to the truth as $n_{\rm f}$ increases.

yield reasonable power spectra and cross-correlation. The mean error on the density power spectrum is approximately 20% by $k = 1\,h\,\mathrm{Mpc}^{-1}$ with $r^2(k) > 0.93$ at these scales, while the velocity power spectrum has slightly smaller errors — around 10% — and $r^2(k) \approx 0.85$ by $k = 1\,h\,\mathrm{Mpc}^{-1}$. This moderate agreement with the truth is enabled by using the initial density field rather than the white noise field as the input to the emulator (see Sect. 3.3). Indeed, even if the initial density appears different from that of the training simulations, the emulator does not have to predict the relevant initial matter power spectrum, which contains the entire dependence on all cosmological parameters except $\Omega_{\rm m}$. Additionally, since we would expect $\mathbf{p}_{\rm ML}$ to be sourced only by local contributions in Lagrangian coordinates, the sensitivity to cosmological parameters should be relatively small. Similar moderately accurate extrapolation behaviour has also been observed in other cosmological simulation emulators (He et al. 2019; Kodi Ramanah et al. 2020; Alves de Oliveira et al. 2020; Lanzieri et al. 2022; Payot et al. 2023; Saadeh et al. 2025).

Despite the moderate performance of this emulator in the presence of cosmological parameter mis-specification, without any force evaluations (i.e. with $n_{\rm f} = 0$), the error on the matter power spectrum would be too large for current cosmological analyses (Taylor et al. 2018). Therefore, relying solely on an emulator of particles' trajectories (i.e. a frame of reference emulator with $n_{\rm f} = 0$) as a forward model would produce inappropriate results and would not be a safe use of ML. However, trajectories can be rectified in the COCA framework by evaluating gravitational forces and solving for the residual displacements with respect to the emulated frame of reference. In our test, using just 8 force evaluations is sufficient to achieve percent-level agreement in both $P(k)$ and $r^2(k)$ for the density field, for all $k < 1\,h\,\mathrm{Mpc}^{-1}$ (see Fig. 11). The same conclusion is true for the velocity field up to $k \approx 0.6\,h\,\mathrm{Mpc}^{-1}$.

It is tempting to question whether an emulator trained on an incorrect cosmology could yield a worse frame of reference than 2LPT at the correct cosmology, thereby negating any advantage of COCA over COLA. We tested this question in Fig. 12, where we compared the performance of COCA using a reference frame derived from a mis-specified cosmology (while all other computations, including gravitational evolution, employ the correct cosmology as noted above), against COLA, which uses the correct cosmological parameters. For all summary statistics, we find that even with an incorrect cosmology, COCA produces substantially more accurate density fields for any given number of force evaluations. Thus, despite the presence of emulation inaccuracies in the mis-specified scenario, our emulator still outperforms the 2LPT frame of reference and significantly reduces the number of force evaluations required to achieve an accurate density field. We have verified that this conclusion remains valid when varying $\Omega_{\rm m}$ within the range of $[0.2, 0.4]$.

Thus, with only a small additional computational cost, we can convert an unsafe use of ML in cosmology into a well-behaved one, even when the emulator is applied outside the range of its training data. This is one of the main benefits of COCA compared to traditional emulators of N-body simulation results.

4.4. COCA versus a Lagrangian displacement field emulator

In this work, we advocate for using an emulator for the frame of reference in N-body simulations, as it allows for correcting emulation errors by introducing force evaluations. This approach contrasts with previous emulators (e.g. He et al. 2019;

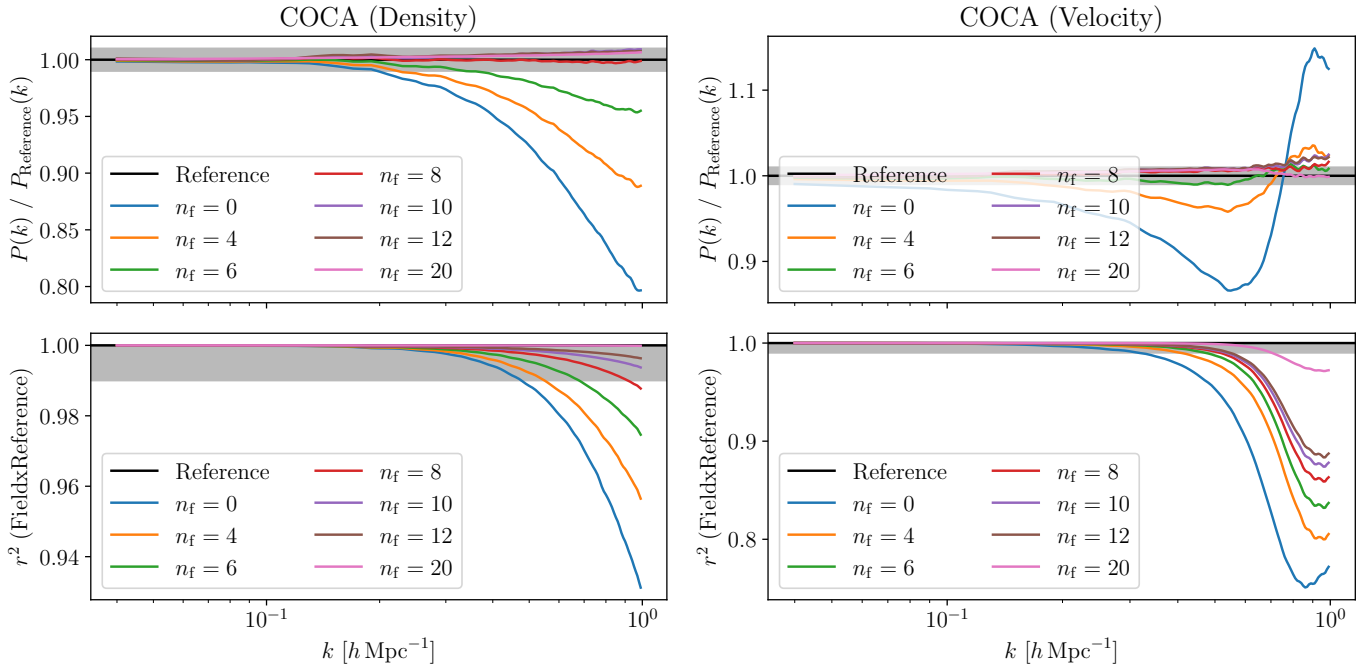

Fig. 11. Performance of the COCA framework when applied outside the range of the training data. We compare the power spectrum (top row) and cross-correlation coefficient (bottom row) of the matter density field (left column) and velocity potential (right column) when using our emulator with a different set of cosmological parameters than it was trained on. The coloured lines show the mean across 50 test simulations as a function of the number of force evaluations, $n_{\rm f}$. The grey band indicates 99% agreement. In this test of robustness to cosmological parameter mis-specification, only eight force evaluations are required to correct the emulation error up to $k \gtrsim 0.6\,h\,\mathrm{Mpc}^{-1}$.

Alves de Oliveira et al. 2020; Jamieson et al. 2023b, 2024), which directly predict the Lagrangian displacement field, namely, the simulation output. This section compares the relative accuracy of these two approaches as a function of the number of force evaluations in COCA.

To investigate this question, we train a time-dependent emulator for the residual displacement field $\boldsymbol{\Psi}_{\rm ML}(\mathbf{q}, a)$, defined as the difference between the true Lagrangian displacement field $\boldsymbol{\Psi}(\mathbf{q}, a)$ and that predicted by LPT, $\boldsymbol{\Psi}_{\rm LPT}(\mathbf{q}, a)$. We opted to train a new $\boldsymbol{\Psi}$-emulator rather than directly compare COCA to existing literature results to minimise the impact of differences in gravity solvers, training set sizes, architecture choices, and training procedures. For a fair comparison, we trained our $\boldsymbol{\Psi}$-emulator using the same simulations as for the frame of reference emulator, employing the same architecture and training procedure outlined in Sect. 3.4 (with $\mathbf{p}$ replaced by $\boldsymbol{\Psi}$ in the loss function).

In a similar manner as before, we begin by normalising the target variable by defining the function, $\psi(a)$, such that

$$\boldsymbol{\Psi}_{\rm ML}(\mathbf{q}, a) \equiv \psi(a)\tilde{\boldsymbol{\Psi}}_{\rm ML}(\mathbf{q}, a), \tag{19}$$

where $\tilde{\boldsymbol{\Psi}}_{\rm ML}(\mathbf{q}, a)$ has unit standard deviation. Applying symbolic regression to the function $\psi(a)$, we find that it is well approximated by

$$\psi(a) \approx a^{\phi_0} + \phi_1, \tag{20}$$

with $\phi_0 = 1.2412539$ and $\phi_1 = -0.05402543$, yielding a root mean squared error of 5×10^{-4}. We took the linear density field as input, but this time an output of $\tilde{\boldsymbol{\Psi}}_{\rm ML}(\mathbf{q}, a)$. We evaluated this emulator on the same test simulations as for the frame of reference emulator and converted the returned Lagrangian displacements into an Eulerian density field using a cloud-in-cell estimator. We computed the power spectrum, cross-correlation coefficient, and bispectra (see Sect. 3.5). These are plotted in Fig. 12, where we compare against COCA without force evaluations ($n_{\rm f} = 0$; using solely the frame of reference emulator) and both $n_{\rm f} = 4$ and $n_{\rm f} = 8$. We performed this analysis for both the fiducial and mis-specified cosmology (see Sect. 4.3).

When compared to COCA with $n_{\rm f} = 0$, the Lagrangian displacement field emulator more accurately recovers the reference density field. The power spectra of the two methods are relatively similar with fiducial cosmological parameters, but the difference becomes more pronounced when the cosmology is mis-specified. For all other metrics, the $\boldsymbol{\Psi}$-emulator produces summary statistics that are closer to the reference. This behaviour is expected: the $\boldsymbol{\Psi}$-emulator is designed to optimise the prediction of dark matter particle positions through its loss function, naturally resulting in an accurate density field. In contrast, the frame of reference emulator in COCA aims to match particle momenta $\mathbf{p}$. Consequently, without force evaluations, emulation errors in $\mathbf{p}$ accumulate over time, reducing the quality of the final density field.

Although the $\boldsymbol{\Psi}$-emulator performs better than COCA with no force evaluations, there is no way to correct its errors, meaning its performance cannot be improved. Conversely, in COCA, force evaluations can be added to correct the errors made by the frame of reference emulator. Figure 12 shows that the addition of only four force evaluations results in performance nearly identical to that of the $\boldsymbol{\Psi}$-emulator for the bispectra, and better results for two-point statistics with residual errors reduced by a factor of 1.4. Residual errors almost entirely disappear when eight force evaluations are used in COCA: the final power spectrum $P(k)$ has approximately four to five times smaller errors than the one derived from the $\boldsymbol{\Psi}$-emulator at all scales for both cosmologies. Thus, even with very limited additional computations beyond the emulation, the COCA framework outperforms a Lagrangian displacement field emulator.

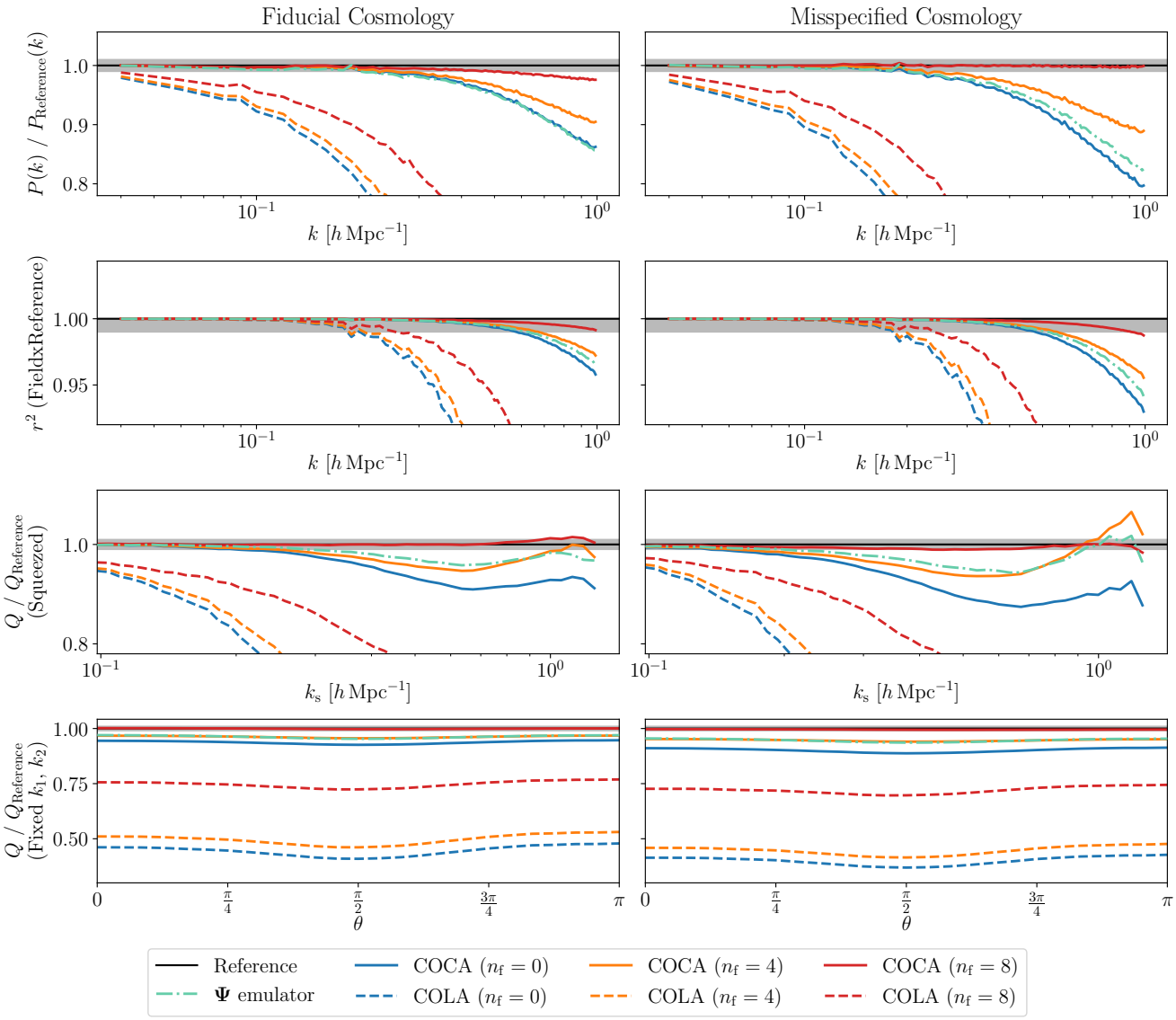

Fig. 12. Relative performance of COCA and COLA versus an emulator of the displacement field $\mathbf{\Psi}$. We compute the summary statistics outlined in Sect. 3.5 for the final ($a = 1$) matter density field and compare results at both the training cosmology and a mis-specified one for COCA (COLA is evaluated at the correct cosmology in both cases). Although directly emulating $\mathbf{\Psi}$ produces a more accurate density field than simply emulating the momentum field $\mathbf{p}$ (with $n_{\rm f} = 0$), using the COCA framework (emulating the frame of reference and employing additional force evaluations) yields the best performance. Moreover, even when using a mis-specified cosmology to predict the frame of reference, COCA significantly outperforms COLA for any given number of time steps.

We note that our displacement emulator is slightly less accurate than that of He et al. (2019), who also emulated a PM-like output. They achieved errors on $P(k)$ of 0.8% and 4% at $k = 0.4\,h\,\mathrm{Mpc}^{-1}$ and $k = 0.7\,h\,\mathrm{Mpc}^{-1}$, respectively, whereas our emulator is accurate to 3% and 9% at these scales. We attribute this discrepancy to our use of fewer training simulations (2000 fields compared to 10 000 in He et al. 2019), the need for our emulator to learn time-dependence (only 100 of the 2000 training fields are at $a = 1$), and He et al. (2019) employing a more optimised architecture and training schedule. As mentioned in Sect. 3.1, since the aim of this paper is to demonstrate how to correct for emulation errors rather than produce the optimal emulator, we chose not to increase the number of training simulations or fine-tune the architecture, as our emulator is already of similar quality to those in the literature. If a frame of reference emulator with performance similar to that of He et al. (2019) were used in COCA, fewer time steps would be needed to correct for emulation errors, thereby achieving the same theoretical guarantees with reduced computational expense.

4.5. Timing tests

To assess the computational performance of COCA, in Fig. 13, we show the required amount of CPU/GPU time for each of the stages of the framework. To perform the timing tests, we use an Intel Xeon Gold 6230 processor with 40 CPU cores and an Nvidia V100 GPU. We compare the results of running COLA with 20 time steps (and 20 force evaluations) and COCA with the same number of time steps, but with varying numbers of force evaluations (0, 8, and 20). All other settings are identical to those in Sect. 4.2.

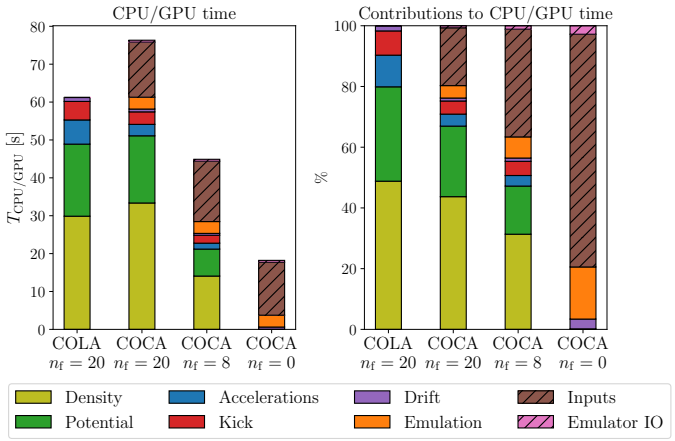

Fig. 13. Total CPU/GPU time for COCA with varying numbers of force evaluations, n_f, compared to COLA. Both COCA and COLA use 20 time steps. In the left column, we report the total time, and in the right column, we report the relative contributions of the various operations. Solid bars correspond to the main computations, and hashed bars indicate inputs and outputs. We note that the current implementation of COCA in SIMBELMYNË decouples emulation and other computations, which is not a requirement–emulation could occur on the fly during N-body evolution. In this way, the cost of input+output operations would be effectively reduced to zero.

The COLA simulation takes approximately 61 CPU-seconds to run, with almost half of this time spent on cloud-in-cell binning (converting particle positions to the density field). In our test, running COCA with $n_f = 0$ is approximately four times faster. In the current implementation, the emulation and the simulation codes are disjoint, with the frame of reference being written to and then read from disk at each kick time step (a process responsible for 77% of the CPU/GPU time in this case). Separating emulation and N-body evolution is not a fundamental requirement of the COCA framework: we could emulate on the fly, which would effectively reduce the input/output time to zero for a slightly higher memory cost (two $\mathbf{p}_{\mathrm{ML}}$ fields need to be kept in memory for each kick operation, see Eq. (A.21)). Such an approach would make the $n_f = 0$ case 18 times faster than the COLA simulation, with this emulator.

To enable the safe use of an ML emulator, COCA relies on a finite number of force evaluations. Naturally, if we use the same number of force evaluations, then COCA is more computationally expensive than COLA, since it must perform the same steps as COLA, but with an additional emulation stage. However, because the ML correction makes the frame of reference more accurate than LPT alone, the number of force evaluations can be reduced to approximately eight (see Sect. 4.2). With the current implementation of separate emulation and N-body evolution codes, the cost of COCA with $n_f = 8$ is approximately two-thirds of the cost of the COLA simulation. If emulation were done on the fly, the time required for inputs (36% of the total time) would be eliminated, making COCA 2.3 times faster than COLA.

These timing improvements are expected to become more dramatic if COCA were extended to include a more accurate gravity solver, such as a P^3M or tree-based code. The computational expense for computing the forces in these codes is significantly higher than in the PM-based model used in this work. Therefore, reducing the number of force evaluations would dramatically improve run time. We leave such an investigation to future work.

5. Discussion and conclusion

In this paper, we introduce COmoving Computer Acceleration (COCA), a hybrid formalism involving ML and N-body simulations. Unlike previous works that directly emulate the simulation output, COCA solves the dynamics of an N-body simulation using a ML frame of reference. COCA can be seen as an improvement of COLA, which solves the dynamics of an N-body simulation in the LPT frame of reference. By virtue of the principle of Galilean invariance, equations of motion can be solved in any frame of reference, making COCA a ML-safe framework. COCA is the first framework to use physics to determine the (otherwise uncorrected) emulation error in N-body simulations using ML and correct for it.

The concept behind COCA is entirely independent of the N-body solver and of the ML emulation algorithm used. For this proof-of-concept, we employed a PM approach to solving the equations of motion and a V-net architecture for the frame of reference, with fixed cosmological parameters. We have demonstrated that after the ML-prediction of the optimal frame of reference (the one in which all particles are at rest), running the N-body simulation corrects for potential emulation errors in the particle trajectories. We have quantitatively shown that the number of force evaluations required to achieve a given accuracy is reduced compared to COLA. The frame of reference emulator achieves between 1% and 10% accuracy when used in isolation, but only eight force evaluations are needed to reduce emulation errors to the percent level, compared to a 100-time step COLA simulation. Therefore, COCA can be utilised as a cheap N-body simulator. Furthermore, with eight force evaluations, COCA is four to five times more accurate than a Lagrangian displacement field emulator, when the frame of reference emulator and the Lagrangian displacement field emulator are trained using the same computational resources. This increased accuracy is due to COCA's ability to correct emulation errors and represents one of the main advantages of this framework compared to the direct emulation approach explored in earlier literature.

In Sect. 4.3, we demonstrate that our frame of reference emulator is moderately robust to changes in cosmological parameters (despite training at a fixed cosmology). However, the COCA framework can correct for extrapolation errors arising from applying the emulator outside the range of validity of the training simulations. Even when the frame of reference is inaccurate (because the ML training+prediction and the N-body evolution use different cosmological parameters), we found that a percent-level accuracy can be reached on final density and velocity fields up to $k \approx 0.6\,h\,\mathrm{Mpc}^{-1}$. Thus, the COCA framework provides ML safety even when models are required to extrapolate. There is no fundamental reason why the emulator cannot depend on cosmological parameters, and future implementations of COCA can include these as additional style parameters of the neural network.

Our example is focussed on relatively small simulation volumes (with a side length of $128\,h^{-1}\,\mathrm{Mpc}$) compared to those required for modern-day surveys (typically several gigaparsecs in length). With the current memory limitations of GPUs, it is not possible to emulate the entire volume with a single emulator at the desired resolution. As a workaround, Jamieson et al. (2023b) splits the volume into several padded sub-boxes and treats each one separately, relying on sequential predictions for particle displacements in each sub-box to cover the full volume. Similarly, in COCA, we could predict the frame of reference for particles in each sub-box and then solve the equations of motion in each sub-box independently. This idea relates to the algorithm introduced

by Leclercq et al. (2020) for perfectly parallel N-body simulations using spatial comoving Lagrangian Acceleration (sCOLA). There, a tiling of the simulation volume is used, and the evolution of tiles is spatially decoupled by splitting the Lagrangian displacement field into large and small-scale contributions. In sCOLA, the frame of reference used in the evolution of tiles is given by LPT, but it could be easily replaced by a frame of reference including both LPT and an ML contribution, as introduced in this paper. Such an approach would overcome the memory limitations of GPUs, which currently limit COCA to small simulation volumes. An additional benefit of this approach would be the inexpensive generation of light-cones. Indeed, when using a tiling approach as with sCOLA, only one tile needs to be evolved to a redshift of zero; the tiles farthest from the observer only need to be evolved until they intersect the light-cone at higher redshifts.

It is important to emphasise that the specific implementation details used in this work are not requirements, but just an example. For instance, it is also possible to use a perturbation theory-informed integrator for the equations of motion (Feng et al. 2016; List & Hahn 2024), an approach that is complementary to COLA for fast generation of approximate cosmological simulations. Furthermore, instead of using training simulations run with a PM gravity solver, the frame of reference given simulations could be learned with a higher level of force accuracy; for example, using a P^3M or tree-based gravity solver. Subsequently, solving the equations of motion in the emulated frame of reference with the same solver would result in simulations with similar accuracy to those of the training set, but with significantly reduced computational cost. The guarantee that any emulation mistakes are removed asymptotically as the number of force evaluations increases (a central feature of COCA) will remain. This ML safety cannot be guaranteed through direct emulation of P^3M or tree-based simulations. As with COLA, the COCA framework could be adapted to include more extended physical models, such as neutrinos, which induce a scale-dependent growth factor (Wright et al. 2017). Finally, although this work has been focussed on gravitational N-body simulations in a cosmological context, the approach of solving equations of motion in an emulated frame of reference could be applied to any kind of simulation involving interacting particles (e.g. electrodynamics, hydrodynamics, radiative transfer, magnetohydrodynamics). We generally expect a reduction in computational demands, while retaining physical guarantees of convergence to the truth.

Benefiting from its modest computational cost, COCA could be used in analyses of cosmological data using fully nonlinear models. It could straightforwardly be used as a forward model in implicit likelihood inference algorithms such as DELFI (Alsing et al. 2019; Makinen et al. 2021), BOLFI (Leclercq 2018), SELFI (Leclercq et al. 2019; Leclercq 2022), or the LTU-ILI pipeline (Ho et al. 2024). As COCA is an ML-safe framework, its use as a forward model will not bias the inference result. We also note that using a V-net emulator and a PM force solver, the entire COCA framework is differentiable. For the emulation of the frame of reference, differentiability is achieved via automatic differentiation. For the N-body evolution, differentiable PM simulators already exist (Wang et al. 2014; Jasche & Lavaux 2019; Modi et al. 2021; Li et al. 2022). Building upon these, future work could be dedicated to writing a differentiable COCA solver, which could be used in Bayesian large-scale structure inference using an explicit field-level likelihood (see Jasche & Lavaux 2019; Doeser et al. 2024; Wempe et al. 2024).

Machine learning offers great promise in the acceleration of forward modelling in the physical sciences. The output of any ML model is usually an approximation with inevitable emulation errors. In this paper, we have shown that emulation errors are correctable in gravitational N-body simulations. By solving the correct physical equations, while using the ML solution as an approximation, we can exploit the speed of ML and simultaneously retain the safety of more traditional methods.

Acknowledgements. We thank Guilhem Lavaux, Natalia Porqueres, Benjamin Wandelt, and Ewoud Wempe for useful comments and suggestions. DJB and LD are supported by the Simons Collaboration on "Learning the Universe." FL and MC acknowledge financial support from the Agence Nationale de la Recherche (ANR) through grant INFOCW, under reference ANR-23-CE46-0006-01. LD acknowledges support by the Swedish Research Council (VR) under the project 2020-05143 – "Deciphering the Dynamics of Cosmic Structure". This work has received funding from the Centre National d'Etudes Spatiales (CNES). This work was done within the Aquila Consortium (https://www.aquila-consortium.org/). *Statement of Contribution:* DJB implemented the frame of reference emulator, ran the COCA and COLA simulations, produced the plots, and contributed to the analysis and interpretation of the results and the writing of the paper. MC contributed to the model equations and the validation of COCA. LD provided code for the bispectrum analyses, advised on the design of the emulator, and contributed to the interpretation of ML performance results. FL conceived and designed the study, wrote the model equations, modified the SIMBELMYNË code to accept any arbitrary frame of reference as input, contributed to the analysis and interpretation of results, edited the manuscript, advised early-career authors, and secured funding. All authors read and approved the final manuscript.

References

Alsing, J., Charnock, T., Feeney, S., & Wandelt, B. 2019, MNRAS, 488, 4440
Alves de Oliveira, R., Li, Y., Villaescusa-Navarro, F., Ho, S., & Spergel, D. N. 2020, arXiv e-prints [arXiv:2012.00240]
Angulo, R. E., & Hahn, O. 2022, Liv. Rev. Comput. Astrophys., 8, 1
Bartlett, D. J., Desmond, H., & Ferreira, P. G. 2022, in IEEE Transactions on Evolutionary Computation (Institute of Electrical and Electronics Engineers (IEEE)), 1
Bernardeau, F., Colombi, S., Gaztañaga, E., & Scoccimarro, R. 2002, Phys. Rep., 367, 1
Birdsall, C. K., & Langdon, A. B. 1985, Plasma Physics via Computer Simulation (CRC Press)
Bouchet, F. R., Colombi, S., Hivon, E., & Juszkiewicz, R. 1995, A&A, 296, 575
Buchert, T. 1989, A&A, 223, 9
Chartier, N., Wandelt, B., Akrami, Y., & Villaescusa-Navarro, F. 2021, MNRAS, 503, 1897
Conceição, M., Krone-Martins, A., da Silva, A., & Moliné, Á. 2024, A&A, 681, A123
Dai, B., & Seljak, U. 2021, Proc. Nat. Acad. Sci., 118, e2020324118
Ding, S., Lavaux, G., & Jasche, J. 2024, A&A, 690, A236
Doeser, L., Jamieson, D., Stopyra, S., et al. 2024, MNRAS, 535, 1258
Feng, Y., Chu, M.-Y., Seljak, U., & McDonald, P. 2016, MNRAS, 463, 2273
Frontiere, N., Heitmann, K., Rangel, E., et al. 2022, ApJS, 259, 15
Giusarma, E., Reyes, M., Villaescusa-Navarro, F., et al. 2023, ApJ, 950, 70
Hahn, O., Angulo, R. E., & Abel, T. 2015, MNRAS, 454, 3920
He, K., Zhang, X., Ren, S., & Sun, J. 2016, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770
He, S., Li, Y., Feng, Y., et al. 2019, Proc. Nat. Acad. Sci., 116, 13825
Heitmann, K., Finkel, H., Pope, A., et al. 2019, ApJS, 245, 16
Ho, M., Bartlett, D. J., Chartier, N., et al. 2024, Open J. Astrophys., 7, 54
Hockney, R. W., & Eastwood, J. W. 1981, Computer Simulation Using Particles (McGraw-Hill)
Howlett, C., Manera, M., & Percival, W. J. 2015, Astron. Comput., 12, 109
Ishiyama, T., Prada, F., Klypin, A. A., et al. 2021, MNRAS, 506, 4210
Izard, A., Crocce, M., & Fosalba, P. 2016, MNRAS, 459, 2327
Jamieson, D., Li, Y., He, S., et al. 2023a, PNAS nexus, 2, pgac250
Jamieson, D., Li, Y., de Oliveira, R. A., et al. 2023b, ApJ, 952, 145

Jamieson, D., Li, Y., Villaescusa-Navarro, F., Ho, S., & Spergel, D. N. 2024, arXiv e-prints [arXiv:2408.07699]
Jasche, J., & Lavaux, G. 2019, A&A, 625, A64
Karras, T., Laine, S., Aittala, M., et al. 2020, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 8110
Kingma, D. P., & Ba, J. 2014, arXiv e-prints [arXiv:1412.6980]
Koda, J., Blake, C., Beutler, F., Kazin, E., & Marin, F. 2016, MNRAS, 459, 2118
Kodi Ramanah, D., Charnock, T., Villaescusa-Navarro, F., & Wandelt, B. D. 2020, MNRAS, 495, 4227
Lanzieri, D., Lanusse, F., & Starck, J.-L. 2022, arXiv e-prints [arXiv:2207.05509]
Leclercq, F. 2015, Ph.D. Thesis, Institut d'Astrophysique de Paris, France
Leclercq, F. 2018, Phys. Rev. D, 98, 063511
Leclercq, F. 2022, Phys. Sci. Forum, 5, 4
Leclercq, F., Jasche, J., & Wandelt, B. 2015, JCAP, 6, 15
Leclercq, F., Jasche, J., Lavaux, G., Wandelt, B., & Percival, W. 2017, JCAP, 6, 049
Leclercq, F., Enzi, W., Jasche, J., & Heavens, A. 2019, MNRAS, 490, 4237
Leclercq, F., Faure, B., Lavaux, G., et al. 2020, A&A, 639, A91
Li, Y., Ni, Y., Croft, R. A. C., et al. 2021, Proc. Nat. Acad. Sci., 118, e2022038118
Li, Y., Lu, L., Modi, C., et al. 2022, arXiv e-prints [arXiv:2211.09958]
List, F., & Hahn, O. 2024, J. Comput. Phys., 513, 113201
Loshchilov, I., & Hutter, F. 2017, arXiv e-prints [arXiv:1711.05101]
Lucie-Smith, L., Peiris, H. V., Pontzen, A., & Lochner, M. 2018, MNRAS, 479, 3405
Makinen, T. L., Charnock, T., Alsing, J., & Wandelt, B. D. 2021, JCAP, 2021, 049
Milletari, F., Navab, N., & Ahmadi, S. A. 2016, in 2016 Fourth International Conference on 3D Vision (3DV), 565
Modi, C., Lanusse, F., & Seljak, U. 2021, Astron. Comput., 37, 100505
Paszke, A., Gross, S., Massa, F., et al. 2019, in Advances in Neural Information Processing Systems, eds. H. Wallach, H. Larochelle, A. Beygelzimer, et al. (Curran Associates, Inc.), 32
Payot, N., Lemos, P., Perreault-Levasseur, L., et al. 2023, arXiv e-prints [arXiv:2311.18017]
Perraudin, N., Srivastava, A., Lucchi, A., et al. 2019, Comput. Astrophys. Cosmol., 6, 5
Planck Collaboration VI. 2020, A&A, 641, A6
Potter, D., Stadel, J., & Teyssier, R. 2017, Comput. Astrophys. Cosmol., 4, 2
Quinn, T., Katz, N., Stadel, J., & Lake, G. 1997, arXiv e-prints [arXiv:astro-ph/9710043]
Ronneberger, O., Fischer, P., & Brox, T. 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation (Cham: Springer International Publishing)
Saadeh, D., Koyama, K., & Morice-Atkinson, X. 2025, MNRAS, 537, 448
Tassev, S., & Zaldarriaga, M. 2012, JCAP, 12, 11
Tassev, S., Zaldarriaga, M., & Eisenstein, D. J. 2013, JCAP, 6, 36
Taylor, P. L., Kitching, T. D., & McEwen, J. D. 2018, Phys. Rev. D, 98, 043532
Villaescusa-Navarro, F. 2018, Astrophysics Source Code Library [record ascl:1811.008]
Vogelsberger, M., Marinacci, F., Torrey, P., & Puchwein, E. 2020, Nat. Rev. Phys., 2, 42
Wang, H., Mo, H. J., Yang, X., Jing, Y. P., & Lin, W. P. 2014, ApJ, 794, 94
Wang, Q., Gao, L., & Meng, C. 2022, MNRAS, 517, 6004
Wempe, E., Lavaux, G., White, S. D. M., et al. 2024, A&A, 691, A348
Wright, B. S., Winther, H. A., & Koyama, K. 2017, JCAP, 2017, 054
Zel'dovich, Y. B. 1970, A&A, 5, 84
Zhang, X., Lachance, P., Dasgupta, A., et al. 2024, arXiv e-prints [arXiv:2408.09051]

Appendix A: The actual equations

A.1. Model equations with COCA

Using the notations of Leclercq(2015, Appendix B) and Leclercq et al. (2020), we consider dark matter particles with positions $\mathbf{x}$ and momenta $\mathbf{p}$ in comoving coordinates. Denoting the scale factor as a and the over-density field as δ, the equations to be solved are

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}a} = \mathcal{D}(a)\mathbf{p} \quad \text{with} \quad \mathcal{D}(a) \equiv \frac{1}{a^2\mathcal{H}(a)}, \tag{A.1}$$

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}a} = \mathcal{K}(a)\boldsymbol{\nabla}\left(\Delta^{-1}\delta\right) \quad \text{with} \quad \mathcal{K}(a) \equiv -\frac{3}{2}\frac{\Omega_\mathrm{m}^{(0)}\mathcal{H}^{(0)2}}{a\mathcal{H}(a)}, \tag{A.2}$$

where $\mathcal{H}(a) \equiv a'/a$ is the conformal Hubble factor (where a prime denotes a derivative with respect to conformal time) and $\Omega_\mathrm{m}^{(0)}$ is the matter density parameter at the present time ($a = 1$). For simplicity, we note $\boldsymbol{\nabla}_\mathbf{x} = \boldsymbol{\nabla}$, $\Delta_\mathbf{x} = \Delta$ and $\delta(\mathbf{x}, a) = \delta$.

With $\mathbf{x}(a) = \mathbf{x}_\mathrm{LPT}(a) + \mathbf{x}_\mathrm{ML}(a) + \mathbf{x}_\mathrm{res}(a)$ (denoting the Lagrangian perturbation theory, machine-learnt, and residual contributions to the position, respectively), we note for each contribution $\mathrm{y} \in \{\mathrm{LPT}, \mathrm{ML}, \mathrm{res}\}$:

$$\frac{\mathrm{d}\mathbf{x}_\mathrm{y}}{\mathrm{d}a} \equiv \mathcal{D}(a)\mathbf{p}_\mathrm{y} \quad \text{and} \quad \frac{\mathrm{d}\mathbf{p}_\mathrm{y}}{\mathrm{d}a} = \frac{\mathrm{d}}{\mathrm{d}a}\left(\frac{1}{\mathcal{D}(a)}\frac{\mathrm{d}\mathbf{x}_\mathrm{y}}{\mathrm{d}a}\right) \equiv -\mathcal{K}(a)\mathcal{V}[\mathbf{x}_\mathrm{y}](a), \tag{A.3}$$

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

$$\mathcal{V}[\cdot](a) \equiv -\frac{1}{\mathcal{K}(a)}\frac{\mathrm{d}}{\mathrm{d}a}\left(\frac{1}{\mathcal{D}(a)}\frac{\mathrm{d}\cdot}{\mathrm{d}a}\right). \tag{A.4}$$

Analogously, we write the momenta as $\mathbf{p}(a) = \mathbf{p}_\mathrm{LPT}(a) + \mathbf{p}_\mathrm{ML}(a) + \mathbf{p}_\mathrm{res}(a)$, and thus Eqs. (A.1) and (A.2) take the form

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}a} = \mathcal{D}(a)\left\{\mathbf{p}_\mathrm{res}(a) + \mathbf{p}_\mathrm{LPT}(a) + \mathbf{p}_\mathrm{ML}(a)\right\}, \tag{A.5}$$

$$\frac{\mathrm{d}\mathbf{p}_\mathrm{res}}{\mathrm{d}a} = \mathcal{K}(a)\left\{\left[\boldsymbol{\nabla}\left(\Delta^{-1}\delta\right)\right](a) + \mathcal{V}[\mathbf{x}_\mathrm{LPT}](a) + \mathcal{V}[\mathbf{x}_\mathrm{ML}](a)\right\}. \tag{A.6}$$

The analytical properties of LPT are (see e.g. Leclercq 2015, Eqs. (1.7), (1.96), and (1.118) and Appendix B):

$$\mathbf{x}_\mathrm{LPT}(a) = \mathbf{q} - D_1(a)\boldsymbol{\Psi}_1 + D_2(a)\boldsymbol{\Psi}_2, \tag{A.7}$$

$$\mathcal{D}(a)\mathbf{p}_\mathrm{LPT} = -\frac{\mathrm{d}D_1}{\mathrm{d}a}\boldsymbol{\Psi}_1 + \frac{\mathrm{d}D_2}{\mathrm{d}a}\boldsymbol{\Psi}_2, \tag{A.8}$$

$$\mathcal{V}[\mathbf{x}_\mathrm{LPT}](a) = -D_1(a)\boldsymbol{\Psi}_1 + \left[D_2(a) - D_1^2(a)\right]\boldsymbol{\Psi}_2, \tag{A.9}$$

where $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$ are the time-independent first and second order displacements, with corresponding growth factors D_1 and D_2. This gives

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}a} = \mathcal{D}(a)\left\{\mathbf{p}_\mathrm{res}(a) + \mathbf{p}_\mathrm{ML}(a)\right\} - \frac{\mathrm{d}D_1}{\mathrm{d}a}\boldsymbol{\Psi}_1 + \frac{\mathrm{d}D_2}{\mathrm{d}a}\boldsymbol{\Psi}_2, \tag{A.10}$$

$$\frac{\mathrm{d}\mathbf{p}_\mathrm{res}}{\mathrm{d}a} = \mathcal{K}(a)\left\{\left[\boldsymbol{\nabla}\left(\Delta^{-1}\delta\right)\right](a) - D_1(a)\boldsymbol{\Psi}_1 + \left[D_2(a) - D_1^2(a)\right]\boldsymbol{\Psi}_2 + \mathcal{V}[\mathbf{x}_\mathrm{ML}](a)\right\}. \tag{A.11}$$

Furthermore, for any arbitrary positive function u of a, we can rewrite

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}a} = \mathcal{D}(a)u(a)\left\{\frac{1}{u(a)} \times \mathbf{p}_\mathrm{res}(a) + \frac{1}{u(a)} \times \mathbf{p}_\mathrm{ML}(a)\right\} - \frac{\mathrm{d}D_1}{\mathrm{d}a}\boldsymbol{\Psi}_1 + \frac{\mathrm{d}D_2}{\mathrm{d}a}\boldsymbol{\Psi}_2, \tag{A.12}$$

$$\frac{\mathrm{d}\mathbf{p}_\mathrm{res}}{\mathrm{d}a} = \frac{\mathrm{d}u(a)}{\mathrm{d}a}\left\{\frac{\mathcal{K}(a)}{\mathrm{d}u(a)/\mathrm{d}a} \times \left[\left[\boldsymbol{\nabla}\left(\Delta^{-1}\delta\right)\right](a) - D_1(a)\boldsymbol{\Psi}_1 + \left[D_2(a) - D_1^2(a)\right]\boldsymbol{\Psi}_2 + \mathcal{V}[\mathbf{x}_\mathrm{ML}](a)\right]\right\}. \tag{A.13}$$

A.2. Time stepping with COCA

In this paper, we adopt the second order symplectic 'kick-drift-kick' algorithm, also known as the leapfrog scheme (e.g. Birdsall & Langdon 1985), to integrate the equations of motion, for a series of $n+1$ time steps $t(a)$ between $t_0 = t(a_\mathrm{i})$ and $t_{n+1} = t(a_\mathrm{f})$. This algorithm relies on integrating the model equations on small time steps and approximating the momenta and accelerations that appear in the integrands (the part between curly brackets in the model equations) by their value at some time within the interval.

The discrete versions of the COCA model equations (Eqs. (A.10)–(A.11) or (A.12)–(A.13)) give the Drift and Kick operators for COCA:

$$\mathrm{D}(t_\mathrm{i}^\mathrm{D}, t_\mathrm{f}^\mathrm{D}, t^\mathrm{K}): \quad \mathbf{x}(t_\mathrm{i}^\mathrm{D}) \mapsto \mathbf{x}(t_\mathrm{f}^\mathrm{D}) = \mathbf{x}(t_\mathrm{i}^\mathrm{D}) + \alpha_\mathbf{p}(t_\mathrm{i}^\mathrm{D}, t_\mathrm{f}^\mathrm{D}, t^\mathrm{K})\mathbf{p}_\mathrm{res}\left(t^\mathrm{K}\right) - \left[D_1\right]_{t_\mathrm{i}^\mathrm{D}}^{t_\mathrm{f}^\mathrm{D}}\boldsymbol{\Psi}_1 + \left[D_2\right]_{t_\mathrm{i}^\mathrm{D}}^{t_\mathrm{f}^\mathrm{D}}\boldsymbol{\Psi}_2 + \alpha_\mathbf{p}(t_\mathrm{i}^\mathrm{D}, t_\mathrm{f}^\mathrm{D}, t^\mathrm{K})\mathbf{p}_\mathrm{ML}\left(t^\mathrm{K}\right), \tag{A.14}$$

$$\mathrm{K}(t_\mathrm{i}^\mathrm{K}, t_\mathrm{f}^\mathrm{K}, t^\mathrm{D}): \quad \mathbf{p}_\mathrm{res}(t_\mathrm{i}^\mathrm{K}) \mapsto \mathbf{p}_\mathrm{res}(t_\mathrm{f}^\mathrm{K}) = \mathbf{p}_\mathrm{res}(t_\mathrm{i}^\mathrm{K}) + \beta_\delta(t_\mathrm{i}^\mathrm{K}, t_\mathrm{f}^\mathrm{K}, t^\mathrm{D}) \times \left\{\left[\boldsymbol{\nabla}\left(\Delta^{-1}\delta\right)\right](t^\mathrm{D}) - D_1(t^\mathrm{D})\boldsymbol{\Psi}_1 + \left[D_2(t^\mathrm{D}) - D_1^2(t^\mathrm{D})\right]\boldsymbol{\Psi}_2 + \mathbf{g}_\mathrm{ML}(t^\mathrm{D})\right\}. \tag{A.15}$$

Using Eqs. (A.10) and (A.11), the standard discretisation of operators (Quinn et al. 1997) gives the time prefactors as (Leclercq et al. 2020 Eq. (B3)),

$$\alpha_{\mathbf{p}}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K}) \equiv \int_{t_{\rm i}^{\rm D}}^{t_{\rm f}^{\rm D}} \mathcal{D}(\tilde{t})\,\mathrm{d}\tilde{t} = \int_{t_{\rm i}^{\rm D}}^{t_{\rm f}^{\rm D}} \frac{\mathrm{d}\tilde{t}}{\tilde{t}^2\mathcal{H}(\tilde{t})}, \quad \beta_\delta(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D}) \equiv \int_{t_{\rm i}^{\rm K}}^{t_{\rm f}^{\rm K}} \mathcal{K}(\tilde{t})\,\mathrm{d}\tilde{t} = -\frac{3}{2}\Omega_{\rm m}^{(0)}\mathcal{H}^{(0)2}\int_{t_i^{\rm K}}^{t_f^{\rm K}} \frac{\mathrm{d}\tilde{t}}{\tilde{t}\mathcal{H}(\tilde{t})}. \tag{A.16}$$

The arbitrary function u of a appearing in Eqs. (A.12) and (A.13) can be used to improve upon the standard discretisation of operators (Tassev et al. 2013, Appendix A). Indeed, if during the time step, the terms between brackets in Eqs. (A.12) and (A.13) are closer to constants than the terms between brackets in Eqs. (A.10) and (A.11), the approximation will hold better. Therefore, using Eqs. (A.12) and (A.13), the modified discretisation of operators gives the time prefactors, for any positive function u of t, as (Leclercq et al. 2020, Eq. (B11)),

$$\alpha_{\mathbf{p}}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K}) \equiv \frac{1}{u(t^{\rm K})}\int_{t_{\rm i}^{\rm D}}^{t_{\rm f}^{\rm D}} \mathcal{D}(\tilde{t})u(\tilde{t})\,\mathrm{d}\tilde{t}, \quad \beta_\delta(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D}) \equiv \left[u(t_{\rm f}^{\rm K}) - u(t_{\rm i}^{\rm K})\right] \times \frac{\mathcal{K}(t^{\rm D})}{[\mathrm{d}u(t)/\mathrm{d}t]\,(t^{\rm D})}. \tag{A.17}$$

In this paper, consistently with earlier literature, we use $u(t) \equiv a^{n_{\rm LPT}}$ with $n_{\rm LPT} = -2.5$ (Tassev et al. 2013; Leclercq et al. 2020).

The ML frame of reference gives particles an acceleration $\mathbf{g}_{\rm ML}(t^{\rm D})$ which should satisfy

$$\int_{t_{\rm i}^{\rm K}}^{t_{\rm f}^{\rm K}} \mathcal{K}(t)\mathcal{V}[\mathbf{x}_{\rm ML}](t)\,\mathrm{d}t = \int_{t_{\rm i}^{\rm K}}^{t_{\rm f}^{\rm K}} \frac{\mathrm{d}u(t)}{\mathrm{d}t} \times \left\{\frac{\mathcal{K}(t)}{\mathrm{d}u(t)/\mathrm{d}t}\mathcal{V}[\mathbf{x}_{\rm ML}](t)\right\}\mathrm{d}t \approx \beta_\delta(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})\mathbf{g}_{\rm ML}(t^{\rm D}). \tag{A.18}$$

In the standard discretisation, the integral can be approximated by using the value of $\mathcal{V}[\mathbf{x}_{\rm ML}](t)$ at $t^{\rm D}$ (assuming it is constant during the time step), giving $\beta_\delta(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})\mathcal{V}[\mathbf{x}_{\rm ML}](t^{\rm D})$ with the definition of $\beta_\delta(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})$ given in Eq. (A.16). In the modified discretisation, the integral can be approximated using the value of $\frac{\mathcal{K}(t)}{\mathrm{d}u(t)/\mathrm{d}t}\mathcal{V}[\mathbf{x}_{\rm ML}](t)$ at $t^{\rm D}$, giving also $\beta_\delta(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})\mathcal{V}[\mathbf{x}_{\rm ML}](t^{\rm D})$ but with the definition of $\beta_\delta(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})$ given in Eq. (A.17). In both cases, we get

$$\mathbf{g}_{\rm ML}(t^{\rm D}) \equiv \mathcal{V}[\mathbf{x}_{\rm ML}](t^{\rm D}). \tag{A.19}$$

But the integral is also:

$$\int_{t_{\rm i}^{\rm K}}^{t_{\rm f}^{\rm K}} \mathcal{K}(t)\mathcal{V}[\mathbf{x}_{\rm ML}](t)\,\mathrm{d}t = \int_{t_{\rm i}^{\rm K}}^{t_{\rm f}^{\rm K}} -\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{\mathcal{D}(t)}\frac{\mathrm{d}\mathbf{x}_{\rm ML}}{\mathrm{d}t}\right)\mathrm{d}t = \int_{t_{\rm i}^{\rm K}}^{t_{\rm f}^{\rm K}} -\frac{\mathrm{d}\mathbf{p}_{\rm ML}}{\mathrm{d}t}\,\mathrm{d}t = \mathbf{p}_{\rm ML}(t_{\rm i}^{\rm K}) - \mathbf{p}_{\rm ML}(t_{\rm f}^{\rm K}), \tag{A.20}$$

which gives the alternative form

$$\mathbf{g}_{\rm ML}(t^{\rm D}) \equiv \frac{1}{\beta_\delta(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})}\left[\mathbf{p}_{\rm ML}(t_{\rm i}^{\rm K}) - \mathbf{p}_{\rm ML}(t_{\rm f}^{\rm K})\right]. \tag{A.21}$$

As such, to use the COCA Kick and Drift operators (Eqs. (A.14) and (A.16)), it is not necessary to emulate both $\mathbf{p}_{\rm ML}$ and $\mathbf{g}_{\rm ML}$, but only a single emulator is needed (for $\mathbf{p}_{\rm ML}$), which is evaluated at the kick time steps.

In the end, the time evolution between t_0 and t_{n+1} is achieved by applying the following operator to the initial state $\{\mathbf{x}(t_0), \mathbf{p}(t_0)\}$:

$$\mathrm{L}_+(t_{n+1})\mathrm{E}(t_{n+1}, t_0)\mathrm{L}_-(t_0), \tag{A.22}$$

where $\mathrm{E}(t_{n+1}, t_0)$ is the operator given by (see Fig. 2)

$$\mathrm{K}(t_{n+1/2}, t_{n+1}, t_{n+1})\mathrm{D}(t_n, t_{n+1}, t_{n+1/2})\left[\prod_{i=0}^{n}\mathrm{K}(t_{i+1/2}, t_{i+3/2}, t_{i+1})\mathrm{D}(t_i, t_{i+1}, t_{i+1/2})\right]\mathrm{K}(t_0, t_{1/2}, t_0), \tag{A.23}$$

and $L_\pm$ will be defined in Eq. (A.38).

A.3. Generic Drift and Kick operators for PM, COLA and COCA

The difference between the COCA Kick and Drift operators and the corresponding COLA operators (Leclercq et al. 2020, appendices A and B) is the last term in each operator. Therefore, we can introduce generic operators, valid for both COLA and COCA: for any external momentum $\mathbf{p}_{\rm ext}$ and acceleration $\mathbf{g}_{\rm ext}$,

$$\begin{aligned}\mathrm{D}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K}): \quad & \mathbf{x}(t_{\rm i}^{\rm D}) \mapsto \mathbf{x}(t_{\rm f}^{\rm D}) = \mathbf{x}(t_{\rm i}^{\rm D}) + \alpha_{\mathbf{p}}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K})\mathbf{p}_{\rm res}\left(t^{\rm K}\right) + \alpha_{\rm LPT1}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K})\mathbf{\Psi}_1 + \alpha_{\rm LPT2}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K})\mathbf{\Psi}_2 \\ & \quad + \alpha_{\rm ext}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K})\mathbf{p}_{\rm ext}\left(t^{\rm K}\right), \end{aligned} \tag{A.24}$$

$$\begin{aligned}\mathrm{K}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D}): \quad & \mathbf{p}_{\rm res}(t_{\rm i}^{\rm K}) \mapsto \mathbf{p}_{\rm res}(t_{\rm f}^{\rm K}) = \mathbf{p}_{\rm res}(t_{\rm i}^{\rm K}) + \beta_\delta(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})\mathbf{g}_\delta(t^{\rm D}) + \beta_{\rm LPT1}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})\mathbf{\Psi}_1 + \beta_{\rm LPT2}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})\mathbf{\Psi}_2 \\ & \quad + \beta_{\rm ext}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})\mathbf{g}_{\rm ext}(t^{\rm D}), \end{aligned} \tag{A.25}$$

where

$$\alpha_{\rm LPT1}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K}) \equiv -\left[D_1\right]_{t_{\rm i}^{\rm D}}^{t_{\rm f}^{\rm D}}, \tag{A.26}$$

$$\alpha_{\rm LPT2}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K}) \equiv \left[D_2\right]_{t_{\rm i}^{\rm D}}^{t_{\rm f}^{\rm D}}, \tag{A.27}$$

$$\alpha_{\rm ext}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K}) \equiv \alpha_{\mathbf{p}}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K}) \text{ for COCA or 0 for COLA}, \tag{A.28}$$

$$\mathbf{p}_{\rm ext}(t^{\rm K}) = \mathbf{p}_{\rm ML}(t^{\rm K}) \text{ for COCA or } \mathbf{0} \text{ for COLA}, \tag{A.29}$$

$$\mathbf{g}_{\delta}(t^{\rm D}) \equiv \left[\boldsymbol{\nabla}\left(\Delta^{-1}\delta\right)\right](t^{\rm D}), \tag{A.30}$$

$$\beta_{\rm LPT1}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D}) = -\beta_{\delta}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D}) D_1(t^{\rm D}), \tag{A.31}$$

$$\beta_{\rm LPT2}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D}) = \beta_{\delta}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D}) \left[D_2(t^{\rm D}) - D_1^2(t^{\rm D})\right], \tag{A.32}$$

$$\beta_{\rm ext}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D}) = \beta_{\delta}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D}) \text{ for COCA or 0 for COLA}, \tag{A.33}$$

$$\mathbf{g}_{\rm ext}(t^{\rm D}) = \mathbf{g}_{\rm ML}(t^{\rm D}) \text{ for COCA or } \mathbf{0} \text{ for COLA}. \tag{A.34}$$

We note that these operators also remain valid for a standard PM algorithm, by setting $\alpha_{\rm LPT1}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K})$, $\alpha_{\rm LPT2}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K})$, $\beta_{\rm LPT1}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})$, and $\beta_{\rm LPT2}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})$ to zero.

A.4. Machine learning prediction of the frame of reference in COCA

The goal in COCA is to find the frame of reference in which $\mathbf{p}_{\rm res}$ is as small as possible. Therefore, the machine needs to predict:

1. At any "kick" time step $t^{\rm K}$,

$$\mathbf{p}_{\rm ML}\left(t^{\rm K}\right) \equiv \frac{1}{\alpha_{\mathbf{p}}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K})} \left[\mathbf{x}(t_{\rm f}^{\rm D}) - \mathbf{x}(t_{\rm i}^{\rm D}) - \alpha_{\rm LPT1}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K})\boldsymbol{\Psi}_1 - \alpha_{\rm LPT2}(t_{\rm i}^{\rm D}, t_{\rm f}^{\rm D}, t^{\rm K})\boldsymbol{\Psi}_2\right] \equiv \mathbf{p}_{\rm res}^{\rm COLA}\left(t^{\rm K}\right), \tag{A.35}$$

   that is the momentum residual $\mathbf{p}_{\rm res}^{\rm COLA}\left(t^{\rm K}\right)$ of COLA (Leclercq et al. 2020, Eq. (B5)).

2. At any "drift" time step $t^{\rm D}$,

$$\mathbf{g}_{\rm ML}(t^{\rm D}) \equiv -\frac{1}{\beta_{\delta}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})} \left[\beta_{\delta}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})\mathbf{g}_{\delta}(t^{\rm D}) + \beta_{\rm LPT1}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})\boldsymbol{\Psi}_1 + \beta_{\rm LPT2}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})\boldsymbol{\Psi}_2\right] \tag{A.36}$$

$$= \frac{1}{\beta_{\delta}(t_{\rm i}^{\rm K}, t_{\rm f}^{\rm K}, t^{\rm D})} \left[\mathbf{p}_{\rm res}^{\rm COLA}(t_{\rm i}^{\rm K}) - \mathbf{p}_{\rm res}^{\rm COLA}(t_{\rm f}^{\rm K})\right] \equiv -\mathbf{g}_{\rm res}^{\rm COLA}(t^{\rm D}), \tag{A.37}$$

   that is the residual acceleration $\mathbf{g}_{\rm res}^{\rm COLA}(t^{\rm D})$ of COLA (Leclercq et al. 2020, Eq. (B6)), up to a minus sign.

From Eqs. (A.35) and (A.37), we see that it is sufficient for the machine to predict the momentum residual $\mathbf{p}_{\rm res}^{\rm COLA}\left(t^{\rm K}\right)$ of COLA at any "kick" time step $t^{\rm K}$, as the accelerations $\mathbf{g}_{\rm res}^{\rm COLA}(t^{\rm D})$ can be derived from the momenta.

A.5. Initial and final momenta of particles in COCA

In the initial conditions, we have $\mathbf{p}(t_0) = \mathbf{p}_{\rm LPT}(t_0) + \mathbf{p}_{\rm ML}(t_0)$, which means that the momentum residual in the COCA frame of reference, $\mathbf{p}_{\rm res}(t_0) = \mathbf{p}(t_0) - \mathbf{p}_{\rm LPT}(t_0) - \mathbf{p}_{\rm ML}(t_0)$, should be initialised to zero. Furthermore, if initial conditions are generated with LPT, the ML contribution is $\mathbf{p}_{\rm ML}(t_0) = \mathbf{0}$ initially and we recover $\mathbf{p}_{\rm res}(t_0) = \mathbf{p}(t_0) - \mathbf{p}_{\rm LPT}(t_0)$, as in COLA.

At the end, the momentum $\mathbf{p}_{\rm LPT}(t_{n+1}) + \mathbf{p}_{\rm ML}(t_{n+1})$ of the COCA frame of reference has to be added to $\mathbf{p}_{\rm res}(t_{n+1})$ to recover the full momentum of particles, $\mathbf{p}(t_{n+1})$. These operations correspond respectively to the $\mathrm{L}_{-}(t_0) : \mathbf{p}(t_0) \mapsto \mathbf{p}_{\rm res}(t_0)$ and $\mathrm{L}_{+}(t_{n+1}) : \mathbf{p}_{\rm res}(t_{n+1}) \mapsto \mathbf{p}(t_{n+1})$ operators, given by

$$\mathrm{L}_{\pm}(t): \quad \mathbf{p}(t) \mapsto \mathbf{p}(t) \pm \mathbf{p}_{\rm LPT}(t) \pm \mathbf{p}_{\rm ML}(t) = \mathbf{p}(t) \pm \frac{1}{\mathcal{D}(t)} \left(-\frac{\mathrm{d}D_1}{\mathrm{d}t}\boldsymbol{\Psi}_1 + \frac{\mathrm{d}D_2}{\mathrm{d}t}\boldsymbol{\Psi}_2\right) \pm \mathbf{p}_{\rm ML}(t). \tag{A.38}$$