Fast emulation of cosmological density fields based on dimensionality reduction and supervised machine learning

Miguel Conceição1,2, Alberto Krone-Martins3,4, Antonio da Silva1,2, and Ángeles Moliné5

1 Departamento de Física, Faculdade de Ciências, Universidade de Lisboa, Campo Grande, 1749-016 Lisboa, Portugal
2 Instituto de Astrofísica e Ciências do Espaço, Faculdade de Ciências, Universidade de Lisboa, Campo Grande, 1749-016 Lisboa, Portugal
3 Donald Bren School of Information and Computer Sciences, University of California, Irvine, CA 92697, USA
4 CENTRA, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisboa, Portugal
5 Departamento de Física, ETSI Navales, Universidad Politécnica de Madrid, Avda. de la Memoria 4, 28040, Madrid, Spain

Received 24 April 2023 / Accepted 21 August 2023

ABSTRACT

N-body simulation is the most powerful method for studying the nonlinear evolution of large-scale structures. However, these simulations require a great deal of computational resources, making their direct adoption unfeasible in scenarios that require broad explorations of parameter spaces. In this work we show that it is possible to perform fast dark matter density field emulations with competitive accuracy using simple machine learning approaches. We built an emulator based on dimensionality reduction and machine learning regression combining simple principal component analysis and supervised learning methods. For the estimations with a single free parameter we trained on the dark matter density parameter, \( \Omega_m \), while for emulations with two free parameters we trained on a range of \( \Omega_m \) and redshift. The method first adopts a projection of a grid of simulations on a given basis. Then, a machine learning regression is trained on this projected grid. Finally, new density cubes for different cosmological parameters can be estimated without relying directly on new N-body simulations by predicting and de-projecting the basis coefficients. We show that the proposed emulator can generate density cubes at nonlinear cosmological scales with density distributions within a few percent compared to the corresponding N-body simulations. The method enables gains of three orders of magnitude in CPU run times compared to performing a full N-body simulation while reproducing the power spectrum and bispectrum within \( \sim 1\% \) and \( \sim 3\% \), respectively, for the single free parameter emulation and \( \sim 5\% \) and \( \sim 15\% \) for two free parameters. This can significantly accelerate the generation of density cubes for a wide variety of cosmological models, opening doors to previously unfeasible applications, for example parameter and model inferences at full survey scales, such as the ESA/NASA Euclid mission.

Key words. large-scale structure of Universe – cosmology: miscellaneous – methods: statistical – methods: numerical

1. Introduction

Numerical N-body simulation is arguably the most powerful method to describe the formation and evolution of cosmological structure, especially at late times when the growth of perturbations, driven by gravitational collapse, becomes highly nonlinear. These simulations provide high-accuracy predictions on how the dark matter (DM) density and velocity fields evolve, allowing the identification of model signatures and observables that can be compared against observations. This process often requires the generation of a very large number of computationally intensive simulation runs, posing a prohibitive bottleneck on the number of simulated models that can be explored in the preparation and scientific exploitation of modern cosmological surveys. This penalizes model inference studies, specially using Bayesian methods considering the shear data volume of modern sky surveys such as ESA/Gaia (Gaia Collaboration 2016, 2023) and ESA/Euclid (Laureijs et al. 2011; Euclid Collaboration 2022), and time-resolved surveys such as the Zwicky Transient Facility (Bellm & Kulkarni 2017) and LSST/Rubinkllk (Ivezić et al. 2019).

One typical way to address these types of challenges is through the adoption of simulation emulators. These emulators try to reproduce the output of a simulator by the adoption of multiple regression methodologies (e.g., O’Hagan & Kingman 1978). Gaussian processes, for instance, have been used in such scenarios (e.g., Curran et al. 1988, 1991). More recently, attention has been shifted to the machine learning (ML) approach, where deep models populate most of the density field emulation landscape.

Emulation of dark matter density fields with deep learning techniques such as the Generative Adversarial Networks (GANs) or Variational Auto-Encoders (VAEs) methods have been proposed, in 2D (e.g., Rodríguez et al. 2018; Ullmo et al. 2021) and 3D (e.g., Perraudin et al. 2019). These methods show that it is possible to use deep learning to emulate DM density fields using the Lagrangian approach (e.g., He et al. 2019; Giussarma et al. 2023; Alves de Oliveira et al. 2020; Kodi Ramanah et al. 2020; Jamieson et al. 2023), instead of the less general Eulerian approach. The emerging picture is that deep learning models tend to efficiently emulate DM fields, but generally require large training sets. Their accuracy starts to degrade below scales \( k \approx 1\,h\text{Mpc}^{-1} \), that is, slightly inside the strong nonlinear regime of structure formation. Other works (e.g., Conceição et al. 2021, 2022) focused on the emulation and resolution enhancement of N-body simulations using these methods with minimal computational resources.

Although the deep learning approach enables the decrease in time and computational resources needed to obtain an output similar to that of an N-body simulation, it still remains a...
reasonably complex approach that is often difficult to interpret. Since the $N$-body density fields usually consist of extremely large arrays that contain millions of matter density values with a fair degree of nonlinear behavior, it is compelling to think that only deep models should be able to handle such data. However, considering that we are dealing with nonrandom data with a high degree of symmetry brought by the cosmological principle, it should be possible to take advantage of the redundancy in this data, compressing it in a more compact basis. Using this reduced data representation, it becomes more feasible to apply simpler machine learning models, decreasing further the amount of computational resources and time needed to obtain the outputs, and hopefully increasing the interpretability of the models.

In this work we propose a methodology that aims to achieve these goals. Our methodology first adopts a simple data projection method, principal component analysis (PCA; Pearson 1901; Hotelling 1933; Jollife 2002), to project the $N$-body training data into a smaller basis of coefficients called Principal Components (PCs), and then uses the transformed representation as an input to a set of simple and interpretable supervised learning methods: random forests (RF; Breiman 2001), extremely randomized trees (ERTs; Geurts et al. 2006), shallow neural networks (NNs; McCulloch & Pitts 1943) and support vector machines (SVMs; Cortes & Vapnik 1995).

We tested our methodology in two different scenarios. In the first we performed the emulations using the dark matter density ($\Omega_m$) as a single free parameter. In the second scenario we slightly increased our dataset and included the redshift $z$ as an additional free parameter, enabling us to test the generalization capacity of our models under higher-dimensional contexts. To gauge the performance of the trained models and of our proposed methodology in general, we compare the power spectrum and bispectrum of our emulations with that of an $N$-body simulation generated for the same cosmology and initial conditions. In this way, we ensure that our emulated density fields have statistical properties as close to those of the true simulation as possible, on large and small scales.

In this paper we present our proposed methodology and the adopted comparison metrics in Sect. 2. We show the results of its application to emulate $N$-body simulation density fields in Sect. 3. Then, we present a brief discussion of the computational efficiency of our method in Sect. 4. Finally, we present our conclusions in Sect. 5.

2. Methodology

A consequence of the highly redundant amount of information present within the large-scale structure of the Universe is that this information can be highly compressible. Accordingly, the density field emulator design described in this section is based first on the application of a dimensionality reduction method and then on a supervised machine learning method. These methods require the construction of a training (and test) set of $N$-body simulations and evaluation metrics that we also describe in this section.

2.1. The density field emulator

2.1.1. PCA projection

We started from a library of $k$ density fields, each created from an independent $N$-body simulation with parameters $D_l$. Each cube of size $n^3$ of this library is rearranged into a row of a matrix $M_{\text{train}}$, with $k$ rows and $n^3$ columns. The first step in our methodology was to solve a data compression problem: how to find a sparse representation $S_{\text{com}}$ of the matrix $M$, with $l \ll n$ and $l \leq k$. We applied PCA, which is a very simple and well-established methodology adopted in multiple contexts in astronomy (e.g., Steiner et al. 2009; Ishida & de Souza 2011; Krone-Martins & Moutinho 2014; Euclid Collaboration 2021; Conceição et al. 2021, 2022) to perform this compression. We note, though, that although we adopted PCA here for simplicity, any other invertible dimensionality reduction or compression method could be used. To perform the PCA, we adopted a singular value decomposition. It decomposes the $M$ matrix in three matrices $M = UZV^T$, where $U$ is an unitary $k \times k$ matrix, $Z$ is a diagonal matrix with the singular values, and $V$ is a $n^3 \times k$ matrix containing the eigenvectors of the covariance matrix in each column, which are the principal directions. The principal components (e.g., $MV$ or equivalently $UZ$) are obtained by projecting the data in the principal directions. The crucial step when performing the dimensionality reduction is to select the first $l$ columns of $U$ and the left-right $l \times l$ block of $Z$. Their product gives us our $S_{\text{com}}$ matrix with the first $l$ principal components, capturing most of the data variance. The reconstruction or decomposition can be performed with the first $l$ PCs by multiplying $S$ by the transpose of the matrix $V$, $V^T$.

2.1.2. Supervised learning regression

The second step of our methodology is to solve a regression problem, that is, to find a map $f : D \rightarrow S$. Thus, given any set $i$ of cosmological parameters $D_i$, it is possible to estimate the compressed field $S_i$, which after decompression generates an estimation of the density field $M_i$ as if we had executed a full $N$-body simulation with the parameters $D_i$. To construct this map, we studied four supervised machine learning algorithms in this work: ERT, RF, NN and SVM. Our motivation to use multiple methods is to compare the emulation yields obtained with them in this first exploratory study, since they could, in principle, present better results in different subdomains of the parameter space and in different scales of the density field. That could justify, in a future study, the construction of more precise ensembles (e.g., Mendes-Moreira et al. 2012).

The adoption of supervised learning methods in astronomy has a long history (e.g., Djorgovski et al. 2022). Neural networks (McCulloch & Pitts 1943) and support vector machines (Cortes & Vapnik 1995) are classical methods that have been successfully used in multiple contexts. For instance, before the recent boom in citations to computer science works in astronomical papers (Veneri et al. 2022), mostly due to the spread of deep learning approaches, NNs had already been used for almost four decades in astronomy, in optimization (Jeffrey & Rosner 1986), in heliophysics (Koons & Gorney 1990), adaptive optics (Angel et al. 1990; Sandler et al. 1991), star–galaxy separation (Odewahn et al. 1992), and galaxy classification (Storrie-Lombardi et al. 1992). Support vector machines, on the other hand, started to be used almost exclusively in extragalactic contexts in the 2000s (e.g., Zhang et al. 2002; Wadadekar 2005; Tsalmantza et al. 2007; Huertas-Company et al. 2008; Krone-Martins et al. 2008; Bailar-Jones et al. 2008), but they were also adopted in the classification of transients (Mahabal et al. 2008), stellar classification (e.g., Smith et al. 2010), interstellar medium structure (Beaumont et al. 2011), and heliophysics (e.g., Alipour et al. 2012), among others.

Random Forests (Ho 1995; Breiman 2001) started to be used in astronomy in the early 2000s. Like SVMs, they were
first used in an extragalactic context to search for quasars (Breiman et al. 2003) and supernovae (Bailey et al. 2007), to identify high-energy sources (Scaringi et al. 2008), and to perform photometric redshift estimation (Carliles et al. 2010), but they later found other applications such as variable star classification (Richards et al. 2011; Dubath et al. 2011), feature selection for stellar membership analysis (Sarro et al. 2014), anomaly detection (Nun et al. 2014), and young stellar object classification including missing data (Ducourant et al. 2017). Finally, the last method adopted here, extremely randomized trees (Geurts et al. 2006), is a supervised learning method that is a variant of the more commonly used RF. Although not as widely adopted in astronomy as the other methods, in the past decade ERTs have been successfully used in the context of stellar astrophysical parameter determination (Bailer-Jones et al. 2013; Delchambre 2018), quasar selection (Graham et al. 2014), prediction of galaxy properties from dark matter halo simulations (Kamdar et al. 2016), and the discovery of new gravitationally lensed quasar candidates (Martins et al. 2018; Delchambre et al. 2019). Like RFs, ERTs are based on ensembles of decision trees, but they split the trees differently: RFs split trees deterministically, while the ERT splits are chosen entirely at random (Geurts et al. 2006).

2.1.3. Algorithm optimization

As with most machine learning methods, supervised learning algorithms depend on hyper-parameters that can be optimized. For each method, balancing hyper-parameters can be important to improve the speed and performance of the method. To optimize them, we explored each method's hyper-parameter space with different optimization schemes. We chose to experiment with the standard grid-search approach (e.g., Lecun et al. 1998; Bergstra & Bengio 2012), bootstrap aggregating (e.g., Breiman 1996), and k-fold cross-validation (K-CV; e.g., Hastie et al. 2009) optimization methods. Each optimization method provides a different optimal model. To discriminate between them, we compared the best emulation from each method with the corresponding ground truth test simulation using the cosmological power spectrum (see Sect. 2.4). For this, we define the power spectrum distance ratio metric as

\[ D = \sum_{i=1}^{n} \left| 1 - \frac{P_{\text{est}}^i}{P_{\text{sim}}^i} \right| , \]

where \( n \) is the power spectrum vector size, \( P_{\text{est}}^i \) and \( P_{\text{sim}}^i \) are respectively the estimated and simulated power spectrum of the \( i \)th wave mode, \( k \). Consequently, we take as the best performing optimization method the one providing the smallest \( D \).

The application of these methods to the different supervised models (i.e., of our emulator involves optimizations over different sets of hyper-parameters. Our choice of hyper-parameters and methodology, in each case, was as follows.

For the RF method, the hyper-parameter optimization was on the number of trees in the ensemble (\( n_{\text{tree}} \)), the minimum number of observations allowed in the terminal leaves of the trees (\( \text{nodesize} \)), and the number of independent variables randomly sampled as candidates for each split (\( m_{\text{try}} \)). Therefore, the optimization of the last parameter is only possible for regression problems with more than one independent variable (i.e., in the case of emulations in \( \Omega_m \) and \( z \)). To perform the search over this hyper-parameter space, we first performed a simple grid-search over the number of trees for the values \( n_{\text{tree}} = \{500, 1000, 2000\} \), fixed the best performing value, and proceeded to optimize the remaining hyper-parameters, experimenting with both bootstrap and k-fold cross-validation. This procedure was implemented using the e1071 package (Meyer et al. 2021).

For the ERTs, our optimization approach is similar to the RF case, except that instead of optimizing \( \text{nodesize} \), we optimized the number of random splits considered at each partitioning node of the tree (\( \text{numRandomCuts} \)). Then the optimization proceeded in similar way to that of RF. First, by fixing \( n_{\text{tree}} \) with a grid-search over the same possibilities, followed by a search over \( m_{\text{try}} \) and \( \text{numRandomCuts} \) and experimenting with the same optimization schemes. This procedure was implemented using the caret package (Kuhn 2008).

For the NNs we decided to use a single-layer network as our aim is to study the simplest possible architectures. We optimized the parameter size, corresponding to the number of nodes in the single hidden layer, and the parameter decay, corresponding to a regularization constant in the weight decay formula, which constrains the number of free parameters (weights) in the NN model.

Finally, for the SVMs we first performed a grid-search over the different kernel possibilities, with the remaining hyper-parameters set to their default values. After finding the optimal kernel, we optimized the standard SVM hyper-parameters \( c \) and \( \gamma \).

2.2. Training and test datasets

One advantage of our emulation method is that it is a simple and generalizable approach to the emulation of \( N \)-body density fields. To demonstrate the flexibility of the method, we decided to make all computations on a simple computer system (a quad-core Intel core i5 with 16Gb of RAM and no discrete GPUs) running Linux. We therefore planned our set of \( N \)-body simulation runs to use for the training of our emulator, with size and resolution specifications capable of resolving a range of nonlinear evolving cosmological scales in the chosen platform.

Our simulations were generated for a family of cosmological models based on the flat cold dark matter (CDM), ACDM cosmology with Hubble parameter \( h = 0.7 \) and dark matter density parameter \( \Omega_m = 0.7, \Omega_k = 1 - \Omega_m, h \) where \( \Omega_m \) is the total matter parameter and \( \Omega_k \) is the energy density parameter associated with the cosmological constant. In this way the energy density of radiation is neglected and baryons are treated in the simulations as collisionless dark matter. The initial conditions matter power spectrum was modeled with the CDM BBKS transfer function with a shape parameter given by \( \Gamma = \Omega_m h \) (Bardeen et al. 1986; Sugiyama 1995) and a normalization controlled by the \( \sigma_8 \) parameter, which is the square root of the variance of the smoothed overdensity field on scales of \( 8 h^{-1} \) Mpc at present. The primordial power spectrum index of scalar perturbations was set to \( n_s = 1 \). The simulation runs were generated with the Hydra public code (Couchman et al. 1995) that implements an adaptive mesh refinement AP3M method to compute gravitational forces and evolves a set of CDM mass particles in serial mode. We used the cosmic initial conditions generator provided in the Hydra code package to generate fields for \( 160^3 \) particles of dark matter in a cubic volume with comoving length \( L = 100 h^{-1} \) Mpc on the side. In physical coordinates the simulation’s gravitational softening was held fixed to 25 h\(^{-1}\) kpc up to \( z = 1 \), and scaled as \( 50(1 + z)^{-1} h^{-1}\) kpc above this redshift. With this choice of parameters and force resolution, our simulations could in principle resolve scales, in high-density regions, within a minimum and maximum wave mode of about \( k_{\text{min}} \approx 0.06 h\text{Mpc}^{-1} \) and \( k_{\text{max}} \approx 30 h\text{Mpc}^{-1} \), respectively. All
our simulation runs share the same initial conditions seed and start from an initial snapshot realization at $z = 49$ that is evolved until $z = 0$.

Our training dataset consists of 23 simulations with $\Omega_m$ taking values in the interval $\Omega_m \in [0.05, 0.6]$ with a regular separation equal to $\Delta \Omega_m = 0.025$. All other parameters were held fixed, with the exception of the primordial shape parameter that varied according to $\Gamma = \Omega_m h$. For each of these runs we store ten snapshots (cubic volumes) containing particle positions and velocities at each redshift. Our test dataset consists of a single simulation with $\Omega_m = 0.31$ and the same parameter setup, to be used as the ground truth. Before using all these runs with our emulation pipeline we first needed to sample the dark matter densities on a regular cubic grid for all snapshots. Hydra is a particle-based code that follows volume elements (the particles) using a Lagrangean description, so it does not provide densities directly. To convert these into a Eulerian grid of densities, we adopted the mapping algorithm in da Silva et al. (2001) and Ramos et al. (2012). For the purpose of this work, we adopted a regular grid with $128^3$ voxels inside the simulation volume, which is grid choice that limits the resolution in our emulations to a maximum wave mode of about $k_{max} = 3 \, h \, \text{Mpc}^{-1}$. Finally, our simulation training set is therefore a template of simulations cubes, all with $128^3$ voxels, spanning a range of 23 values of $\Omega_m$ and 10 values of $z$ (which amounts to a total of 230 simulation cubes). In practice, we work with different subsets of these data for the cases of emulations with one or two free parameters, as described below.

### 2.3. Emulation pipeline

The first step in our emulation pipeline is to project our training dataset with PCA. We concatenate simulation arrays as vectors containing the base-10 logarithm of the density cubes. The vectors are then put in a matrix form before applying the PCA method with the data re-scaled and centered.

For emulations with a single free parameter at a fixed redshift (i.e., with $\Omega_m$), this procedure leads to 23 PCs for each simulation. We chose to preserve the total number of PCs in order to maintain the total amount of information of each simulation. After the PCA was implemented we obtained a new training dataset, with 23 PCs for each of the 23 simulations, represented in a $23 \times 23$ matrix. Once the training data was projected in its principle component representation we used the PCs to build and optimize the supervised learning algorithms. Since the density fields are represented by PCs, and our aim was to estimate the field itself, the training was performed using the PCs as dependent variables and $\Omega_m$ as independent variables. To achieve this, we created a 23 iteration loop where in each iteration our algorithms build a regression model accounting for that specific PC and the same constant set of $\Omega_m$ values. For the case of the NNs, the data was re-scaled to the Alipour et al. (2012) range to reduce the chances of getting stuck in local minima. After obtaining the new PCs, the final estimated cube was obtained by de-projecting them onto the original representation using the inverse de-projection operations:

$$M_{\text{orig}} = 10^{8L^5 s^2 + c^2},$$

where $s$, $L$, $c$, and $e$ are the scores, loadings, scaling, and centering matrices and vectors, respectively.

For emulations with two parameters, the main difference with the one free parameter case is the size of the training dataset. To incorporate the training in redshift for the chosen computer platform, we decided to use only four redshifts, $z = [0, 0.25, 0.75, 1]$ from our initial dataset. This translated into including four of the ten generated $N$-body snapshots in the training dataset, for each of the 23 simulations. For each of these redshifts we kept the same 23 $\Omega_m$ density cubes we had in the case of regressions with a single parameter, $\Omega_m$. Our training dataset therefore had $23 \times 4 = 92$ simulation cubes, and we kept the cube with $\Omega_m = 0.31$ and $z = 0.5$ to compare emulation results against the corresponding simulation ground truth. The whole pipeline took a significantly longer time to run in this case: our dataset was significantly larger as it involved working on the basis of 92 PCs, instead of only 23 for the single free parameter emulation. Once again we chose to maintain all of the 92 PCs to perform the ML regressions, so in this case our training dataset was contained in a $23 \times 92$ matrix. We note that in this case we were making predictions using $z$ and $\Omega_m$, while in the case of the $\Omega_m$ single free parameter emulations we only used 23 simulations for training at each redshift.

Our emulation pipeline was fully written in the R programming language (R Core Team 2021). This includes the module of PCA decomposition, implementation of the different supervised regression algorithms, optimization with the grid-search, bootstrap, aggregating, and k-fold cross-validation methods, and finally density cube emulations using up to two free parameters.

### 2.4. Evaluation statistics

To assess the performance of our emulation method, we compare in Sect. 3 the power spectrum and bispectrum of our emulated volumes with those of the ground truth simulations. The power spectrum statistic is also used as part of our optimization of the method, as described in Sect. 2.1.3.

The power spectrum is the Fourier transform of the two-point correlation function

$$\xi(r,z) = \frac{V}{(2\pi)^3} \int \langle \delta(r, k, z) \rangle d^3 k,$$

where $V$ is the box volume and $P(k,z) = \langle |\delta(k, z)|^2 \rangle = \frac{1}{V} \int \xi(r,z) e^{ikr} d^3 r$

is the power spectrum. For a Gaussian distribution of densities, $P(k,z)$ provides a complete statistical description of the density field. However, the density field in $N$-body simulations is not Gaussian due to the nonlinear growth of cosmic structure. We compute the nonlinear power spectrum of the emulations and ground truth simulations with the public version of the cosmo code (Colombi et al. 2009).

To measure departures from non-Gaussianity, we also computed the bispectrum of the density cubes. The bispectrum is associated with the Fourier transform of the three-point correlation function and, as a higher-order correlation function, it is sensitive to the nonlinear growth of cosmic structure. It is usually defined as $B$ (see, e.g., Sefusatti et al. 2006)

$$\langle \delta(k_1, \delta(k_2), \delta(k_3)) \rangle = \delta_P(k_1 + k_2 + k_3) B(k_1, k_2, k_3),$$

where $\delta_P$ is the Kronecker delta function, imposes a triangular relation between wave-mode vectors, $k_1 + k_2 = -k_3$. Here we adopt a configuration of wave modes where $k_1$ and $k_2$ have fixed magnitudes, $k_1 = |k_1| = 0.5 \, h \, \text{Mpc}^{-1}$ and
$k_2 = |k_2| = 0.6 \text{ h Mpc}^{-1}$, and $k_3 = |k_3|$ varies with the angle $\theta$ between $k_1$ and $k_2$. Our choice of wave-mode amplitudes differs from that in Kodi Ramanah et al. (2020) and Giusarma et al. (2023) because our simulation’s boxsize is smaller. This improves the statistics of our bispectrum determinations on the scales being probed by galaxy clustering surveys. Nevertheless, we checked that our bispectrum results agree with those in Kodi Ramanah et al. (2020) and Giusarma et al. (2023) for their choice of wave modes $k_1 = |k_1| = 0.15 \text{ h Mpc}^{-1}$ and $k_2 = |k_2| = 0.25 \text{ h Mpc}^{-1}$. In this work we use the Pylians3 public software\footnote{https://github.com/franciscovillaescusa/Pylians3} to compute the bispectra presented in Sect. 3.

3. Results

In this section we present our main results. We show how each of our estimated density fields compares against the simulated density field using their power spectra and bispectra as comparison metrics. For the single free parameter case, we present the results concerning the implementation of our pipeline at four different redshifts.

3.1. The single free parameter case

For emulations with a single free parameter, we first show a visual comparison of one of our emulations (the best performing one) with the actual ground truth simulation in Fig. 1, where it is possible to see a remarkable similarity between the two representations. This figure shows the base-10 logarithm of the overdensity field using their power spectra and bispectra as comparison metrics. For the single free parameter case, we present the results comparing different emulator methods to the ground truth simulation and emulation volume represented in Fig. 1. The distributions (in the top panel) are clearly non-Gaussian and their ratios (in the bottom panel) indicate that the emulator performs best in the range of densities that are more frequent in the simulated volume. This result confirms expectations because the tails of the distributions are rare over- and underdensities (in high-density structures or deep voids) that are also underrepresented in the training dataset. Nevertheless, the agreement between the distributions is remarkable, especially if we take into account that this emulation was done with a training dataset with only 23 density cubes and a NN geometry with a single hidden layer. The differences are typically below 2.5\% in the range 0.01$(\Delta)$.100. Outside this range, where densities correspond to voxels inside collapsed halos and deep voids, the agreement between distributions is still better than 5\%.

Figure 3 shows the power spectrum from emulations with one free parameter, $\Omega_m = 0.31$, at fixed redshifts, $z = 0$ (top left), $z = 0.5$ (top right), $z = 1$ (bottom left), and $z = 10$ (bottom right). In each plot the top panel shows the power spectra of the ground truth simulation and our emulator implementations, featuring ERTs, NNs, RF, and SVMs as supervised learning algorithms. The bottom panel in each plot shows the ratio of the power spectrum of different emulator methods to the ground truth power spectrum. Overall, the NN is our best performing algorithm; it achieves an accuracy above 99\% at all redshifts and for most of the $k$ domain. Random Forests manages to show similar accuracy.
Fig. 3. Power spectrum emulations at redshifts $z = 0$ (top left), $z = 0.5$ (top right), $z = 1$ (bottom left), and $z = 10$ (bottom right) for model regressions with a single parameter, $\Omega_m = 0.31$. The top panel of each plot shows the power spectra obtained with our different emulation variants, extremely randomized trees (ET), neural networks (NN), Random Forests (RF), and support vector machines (SVM). The bottom panel in each plot shows the ratio of the power in each emulation to the power in the ground truth $N$-body simulation ($P_E(k) = \text{Estimated}; P_S(k) = \text{Simulated}$).

At all redshifts except at $z = 1$ where performance decreases slightly, but still achieves ratios above the 96% level at all $k$. The other algorithms have somewhat worse performance, particularly at $z = 0.5$ and $z = 1$. At $z = 10$ all algorithms show a strong agreement with the power spectrum of the ground truth simulation. This agrees with expectations because most of the resolved scales in our simulations are still evolving linearly at that epoch, and therefore strong underrepresented over- and underdensities do not have enough time to form by that redshift. In light of this argument one could expect larger power spectrum deviations.
as redshift decreases because progressively larger nonlinearities develop at low $z$. Our results seem to contradict this trend at redshift $z = 0.5$ and below, because the different power spectrum ratios in Fig. 3, generally, tend to approach 1 as redshift decreases. This apparent contradiction can still be understood as a consequence of the level of representation (amount) of nonlinearities in the training set if we take into consideration the dynamics of the $\Lambda$CDM model at $z < 1$. Below this redshift, the cosmological constant reverts the Hubble expansion rate from a decelerated phase, due to matter domination, into an accelerated expansion. As larger over- and underdensities develop inside our probed volume, their number is better represented in the training set below this redshift than at higher-$z$ (where structure grows in a decelerating matter-dominated universe).

Another interesting feature of our results is the decrease in power spectrum ratios at small $k$, that is, at large scales. This trend is common to all algorithms, and again can be interpreted as a lack of independent modes in the training dataset.

Figure 4 presents the results from our bispectrum analysis. As before, the figure shows bispectra and bispectra ratios at redshifts $z = 0$ (top left), $z = 0.5$ (top right), $z = 1$ (bottom left), and $z = 10$ (bottom right), for the $\Omega_m = 0.31$ cosmology. On the $x$-axis, $\theta$ is the angle between the wave modes $k_1$ and $k_2$ of our bispectra configuration (see Sect. 2.4). The relative performance of the algorithms is similar to the power spectrum results. The NN tends to be the best performing algorithm, followed closely by the RF implementation, whereas the other algorithms show lower performance. Our results also show that at the extremes of the $\theta$ domain the different algorithms also tend to underperform, especially at large $\theta$. The level of disagreement between emulated and ground truth bispectra is also greater than in the case of the power spectrum. The larger deviations from the ground truth are a consequence of the larger uncertainties, generally, associated with the measurement of tree-point, and other higher-order, correlation functions from a finite volume. On the other hand, the larger deviations at large $\theta$ are linked to the choice of our bispectrum configuration, where $k_1 = 0.5 \, h \, \text{Mpc}^{-1}$ and $k_2 = 0.6 \, h \, \text{Mpc}^{-1}$ were set to scales that are closer to the lower end of the $k$ domain in our simulations, and therefore correspond to larger scales that are less represented in our training set.

3.2. The two free parameters case

In an attempt to expand the cosmological parameter space of our emulation method and test it under higher-dimensional settings, we included the redshift as an additional free parameter in our emulation pipeline. Here we present results for emulations at $\Omega_m = 0.31$ and $z = 0.5$.

Figure 5 shows the power spectrum (top panel) and power spectrum ratios (bottom panel) of our emulator variants (ET, NN, RF, SVM) versus the ground truth simulations. As expected, there is a general decrease in performance of the algorithms due to the increased dimensionality of the problem. NN is no longer the best performing algorithm, with errors reaching 25% at around $k = 0.6 \, h \, \text{Mpc}^{-1}$. The RF and NN algorithms both overestimate the power spectrum in the range around $0.1 \, h \, \text{Mpc}^{-1} \leq k \leq 2 \, h \, \text{Mpc}^{-1}$, while SVM and ET consistently underestimate it.

For our emulation setup with two free parameters, the SVM is by far the best performing algorithm. It achieves an accuracy of more than 95% for $0.3 \, h \, \text{Mpc}^{-1} \leq k \leq 2 \, h \, \text{Mpc}^{-1}$, decreasing to about 90% at the high-$k$ (small-scale) extreme and ~85% at the low-$k$ (large-scale) extreme. This suggests an advantage of this algorithm for applications in multi-dimensional datasets.

in particular, due to its adoption of nonlinear kernels before the hyper-plane segregation.

The degradation of results with the increase in dimension of the emulation problem is a consequence of the curse of dimensionality (Bellman et al. 1957) that affects most optimization methods, and thus also supervised learning methods. This problem translates the need of increasing the size of the training set exponentially when adding dimensions in the feature space so that a similar performance to the lower-dimensional can be maintained. In this work our choice of computer platform limits the augmentation of the training set for emulations with two parameters to a factor of only four times the size of the training set of our one free parameter case. To maintain the high degree of accuracy of the single parameter case, we should further increase the size of the training dataset.

Our bispectrum results for emulations with two free parameters are shown in Fig. 6. As before, the top panels show bispectra curves, whereas the bottom panel shows bispectra ratios ($B_k(k) = \text{Estimated}, B_k(k) = \text{Simulated bispectra}$). A striking feature is that all algorithms except ET overestimate the bispectra, something that does not happen in the one parameter emulation scenario. Since ET is the only algorithm that considers only one of the independent variables, redshift or $\Omega_m$, the other algorithms may be overestimating the bispectrum due to a degeneracy between the two parameters. The self-similarity of cosmological structure induced by gravity-only simulations leads to some degree of degeneracy between the redshift epoch and $\Omega_m$ for a given scale of observation. A snapshot at high redshift with a given $\Omega_m$ displays a distribution of densities similar to a lower-redshift snapshot with a lower $\Omega_m$. This is an approximate argument that does not take into consideration the impact of $\Omega_m$ on the epoch of transition from matter to cosmological constant domination (which happens at different $z$ for different $\Omega_m$), but generally applies to simulation snapshots away from these transition redshifts.

These results further indicate that SVMs are the best method among those studied in this work since, as we can see, even when considering higher-order correlations it still out-performs the other algorithms. It is the only algorithm able to maintain an accuracy above 85% for the entire $\theta$ domain.

4. Discussion

In this section we discuss some of our key findings. We first discuss the computational efficiency of our methods, as measured by CPU run times. Subsequently, we analyze the algorithms employed in our investigation, providing a detailed assessment of their performance and potential for application in related research. Finally, we conclude by outlining possible avenues for improvements in our approach.

4.1. Efficiency

In Table 1 we present the CPU run times for the best performing algorithms in both emulation cases considered in this work. These cases correspond first to considering $\Omega_m$ as a single free parameter (first row of the table) and then to using both $\Omega_m$ and $z$ as free parameters (second row). For the single free parameter case, we present the results trained with the dataset of 23 simulation snapshots at $z = 0$. The run times shown in the table correspond to both the PC estimation and to the de-projection into the final emulated density field, and we also present the sum of the two as the “Total emulation time” column.
Fig. 4. Bipectrum emulations at redshifts $z = 0$ (top left), $z = 0.5$ (top right), $z = 1$ (bottom left), and $z = 10$ (bottom right) for model regressions with a single parameter, $\Omega_m = 0.31$. The top panel of each plot shows bispectra for our different emulation variants, extremely randomized trees (ET), neural networks (NN), Random Forests (RF), and support vector machines (SVM). The bottom panel in each plot shows the bispectra ratios from emulations, $B_E(\theta)$, and the $B_S(\theta)$ the ground truth simulation.

In terms of pure computational efficiency, the outcomes of our emulation pipeline seem highly satisfactory, particularly when considering that an $N$-body simulation of a similar scale requires 84 min to complete using the same system used for our emulation experiments. As exemplified in Table 1, the total emulation times for the single parameter scenario and two free parameters scenario were on the order of 0.8 and 4.5 s, respectively. This represents a substantial improvement in efficiency as
4.2. Algorithms

The supervised learning algorithms used in this study warrant some discussion, as some noteworthy observations can be made.

First, our results indicate that the RF algorithm was successful at reproducing the statistical properties of the density fields, particularly the non-Gaussian features, as evidenced by the bispectrum results in Fig. 4 for the single free parameter case. However, the opposite behavior is observed in the two free parameters scenario, where RF significantly underperforms in comparison to the other methods. In contrast, the ERT algorithm yields consistently disappointing results across both parametric contexts, emerging as one of the poorest performers, which we find surprising given its conceptual similarity to RF.

Second, the NN and SVM algorithms stand out as both demonstrate superior performance in their respective contexts. NN yields the best results for the one-dimensional $N$-body regression case, as depicted in Figs. 3 and 4. Meanwhile, SVM outperforms all the other methods in the multi-dimensional regression context. Considering that our current cosmological models incorporate multiple parameters beyond those examined in this first study (e.g., Hubble constant, baryonic matter density, power spectrum normalization, spectral index), SVM seems to be a promising method in the more realistic high-dimensional context.

However, these results do not yet allow for definitive conclusions regarding the potential of these supervised algorithms. It is important to keep in mind that our hyper-parameter optimization process may not have achieved a global minimum in the regression error metrics and the algorithm cost functions. As our simple search was restricted to a constrained and finite range of hyper-parameter values, there is a possibility that the algorithms were confined to a local minimum, with further potential for improvement through a better and larger exploration of the hyper-parameter space. Nonetheless, our findings offer some indications of the potential of the analyzed algorithms.

4.3. Future prospects

There are several potential avenues to improve or extend the methods used in this study. Some of these improvements could be implemented without significant modifications to the existing pipeline. For example, one option would be to expand the parameter space to include additional cosmological parameters such as the Hubble constant or spectral index. Naturally, this would also require an increase in the training set to sample the parameter space. Alternatively, the dark matter density fields could be emulated with the inclusion of hydrodynamics, which would similarly require a substantial increase in training data. While hydrodynamics are known to introduce highly nonlinear effects that may be difficult to reproduce, the overall structure of the pipeline should remain very similar, perhaps only requiring the adoption of more advanced dimensionality reduction methods.

On the other hand, methods similar to those proposed here could also be coupled directly within $N$-body codes, perhaps to accelerate certain aspects of the simulation, such as certain scales of the simulation. This would naturally require more extensive modification to the pipeline and of the $N$-body codes.

Finally, an interesting application of our proposed methods that we are currently investigating involves attempting to emulate the hydrodynamics directly from dark matter-only simulations. This requires some fundamental changes to the proposed

---

Table 1. CPU run times for the emulation step our proposed $N$-body density field emulation pipeline.

<table>
<thead>
<tr>
<th></th>
<th>Emulation</th>
<th>PC estimation</th>
<th>De-projection</th>
<th>Total emulation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega_m$ (NN)</td>
<td>0.077s</td>
<td>0.719s</td>
<td>0.796s</td>
<td></td>
</tr>
<tr>
<td>($\Omega_m, z$) (SVM)</td>
<td>0.082s</td>
<td>4.380s</td>
<td>4.462s</td>
<td></td>
</tr>
</tbody>
</table>
pipeline, but this exciting prospect would enable the production of emulated hydrodynamical N-body density fields with computational overheads constrained purely by the computational requirements of gravity simulations, which are naturally significantly smaller than those also considering hydrodynamics.

5. Conclusions

In this work we presented a methodology for fast and accurate emulations of full N-body dark matter density fields based on dimensionality reduction and supervised learning. In this first study, we applied our methodology in 128$^3$ density cells in a box of 100 $h^{-1}$ Mpc, using cosmological parameters as free parameters. We adopted four supervised machine learning methods, namely random forests, extremely randomized trees, support vector machines, and neural networks, in two different emulation scenarios using dark matter density and redshifts as free parameters.

In the first scenario we performed emulations solely based on $\Omega_m$. The proposed emulation pipeline was able to accurately reproduce the power spectrum of the corresponding simulated field with less than a 1% difference for $k > 0.3 h{\text{Mpc}}^{-1}$ at all redshifts. Furthermore, we achieved similar results when reproducing the bispectrum for all $\theta$ values, with less than a 3% difference for $z = 0$ and less than a 2% difference for the remaining redshifts. Our method was trained using 23 N-body simulations with the same reference for the remaining redshifts, but this exciting prospect would enable the production of an extremely accurate simulation.

In the second scenario, the emulator was extended to include redshift as a free parameter, thus slightly increasing the computational requirements of gravity simulations, which are naturally significant smaller than those also considering hydrodynamics.

References

Bellm, E., & Kulkarni, S. 2017, Nat. Astron., 1, 0071
Breiman, L. 1996, Mach. Learn., 24, 123
Conceição, M., Krone-Martins, A., & Da Silva, A. 2021, in 2021 IEEE 17th International Conference on eScience (eScience), 225
Conceição, M., Krone-Martins, A., & Da Silva, A. 2022, in 2022 IEEE 18th International Conference on eScience (eScience), 395
Jollife, I. T. 2002, Principal Component Analysis (Belin: Springer-Verlag)
Koons, H. C., & Gorney, D. J. 1990, EOS Trans., 71, 677

Acknowledgements. This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) through the Portuguese Strategic Programmes and the following Projects and Research Contracts: EXPL/FIS-AST/1368/2021, PTDC/FIS-AST/0005/2021, UIDB/00099/2020, UID/FIS/00099/2019, CERN/SC-PAR/007/2019, PTDC/FIS-OUT/29048/2017, PTDC/FIS-AST/31546/2017, IF/01135/2015, and SFRH/BPD/74697/2010. A.K.M. additionally acknowledges the support of the Caltech Division of Physics, Mathematics and Astronomy for hosting research leaves during 2017–2019 and 2020, when some of the ideas and codes underlying this work were initially developed. Finally, the authors thank Arlindo Trindade for useful discussions and Rafael S. de Souza for the useful comments during the defense of the Master thesis of M.C. and which also helped to improve this work.