Neural physical engines for inferring the halo mass distribution function

Tom Charnock, Guilhem Lavaux, Benjamin Wandelt, Supranta Sarma Boruah, Jens Jasche, Michael Hudson

Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **494,** 50–61 (2020)
Advance Access publication 2020 March 13

doi:10.1093/mnras/staa682

Neural physical engines for inferring the halo mass distribution function

Tom Charnock [1]★ Guilhem Lavaux [1,2] Benjamin D. Wandelt,[1,2,3]
Supranta Sarma Boruah [4,5] Jens Jasche[6] and Michael J. Hudson[5,7,8]

[1]*Sorbonne Université, CNRS, UMR 7095, Institut d'Astrophysique de Paris, 98 bis boulevard Arago, F-75014, Paris France*
[2]*Sorbonne Université, Institut Lagrange de Paris (ILP), 98 bis boulevard Arago, F-75014, Paris, France*
[3]*Center for Computational Astrophysics, Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA*
[4]*Department of Applied Mathematics, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada*
[5]*Waterloo Centre for Astrophysics, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada*
[6]*The Oskar Klein Centre, Department of Physics, AlbaNova University Center, Stockholm University, SE-106 91 Stockholm, Sweden*
[7]*Department of Physics and Astronomy, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada*
[8]*Perimeter Institute for Theoretical Physics, 31 Caroline Street North, Waterloo, ON N2L 2Y5, Canada*

Accepted 2020 March 4. Received 2020 February 27; in original form 2019 September 17

ABSTRACT

An ambitious goal in cosmology is to forward model the observed distribution of galaxies in the nearby Universe today from the initial conditions of large-scale structures. For practical reasons, the spatial resolution at which this can be done is necessarily limited. Consequently, one needs a mapping between the density of dark matter averaged over ∼Mpc scales and the distribution of dark matter haloes (used as a proxy for galaxies) in the same region. Here, we demonstrate a method for determining the halo mass distribution function by learning the tracer bias between density fields and halo catalogues using a *neural bias model*. The method is based on the Bayesian analysis of simple, physically motivated, neural network-like architectures, which we denote as neural physical engines, and neural density estimation. As a result, we are able to sample the initial phases of the dark matter density field while inferring the parameters describing the halo mass distribution function, providing a fully Bayesian interpretation of both the initial dark matter density distribution and the neural bias model. We successfully run an upgraded BORG (Bayesian Origin Reconstruction from Galaxies) inference using our new likelihood and neural bias model with halo catalogues derived from full *N*-body simulations. In preliminary results, we notice there could potentially be orders of magnitude improvement in modelling compared to classical biasing techniques.

Key words: methods: data analysis – methods: statistical – galaxies: haloes – dark matter – large-scale structure of Universe.

1 INTRODUCTION

Observations of the large-scale structure of the Universe provide a window that allows us to constrain physical models of the Universe. Although cosmological models can predict the statistical nature of the structures that we see, it is difficult to extract the wealth of information that we observe in real objects. One is able to make use of these real structures by constraining the initial amplitudes and phases of the dark matter distribution conditional on the observed data. This ambitious task is made possible using algorithms such as the Bayesian Origin Reconstruction from Galaxies (BORG) algorithm (Jasche & Wandelt 2013; Jasche, Leclercq & Wandelt 2015; Lavaux & Jasche 2016) and/or other reconstruction schemes (e.g. Kitaura & Enßlin 2008; Kitaura 2013; Wang et al. 2016; Feng, Seljak & Zaldarriaga 2018). With the BORG algorithm, the initial distribution of dark matter is evolved forward to the dark matter density today, at which point a bias model is used to compare to the observed distribution of galaxies via a choice of metric (the likelihood). In essence, the bias model contains a phenomenological description of the complex astrophysics that dictates how galaxies trace the dark matter distribution (Peebles 1980; Kaiser 1984). The bias model therefore is a parametrized surrogate for the extremely non-linear, scale-dependent, and environment-dependent astrophysics of galaxy formation and evolution. By assuming that galaxy formation is a local function of the dark matter density, one can still gain information about cosmology from the large-scale distribution of galaxies that is related to the large-scale mass density distribution. However, a poor choice of bias model can massively impact the inference of the initial conditions due to the

★ E-mail: charnock@iap.fr

mapping between the dark matter distribution and the observables being incorrect (Elsner et al. 2020).

In this work, we present a novel suite of methods for learning a bias model based on physically motivated, neural network-like algorithms (which we dub 'neural physical engines', NPEs) and neural density estimators: a neural bias model. The 'neural' adjective is used here because we are inspired by the computational machinery introduced by the machine learning community, and in particular neural networks, to solve for the parameters of the bias model. Furthermore, the parameters of the network will be inferred as part of the feed-forward inference of the initial density phases within the BORG framework. As such, the parameters for such a neural bias model become part of the fully Bayesian interpretation of the constraints. By using the physically motivated architectures of an NPE, we can both massively decrease the number of parameters in the model and drastically increase the interpretability of where the information about the bias model arises in the data.

The combination of the NPE and the Bayesian sampling of the parameters of the model provides us with a method for using neural networks *without any training data* since the distribution of weights (parameters of the bias model) is conditional on the true (observed) data. Our approach is therefore the realization of an ultimate version of *zero-shot learning*; the neural bias model is learned directly from the data to be analysed, without reference to any training data.

For convenience, we will limit ourselves to modelling the relation between the large-scale dark matter distribution and its embedded small-scale haloes. Haloes are gravitationally bound objects that host the galaxies that we see. It is useful to be able to describe the distribution of haloes of a particular mass within a given density environment. By learning to parametrize this distribution as a function of the local density field, it becomes possible to sample realizations of the observed haloes that can then be constrained using halo catalogues. Therefore, this method provides a way to emulate some of the features of the halo occupation distribution model using only differentiable operations.

The paper will follow as such. In Section 2, we will describe the properties of the halo mass distribution function that relates the dark matter distribution today to (observed) haloes. We will then show in Section 3 that we can build a neural network capable of learning this function from data, and how this network can be made efficient and interpretable using physical principles. We will also explain how the parameters of this NPE can be inferred to provide us with a fully Bayesian interpretation of the neural bias model. In Section 4, we will elucidate the BORG framework and how the neural bias model can be included. Finally, in Section 5 we present the data simulation, model, and results before concluding in Section 6.

2 HALO MASS DISTRIBUTION FUNCTION

The halo mass distribution function conditioned on density, $n(M|\delta)$, is a measure of the number of counts of haloes as a function of mass, given the environment (Press & Schechter 1974; Bardeen et al. 1986; Mo & White 1996; Sheth & Tormen 2004a). In particular, there are a number of different effects that influence the form of this function, including the formalism of Press & Schechter (1974) in which the formation of gravitationally collapsed objects from the initial density fluctuations is described by a power law for small masses with an evolving exponential mass cut-off. Further, the local density field is known to affect the shape of the function (Kaiser 1984; Bardeen et al. 1986; Sheth & Tormen 2004b). Stochastic and higher order effects are also known to influence the mass distribution of haloes, in particular the departure from a Poissonian distribution of the bias (Kitaura, Yepes & Prada 2014; Saito et al. 2014).

We use this principle to motivate a physical mapping between the dark matter density distribution and the dark matter haloes. To include all relevant information, and allow freedom to correctly fit the true halo mass distribution, we will consider a neural bias model with the ability to learn about the important structures of the environment that affect the number of haloes with different masses, directly from the data, without additional training data. In this first outlook into learning such tracer biases, we will consider a Poissonian sampling of the halo mass distribution function, where we allow some freedom from the Poissonity by non-linearly combining local patches of the density field. It should be noted that here we are also only considering real space simulations and not observation in redshift space, which will be necessary for use with real surveys. This can be generalized straightforwardly, and this will be explored in future work.

For very narrow mass bins $[\mathfrak{m}, \mathfrak{m} + \Delta\mathfrak{m}]$, we may express the Poisson intensity from the expectancy $\lambda_{i,m}$ of observing $N_{i,m}$ haloes within a particular bin as

$$\begin{aligned}\lambda_{i,m} &\equiv \langle N_{i,m} \rangle \\ &= V \int_{\mathfrak{m}}^{\mathfrak{m}+\Delta\mathfrak{m}} \mathrm{d}M\, n(M|\{\delta_j | j \in \text{local patch around } i\}),\end{aligned} \tag{1}$$

with V the volume of a voxel of the grid where i labels a specific voxel and j labels the voxels in some local patch around i. The density field $\boldsymbol{\delta}(\boldsymbol{\theta}) \equiv \{\delta_i | \, i \in \text{voxels}\}$ is a function of some set of parameters $\boldsymbol{\theta}$ that describe cosmology, the tracer bias between the halo and dark matter, and the initial conditions of the dark matter density field. In the above, we note that the halo distribution may be voxel dependent. For a Poisson likelihood, for which all mass bins and all voxels are independent, we may write the logarithm of the likelihood $\mathcal{L}(\boldsymbol{\theta}|\mathbf{d})$ as

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{d}) = \sum_{\substack{m \in \text{mass bins} \\ i \in \text{voxels}}} \left(-\lambda_{i,m}(\boldsymbol{\theta}) + N_{i,m}^{\text{observed}} \log \lambda_{i,m}(\boldsymbol{\theta})\right), \tag{2}$$

with the observed data $\mathbf{d} = \{N_{i,m}^{\text{observed}} | \, i, m \in \text{catalogue}\}$ as the number of haloes observed in mass bin m and at voxel i in the grid. The above equation is just the sum of the logarithm of the Poisson probability distribution, which arises naturally since Poisson distributions are additive. For very narrow mass bins, such that $\Delta\mathfrak{m} \to 0$, the number of observed haloes in each mass bin can only be 0 or 1, $N_{i,m}^{\text{observed}} = 0 \vee 1$. We may thus reorganize the summation as

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{d}) = \sum_{h \in \text{catalogue}} \log \lambda_{i_h, m_h}(\boldsymbol{\theta}) \sum_{\substack{m \in \text{mass bins} \\ i \in \text{voxels}}} \lambda_{i,m}(\boldsymbol{\theta}), \tag{3}$$

where λ_{i_h, m_h} corresponds to the expected Poisson intensity for the h^{th} halo in the catalogue whose mass is in the mass bin m_h and is located at the voxel i_h. By substituting equation (1) into equation (3), we can see

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\mathbf{d}) = &\sum_{h \in \text{catalogue}} \log\left(n(m_h|\{\delta_j | \, j \in \text{local patch around } i_h\})\right) \\ &- V \sum_{i \in \text{voxels}} \int_{\mathfrak{m}_\tau}^{\infty} dM\, n(M|\{\delta_j | \, j \in \text{local patch around } i\}),\end{aligned} \tag{4}$$

where we have discarded the constant induced by the logarithm of the infinitesimal binning $\Delta\mathfrak{m}$. The first term involves calculating

the halo mass distribution function for every halo of the catalogue given the density at each halo's position on a grid. The second term evaluates the halo mass distribution function at every voxel in the gridded density field and integrates over the mass from some mass threshold $\mathfrak{m}_\tau$.

3 NEURAL BIAS MODEL

We wish to build an automatic method for modelling the halo mass distribution function based on the physical principles that we know are relevant, but which is parametrized as simply as possible to provide us with the ability to effectively sample the parameters of this model. The added bonus of such a simple model is the increased interpretability of the function.

Our model for $n(m_h|\{\delta_j|\, j \in \text{local patch around } i_h\})$ consists of two parts. First, we need to determine the influence of the central and adjacent voxels $\{\delta_j|\, j \in \text{local patch around } i_h\}$. How to determine this mapping using convolutional kernels is described in Section 3.1. Secondly, since we want to build the output of the neural bias model to be the halo mass distribution function, we need to ensure that the outputs are evaluations of this function for haloes of a certain mass given a density environment. To do so, we can consider neural density estimators, and in this case a simple mixture density network (MDN) is sufficient. MDNs are explained in further detail in Section 3.2. Finally, it is essential that we can infer the parameters of our neural bias model to provide us with a fully Bayesian interpretation of the initial density field and the tunable parameters of the network. The method by which we are able to efficiently sample these parameters is described in Section 3.3.

3.1 Neural physical engines

We will here use terminology familiar to those in the machine learning and computer science communities (for full details, see e.g. Goodfellow, Bengio & Courville 2016). Much of the work considered derives from the understanding of neural networks. Neural networks are described as a set of algorithms with *trainable* parameters known as weights and biases that map an input to an output. The algorithms usually consist of simple operations such as matrix multiplications or convolutions between input and weight vectors. The output vectors of these operations are then normally acted upon by a scalar *activation* function, which allows one to add non-linearity to the mapping. Finally, many *layers* of these operations can be stacked input to output, building a great deal of abstraction between the original input and the final output. This provides us with a highly complex, non-linear, and arbitrary function that can be fitted using training data.

It is currently fashionable to build extremely large, deep neural networks capable of huge abstraction from the input to the output. However, in physics we have models that can drive the design and conception of the architecture that we wish to use. The symmetries of the physical model that describes the data can be used to massively reduce both the amount of data and effort needed to train such an NPE. Since an NPE is greatly restricted in the freedom along the directions that we know are constrained by the data, there is far less chance of overfitting. Along with this, we obtain a much greater interpretability of what the neural network is doing. In this work, we will denote the NPE $\mathbb{N}(\boldsymbol{\theta}^{\rm NPE}) : \boldsymbol{\delta} \rightarrow \boldsymbol{\psi}$, i.e. the physically motivated mapping with tunable parameters $\boldsymbol{\theta}^{\rm NPE}$ that takes a patch of the evolved density field $\boldsymbol{\delta}$ to a transformed informative vector, $\boldsymbol{\psi}$, for the halo mass distribution function. Note in this work that we consider the size of $\boldsymbol{\psi}$ to be 1 as it represents the transformed density field in the central voxel.

As described at the start of Section 2, we already know that the local density fluctuations are important for describing the halo mass distribution, as is the large-scale overdensity. We also know that the action of the tracing of the dark matter distribution by haloes is both translationally and rotationally invariant, although local distortions can give rise to more optimal environments for the development of haloes. Furthermore, we know that the bias model is non-linear, all of which we can build into our neural bias model.

Thus, in summary, we seek a mapping that will convolve the density in nearby voxels and return a value, $\boldsymbol{\psi}$, which characterizes the environment. This summary will then be used as the informative input in determining the halo distribution function.

We start constructing our model using convolutional kernels that respect symmetries. Naturally, the idea of convolutional kernels respects the translational symmetry of the problem. As is usual in the machine learning literature, convolutional kernels are matrices where every element is a trainable parameter known as a weight. Since we also want to build in rotational invariance and local deformations of rotational invariance, we expand the kernels in terms of multipoles. This provides us with a hierarchy of convolutional kernels with each multipole describing further deformation from exact rotational symmetry.[1] In three dimensions (as considered here), the basis of rotational symmetries of the convolutional kernels is spherical harmonics. In the $\ell = 0$ case, such a kernel is constructed by associating the same weight in the convolutional kernel to each matrix position that is equidistant from the centre. Then, for $\ell = 1$ we have $2\ell + 1$ kernels where each weight is associated by distance and by the values of $Y^\ell_m(\theta, \phi)$ at angles θ and ϕ from the centre of the matrix. This expansion can be continued to learn further information about deformations away from exactly rotationally invariant environments. As an example of the drastic savings in the number of weights that one can obtain by expanding the kernels in terms of multipoles, consider a 3^3 kernel. This kernel would traditionally have 27 independent trainable parameters. The $\ell = 0$ kernel has only four weights, one at the centre and one shared on every edge, every face, and every corner. The $\ell = 1$ kernel has $(2\ell + 1) = 3$ output *feature* maps, each kernel containing three independent weights. In this case, only 9 parameters are needed in the place of 27. It should be noted that if the $\ell = 2$ kernel is also included, all 27 of the available parameters of a 3^3 kernel are exhausted. As these kernels are fitted, using whichever optimization procedure is most suitable, information is extracted from the data via these orthogonal pathways. By analysing the scale of the weights in each of these kernels, it becomes obvious to see where the information in the data is contained. That can be, for example, in the radially symmetric patches of the data, or in the shear component of the data. To summarize, the output feature maps, $\boldsymbol{C}^{\ell,m}$, of the symmetry respecting convolutional kernels, $\boldsymbol{K}^{\ell,m}$, take the form

$$C^{\ell,m}_j = \sum_{i=-\kappa/2}^{\kappa/2} K^{\ell,m}_i \delta_{j-i} \, . \tag{5}$$

Here, for simplicity, we have written a single sum representing the N summations for a kernel $\boldsymbol{K}^{\ell,m} \in \mathbb{R}^N$, where κ represents the size of each of the N dimensions of $K^{\ell,m}$. The values of $K^{\ell,m}_i$ depend on

[1] The code for generating multipole kernels in TENSORFLOW (with available KERAS module) is available at https://github.com/tomcharnock/multipole_kernels.

the multipole (ℓ, m). Each ℓ and m kernel has a different prescription of the number of shared weight elements and the position of those shared elements within the kernel.

The non-linearity of the bias model is included via the application of an activation function to the output of the convolutional kernels. Formally, we thus have

$$\boldsymbol{\phi}^{\ell,m} = A(\boldsymbol{C}^{\ell,m}) \,, \tag{6}$$

with $A(x)$ a scalar activation function and $\boldsymbol{\phi}^{\ell,m}$ as the ℓ and m components of the activated feature maps. $A(x)$ is normally chosen to be a non-linear function that can be optimally specified depending on the function that the network is approximating. The more concurrent layers in the NPE, the more non-linear the response from the input patch can be. Moreover, deep, activated stacks independent along the orthogonal pathways can provide extremely complex functions that remain, for example, rotationally invariant for $\ell = 0$, provided the activation function remains scalar. This is a particular example, relevant to the case under consideration here, because we restrict to $\ell = 0$. For broader applications, coupling the multipoles would drastically broaden the scope of physical effects that can be modelled.

To ensure that the relevant local information is taken into account, the size of the kernel (or the total combined receptive patch of many layers of convolutions) must be large enough to pull all relevant information from the data. This size should generally be known, thanks to the physical model. However, regardless of this, one can construct large spatial multipole kernels if necessary, since the number of parameters is already massively reduced. The parameter values from regions of the kernel that are not important will tune to zero and be interpretable as the distance over which information is relevant for the problem. Furthermore, the scales of each of the kernels for different multipoles will indicate on which scales the degree of deformation is most important.

Finally, we choose to use the late-time overdensity field as our input such that the neural bias model is provided with the non-local information about the density field via the mean of the distribution. We could consider, instead, biasing the initial conditions and transporting that along with the dynamical model forming a Lagrangian neural bias model, which folds in the dark matter history. We will explore this option in future works since it is possible that a sufficiently complex neural bias model will be able to obtain these aspects of the history from the final time-step.

Using these techniques, we build a well-reasoned model that performs the real space mapping to the quantity of interest while remaining free enough to fit the complex function, all the while being interpretable.

3.2 Mixture density networks

To approximate the halo mass distribution function, we make use of a very slight variation of an MDN (Bishop 1994). MDNs use neural networks fitted to predict the normalized weights and parameters of several distributions, such as the mean and standard deviation of a Gaussian distribution, the event rate of a Poisson distribution, or the shape parameters of a β-distribution. By combining a mixture of distributions, new distributions of many different forms can be built:

$$\mathcal{P}(\mathbf{x}|\mathbf{d}, \boldsymbol{\theta}) = \sum_{\iota=1}^{N} \alpha_\iota(\mathbf{d}, \boldsymbol{\theta}) \mathcal{F}_\iota(\mathbf{x}|\mathbf{d}, \boldsymbol{\theta}), \tag{7}$$

where $\mathcal{P}(\mathbf{x}|\mathbf{d}, \boldsymbol{\theta})$ is the mixture of probability distributions, $\mathcal{F}_\iota(\mathbf{x}|\mathbf{d}, \boldsymbol{\theta})$ is the collection of N parametrizable distributions, $\alpha_\iota(\mathbf{d}, \boldsymbol{\theta})$ is a set of amplitudes for each distribution, and $\mathbf{x}$, $\mathbf{d}$, and $\boldsymbol{\theta}$ are the parameters of the model to be inferred, the input data, and the trainable parameters of the neural network, respectively.

To ensure that the output of the MDN can be interpreted as a probability distribution, the set of $\alpha_\iota(\mathbf{d}, \boldsymbol{\theta})$ is *activated* using a softmax function, defined as

$$\text{softmax}(\boldsymbol{\alpha}) = \frac{\alpha_\iota}{\sum_{\iota=1}^{N} \alpha_\iota} \,. \tag{8}$$

This ensures that the amplitudes are normalized. In principle, the number of distributions in the MDN is self-regulating since the amplitudes of any irrelevant distributions vanish. This self-regularization can lead to problems with Bayesian sampling of the network since there is strong multimodality. This multimodality is in practice solved by restricting the parameter space as we show in equation (11).

We are interested in approximating the number density of haloes with a certain mass given a density environment, $n(M|\delta_i) \equiv \bar{N}\mathcal{P}(M|\boldsymbol{\psi}_i, \boldsymbol{\theta}^{\text{MDN}})$, where $\boldsymbol{\psi}_i$ is the (single-valued) vector output of the NPE whose input is $\{\delta_j | \; j \in \text{local patch around } i\}$) as discussed in Section 3.1 and $\bar{N}$ is the mean number density. Since the number density is not a probability distribution, we do not constrain the set of amplitude parameters $\alpha(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\alpha}}_\iota)$ using the softmax function, and instead just ensure their positivity.

Since the halo mass distribution function is relatively smooth (as seen via the diamonds in Fig. 2), we decide to model it using a mixture of Gaussians

$$\begin{aligned} n(M|\delta) &= \sum_{\iota}^{N} \alpha\left(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\alpha}}_\iota\right) \mathcal{N}\left(\mu\left(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\mu}}_\iota\right), \sigma(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\sigma}}_\iota)|M\right) \\ &= \sum_{\iota}^{N} \frac{\alpha\left(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\alpha}}_\iota\right)}{\sqrt{2\pi\left(\sigma\left(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\sigma}}_\iota\right)\right)^2}} \exp\left[-\frac{\left(\log(M) - \mu\left(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\mu}}_\iota\right)\right)^2}{2\left(\sigma\left(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\sigma}}_\iota\right)\right)^2}\right], \end{aligned} \tag{9}$$

where we consider the logarithm of the mass for numerical stability and the insight provided by the Press–Schechter formalism on the mass function. For convenience, we will denote the parameters of the ι^{th} distribution as $\alpha(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\alpha}}_\iota) \equiv \alpha_\iota$, $\mu(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\mu}}_\iota) \equiv \mu_\iota$, and $\sigma(\boldsymbol{\psi}, \boldsymbol{\theta}^{\boldsymbol{\sigma}}_\iota) \equiv \sigma_\iota$. The parameters of our mixture of Gaussians depend on $\boldsymbol{\psi}$ and the trainable parameters $\boldsymbol{\theta}^{\text{MDN}} = \left\{\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \boldsymbol{\theta}^{\boldsymbol{\mu}}, \boldsymbol{\theta}^{\boldsymbol{\sigma}}\right\}$, where each $\boldsymbol{\theta}^{\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\sigma}} = \left\{w^{\alpha,\mu,\sigma}_\iota, b^{\alpha,\mu,\sigma}_\iota \middle| \iota \in [0, N)\right\}$. The parameters of the Gaussians are calculated as

$$\alpha_\iota = \text{softplus}\left(w^{\alpha}_\iota \boldsymbol{\psi} + b^{\alpha}_\iota\right), \tag{10}$$

$$\mu_\iota = \begin{cases} w^{\mu}_\iota \boldsymbol{\psi} + b^{\mu}_\iota, & \text{if} \quad \iota = 0, \\ \max\left[0,\; w^{\mu}_\iota \boldsymbol{\psi} + b^{\mu}_\iota\right] + \mu_{\iota-1}, & \text{if} \quad \iota > 0, \end{cases} \tag{11}$$

$$\sigma_\iota = \text{softplus}\left(w^{\sigma}_\iota \boldsymbol{\psi} + b^{\sigma}_\iota\right), \tag{12}$$

where the function softplus is defined as $\text{softplus}(x) \equiv \log(1 + \exp x)$. The softplus function is used to ensure the positivity of both the amplitude and standard deviations of the MDN, while being differentiable at any point. The means of the MDN are ordered by amplitude from smallest to largest. This is required to remove the degeneracy introduced when removing the softmax constraint on the amplitudes.

Since we have some prior on the function, we decide to use a shifted coordinate system for the weights, making use of our knowledge of the halo mass distribution function (seen in Fig. 2). We therefore include an initial amplitude α^{init}, width σ^{init}, and mass threshold mass threshold $\mathfrak{m}_\tau$:

$$\boldsymbol{b}^{\alpha} \to \boldsymbol{b}^{\alpha} + \alpha^{\text{init}}, \tag{13}$$

$$b_0^{\mu} \rightarrow b_0^{\mu} + \log(\mathfrak{m}_{\tau}), \tag{14}$$

$$\boldsymbol{b}^{\sigma} \rightarrow \boldsymbol{b}^{\alpha} + \sigma^{\rm init}, \tag{15}$$

which allows all the parameters of the MDN, $\boldsymbol{\theta}^{\rm MDN}$, to be approximately centred on zero. This will be useful when sampling the parameters of the neural bias model since the prior is simple to implement (discussed further in Section 3.3).

Using equation (9) as the halo mass distribution function, we can rewrite the log-likelihood (in equation 4) as

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{d}) = \sum_{h\in\text{catalogue}} \log\left[\sum_{\iota}^{N} \frac{\alpha_{\iota,i_h}}{\sqrt{2\pi\sigma^2_{\iota,i_h}}} \exp\left[-\frac{\left(\log(\mathfrak{m}_h) - \mu_{\iota,i_h}\right)^2}{2\sigma^2_{\iota,i_h}}\right]\right] - V\sum_{i\in\text{voxels}}\sum_{\iota}^{N} \frac{\alpha_{\iota,i}}{2}\exp\left[\frac{\sigma^2_{\iota,i}}{2}\right]\text{erfc}\left[\frac{\log(\mathfrak{m}_{\tau}) - \mu_{\iota,i} - \sigma^2_{\iota,i}}{\sqrt{2\sigma^2_{\iota,i}}}\right], \tag{16}$$

where $\alpha_{\iota,i}$, $\mu_{\iota,i}$, and $\sigma_{\iota,i}$ represent the amplitude, mean, and standard deviations of the ιth element of the mixture of Gaussians given an input ψ_i at voxel i. The data $\mathbf{d}$ are described by the haloes in catalogue at positions i_h in mass bin $\mathfrak{m}_h$. This is the effective likelihood surface that we wish to explore to be able to infer the distribution of parameters in the neural bias model at the same time as the initial phases of the dark matter distribution.

Removing the constraint on the amplitudes of the distributions means that the number of distributions can no longer self-regulate. Therefore, we must choose a fixed number of distributions, determined by the expected shape of the function. It is possible to make use of Bayesian optimization and model comparison to perform the regularization, and this will be studied further in future works.

3.3 HMCLET

To obtain a Bayesian interpretation of the neural bias model (and the initial phases of the density field), we need to infer the distribution of the parameters of the model. The landscape of the likelihood surface (in equation 16) is extremely flat in the directions of the parameters of the neural bias model and is highly correlated, often in unpredictable ways. As such, we need to use specific techniques to be able to effectively traverse the likelihood surface. We use a Hamiltonian Monte Carlo (HMC, also known as hybrid Monte Carlo; Duane et al. 1987) sub-block in the BORG framework (which we call HMCLET with -LET denoting the small dimensionality of the HMC) to draw samples from the target conditional posterior distribution of the neural bias model parameters given the matter density field. HMC is a Markov chain Monte Carlo method where the proposed states are dictated by a *momentum*, $\boldsymbol{p}$, i.e. the first-order gradient information of the target distribution, and the acceptance rate is kept high via conservation of energy momentum. In particular, the Hamiltonian is defined as the negative log-probability of the distribution, $\mathcal{P}(\boldsymbol{\theta}, \boldsymbol{p})$, of model parameters, $\boldsymbol{\theta}$, and momenta, $\boldsymbol{p}$:

$$\begin{aligned}\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p}) &= \mathcal{P}(\boldsymbol{\theta}, \boldsymbol{p}) \\ &= \mathcal{K}(\boldsymbol{p}) + \mathcal{V}(\boldsymbol{\theta}) \\ &= \frac{1}{2}\boldsymbol{p}^T\mathbf{M}^{-1}\boldsymbol{p} - \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{d}) - \log\left[\pi(\boldsymbol{\theta})\right] + \text{constant}.\end{aligned} \tag{17}$$

$\mathcal{K}(\boldsymbol{p})$ is a *kinetic energy* with a mass matrix, $\mathbf{M}$, describing the correlation between parameters. $\mathcal{V}(\boldsymbol{\theta})$ is a potential energy formed from the negative logarithm of the likelihood (in equation 16) and the prior, $\pi(\boldsymbol{\theta})$, on the parameters. The state $\boldsymbol{z} = \{\boldsymbol{\theta}, \boldsymbol{p}\}$ is found by solving the ordinary differential equation (ODE) derived from Hamiltonian dynamics:

$$\dot{\boldsymbol{\theta}} = \mathbf{M}^{-1}\boldsymbol{p}, \tag{18}$$

$$\dot{\boldsymbol{p}} = -\nabla\mathcal{V}(\boldsymbol{\theta}), \tag{19}$$

where the dots are derivatives in time (as introduced for the momenta). Proposals of the ith parameters, $\boldsymbol{\theta}_i$, are generated by drawing a momentum from a proposal distribution, $\boldsymbol{p}_i \leftarrow \mathcal{N}(\mathbf{0}, \boldsymbol{M})$, and evolving these using equations (18) and (19) to obtain $\boldsymbol{z}^* = \{\boldsymbol{\theta}^*, \boldsymbol{p}^*\}$. The acceptance condition for the Metropolis–Hasting procedure is obtained by computing the difference in energies between the ith state and the proposed state:

$$\text{acceptance probability} = \min\left[\exp(\Delta\mathcal{H}), 1\right], \tag{20}$$

where $\Delta\mathcal{H} = \mathcal{H}(\boldsymbol{\theta}_i, \boldsymbol{p}_i) - \mathcal{H}(\boldsymbol{\theta}^*, \boldsymbol{p}^*)$ arises from the discretization of solving Hamilton's equations. If the equations were solved exactly (the Hamiltonian is conserved), then every single proposal is accepted. It is typical to use ϵ-discretization (leapfrog method) to solve the ODE, where ϵ describes the step size of the integrator (Verlet 1967). Smaller step sizes result in higher acceptance rate at the expense of longer computational times of the integrator, while larger step sizes result in shorter integration times, but lower acceptance.

Because the likelihood described in equation (16) is flat with sharp edges, the momentum of the model parameters can cause the integrator to step away from the domain of existence of the gradients of the likelihood. To prevent this, we choose a Gaussian prior on each of the parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}^{\rm NPE}, \boldsymbol{\theta}^{\rm MDN}\}$, centred on zero with identical widths:

$$\pi(\boldsymbol{\theta}) \propto \exp\left[-\frac{|\boldsymbol{\theta}|^2}{2\sigma^2_{\rm prior}}\right]. \tag{21}$$

We could choose more general priors independently for each parameter, but this is difficult since we do not have prior information on the scale of the weights of the network. We are able to set the mean of the Gaussian prior to zero by ensuring that all trainable parameters of the neural bias model are close to zero via the use of the initial amplitude, width, and mass threshold mentioned in Section 3.2. As such, we just have to choose the Gaussian prior to be wide enough to allow plenty of freedom for parameter value exploration while preventing the parameters of the neural bias model from becoming extremely large. This also implies that the total probability distribution is now ensured to be proper.

We also use an adaptation to the usual HMC paradigm, the quasi-Newtonian HMC (QNHMC; Fu, Luo & Zhang 2016). This is because there is very little a priori knowledge about the correlations between the parameters of the neural bias model, especially those in the NPE, and therefore the ODE is exceptionally stiff. With the QNHMC, we make use of the second-order geometric information of the target distribution as well as the gradient. This additional information can be efficiently approximated using quasi-Newtonian methods. The QNHMC modifies equations (18) and (19) to

$$\dot{\boldsymbol{\theta}} = \mathbf{B}\mathbf{M}^{-1}\boldsymbol{p}, \tag{22}$$

$$\dot{\boldsymbol{p}} = -\mathbf{B}\nabla\mathcal{V}(\boldsymbol{\theta}), \tag{23}$$

where $\boldsymbol{B}$ is an approximation to the inverse Hessian derived from the L-BFGS technique (Liu & Nocedal 1989) found using quasi-Newton methods:

$$\mathbf{B}^* = \left(\mathbb{I} - \frac{\boldsymbol{s}_i \boldsymbol{y}_i^T}{\boldsymbol{y}_i^T \boldsymbol{s}_i}\right) \mathbf{B}_i \left(\mathbb{I} - \frac{\boldsymbol{y}_i \boldsymbol{s}_i^T}{\boldsymbol{y}_i^T \boldsymbol{s}_i}\right) + \frac{\boldsymbol{s}_i \boldsymbol{s}_i^T}{\boldsymbol{s}_i^T \boldsymbol{y}_i}, \tag{24}$$

where $\boldsymbol{s}_i = \boldsymbol{\theta}^* - \boldsymbol{\theta}_i$, $\boldsymbol{y}_i = \nabla\mathcal{V}(\boldsymbol{\theta}^*) - \nabla\mathcal{V}(\boldsymbol{\theta}_i)$, and $\mathbb{I}$ is the identity matrix.[2] The inverse Hessian effectively rescales the momenta and parameters such that each dimension has a similar scale and thus the movement around the likelihood surface is more efficient and the produced proposals are less correlated. The mass matrix is still present to set the dynamical time-scales of the ODE problem along each direction. The rationale behind the choice of the mass matrix is indicated in Appendix A. Note that the approximate inverse Hessian varies with proposal, but is kept constant while solving the ODE. Obtaining $\mathbf{B}^*$ is extremely efficient because both $\boldsymbol{s}_i$ and $\boldsymbol{y}_i$ are calculated when solving equations (18) and (19) using leapfrog methods. We start with an initial inverse Hessian $\mathbf{B}_0 = \mathbb{I}$ and allow it to adapt to the geometry of the space. Although this requires an estimate of the mass matrix initially, the rescaling of the momenta via $\mathbf{B}$ allows us to be fairly ambiguous about its value. In essence, this all occurs during the burn-in phase of the HMC sampler.

4 BORG FRAMEWORK

Our ability to sample the parameters of the neural bias model builds upon the previously developed BORG algorithm. BORG aims to the analyse the 3D cosmic matter distribution at linear and non-linear scales of structure formation from galaxy surveys (see e.g. Jasche & Wandelt 2013; Jasche et al. 2015; Lavaux & Jasche 2016). Explicitly, the BORG algorithm fits 3D models of gravitational structure formation to data. Via the introduction of a physical model of gravitational structure growth, the inference of the non-linear dark matter distribution today can be posed as a statistical initial condition problem. To do so, the BORG algorithm seeks to infer the cosmic initial conditions from which present 3D structures in the distribution of galaxies have formed via non-linear gravitational mass aggregation.

The BORG algorithm explores the posterior distribution of large-scale structures, consisting of a Gaussian prior for the initial density field at an initial scale factor of $a = 10^{-3}$ and a choice of bias model and likelihood metric at scale factor $a = 1$. The evolution of the initial density fields can be related to the present galaxy (or halo) distribution via a first- or second-order Lagrangian perturbation theory (LPT or 2LPT) or a full particle mesh model of gravitational structure formation (for details, see Jasche & Wandelt 2013; Jasche & Lavaux 2019). By exploiting non-linear structure growth models, the BORG algorithm naturally accounts for the filamentary structure of the cosmic web typically associated with higher order statistics induced by non-linear gravitational processes. Furthermore, the posterior distribution accounts for systematic and stochastic uncertainties, such as survey geometries, selection effects, unknown noise, and galaxy biases as well as foreground contamination (see e.g. Jasche & Wandelt 2013; Jasche et al. 2015; Lavaux & Jasche 2016; Jasche & Lavaux 2017).

In this work, we use the BORG algorithm to sample the initial conditions of the dark matter density field using LPT to evolve the field to the dark matter conditions today and then rely on the neural bias model to sample the parameters of the NPE to provide a field that is most informative about the halo mass distribution function inferred from a halo catalogue. A detailed schematic of the interconnection between parts of the model is shown in Fig. 1. The stochastic uncertainties are assumed to be Poissonian using the likelihood in equation (16), and further study into direct learning of the deviations from Poissonity will be conducted in an upcoming work.

[2]For large dimensions, more computationally and memory efficient methods can be used (Nocedal & Wright 2006).

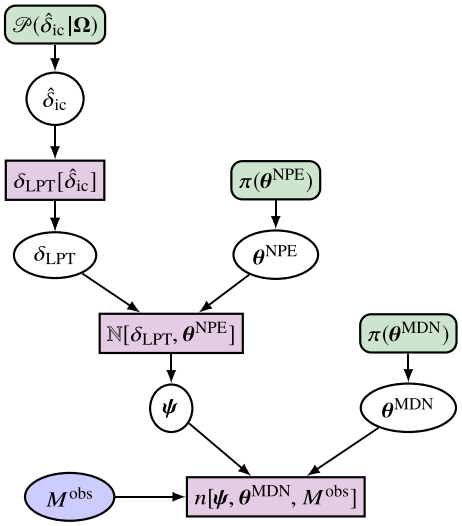

Figure 1. Schematic of the BORG algorithm with the neural bias model. Initial conditions for the density field in Fourier space, $\hat{\delta}_{\rm ic}$, are drawn from a prior given a cosmology $\boldsymbol{\Omega}$, $\mathcal{P}(\hat{\delta}_{\rm ic}|\boldsymbol{\Omega})$. These are then evolved forward using a deterministic prescription, in this example using LPT. The evolved field is then transformed further using the NPE $\mathbb{N}$ that requires parameters $\boldsymbol{\theta}^{\rm NPE}$ that are drawn from a prior $\pi(\boldsymbol{\theta}^{\rm NPE})$. This provides a field $\boldsymbol{\psi}$ from which the halo mass distribution function can be described using the MDN with parameters $\boldsymbol{\theta}^{\rm MDN}$ drawn from a prior $\pi(\boldsymbol{\theta}^{\rm MDN})$. This halo mass distribution function is then compared to the masses of haloes $M^{\rm obs}$ from the observed halo catalogue.

5 RESULTS

To examine the techniques developed in this paper, we will consider a relatively simple mock run using a simulated dark matter distribution and minimal working neural bias model.

5.1 VELMASS simulation

The halo catalogue that we use in this work comes from the VELMASS suite. It is comprised of 10 cosmological simulations, 9 of which are probing slightly different variations of a selection of cosmological parameters while using the same initial phases (for full details of the suite of simulations, see Ramanah, Charnock & Lavaux 2019a). We recall here the salient features that are relevant for this work. The simulation that we use in this work assumes a Planck-like cosmology (Planck Collaboration XIII 2016) with $\Omega_{\rm m} = 0.315$, $\Omega_{\rm b} = 0.049$, $H_0 = 68\,{\rm km\,s^{-1}\,Mpc^{-1}}$, $\sigma_8 = 0.81$, $n_{\rm s} = 0.97$, and $Y_{\rm He} = 0.248$ (named 'central' or Ω simulation). The power spectrum is obtained through the analytic prescription of Eisenstein & Hu (1999), and the initial conditions are generated by MUSIC (Hahn & Abel 2011).

The cosmological simulation covers a volume of $2000\,h^{-1}$ Mpc with 2048^3 particles tracing dark matter, initialized at a redshift $z = 50$ and evolved to present time with GADGET2 (Springel 2005), adopting a softening length for gravity equal to $48\,h^{-1}$ kpc

corresponding to 1/20th of the mean interparticle separation. The ROCKSTAR halo finder algorithm (Behroozi, Wechsler & Wu 2013) was subsequently employed to extract the haloes from the simulation and generate the halo catalogue. The particle mass resolution is $8.10\,h^{-1}\,\mathrm{M}_\odot$.

Using a patch of the central VELMASS simulation of side $250\,h^{-1}$ Mpc gridded on to a 64^3 grid,[3] and a fairly sampled set of haloes from the corresponding halo catalogue, we attempt to constrain both the initial phases of the dark matter density field and the parameters of the neural bias model.

5.2 Neural bias model

In this work, we focus on a minimal model using a single $\ell = 0$ kernel with extent of 3^3 voxels ($\sim 12\,h^{-1}$ Mpc per side). This is sufficient for studying the first-order effect of the beyond-local density environment. We also use a single softplus activation function on the output feature map from the $\ell = 0$ kernel that provides us with the non-linearity necessary to infer the parameters of the MDN. Further studies into the optimal architecture for extracting all the relevant information from the density field about the halo mass distribution function will be considered in future works. In particular, we note that in this work we consider only a real space mock without redshift space distortions. If we were to extend to more realistic observations, we could proceed in two different ways. First, we could use a similar network presented here to provide the bias model and then use codes such as ALTAIR (Ramanah et al. 2019b) as an intermediate step between the neural bias model in real space and the observations. ALTAIR is an extension to BORG that performs a cosmological parameter-dependent coordinate transform from real space into redshift space. Using such a step would allow us to extract information about the cosmological parameters directly using physical models, while the neural bias model provides the agnostic fit of the tracer bias. However, knowing the physical properties of the coordinate transform we could equally add the $\ell = 1$ and higher order multipole kernels to the neural bias model from which we could learn about the real space distortions due to observations in redshift space at the same time as the effect of the biasing. While both methods are valid, we expect that more cosmological information could be gained using ALTAIR due to the exact form of the coordinate transformation. This will be studied when we consider observations in redshift space.

Thanks to the simplicity of the NPE considered in this paper, we only introduce five trainable parameters, $\boldsymbol{\theta}^{\mathrm{NPE}}$. We centre the parameters of the MDN, $\boldsymbol{\theta}^{\mathrm{MDN}}$, on zero by using an initial amplitude $\alpha^{\mathrm{init}} = \log(1 \times 10^{-3})$, mass threshold $\mathfrak{m}_\tau = 2 \times 10^{12}\,h^{-1}\,\mathrm{M}_\odot$, and initial width $\sigma^{\mathrm{init}} = \log(1 \times 10^{3})$. Due to the number of visible features in the halo mass distribution function shown in Fig. 2, we determine that two Gaussians are sufficient to model the halo mass distribution function. As such, the total number of parameters for the MDN is 12, so the neural bias model has 17 parameters to infer. Further study into the number and types of distributions optimal for extracting all of the information from the density field about the halo mass distribution function will be left to future work. We chose a prior width of $\sigma_{\mathrm{prior}} = 10$ to allow for a range of possible parameter values in the neural bias model while preventing numerical stability issues if they become extremely large. A posteriori verification that the prior has no practical impact on the inference is shown in Fig. 3, since the amplitudes of the weights are all well within the 1σ region of the Gaussian prior.

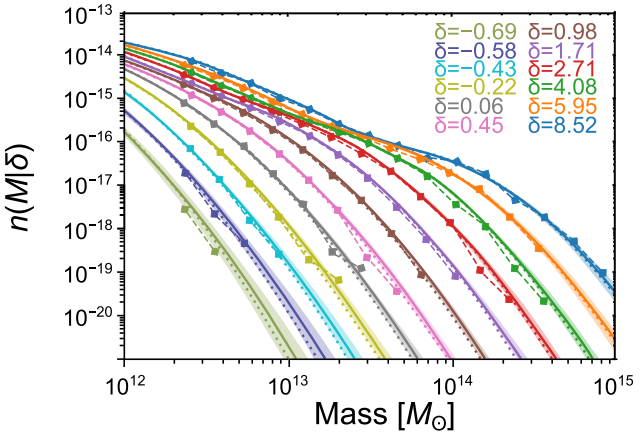

Figure 2. The halo mass distribution function as a function of mass (in solar masses). The diamonds connected by a dashed line indicate the number density of haloes from the central VELMASS halo catalogue of a given mass, where the different colours represent the value of the density environment for those haloes. The lines higher in number density correspond to the more dense regions; i.e. there are more large haloes in denser environments. The solid lines show the mean halo number density from samples (taken from the Markov chain) from the neural bias model, with the shaded bands as the 1σ deviations of these samples. The dotted line indicates the number density of haloes of a given mass using the initialization values of the parameters from the neural bias model. There is a subtlety in the density environments. The diamonds indicate the number density of haloes in a voxel whose density is equal to the numbers shown in the legend, while the neural bias model takes a 3^3 patch whose average density is equal to the numbers shown in the legend. We see that there is a very good agreement between the observed halo number density and that obtained by the neural bias model. We can also see that the sampling captures the distribution of possible number densities about the observed data. Furthermore, the fact that the shape of the function changes as a response to the change in density environment is an indication that the non-linearity of the tracer bias is captured by the neural bias model.

The neural bias model is written using the JULIA (Bezanson et al. 2017) interface to TENSORFLOW (Abadi et al. 2016; Malmaud & White 2018). This is embedded into the HMCLET, which is one sub-block of the BORG algorithm. A skeleton of the JULIA code is available along with this paper.[4]

Since running the HMC is computationally expensive, especially when sampling the initial density field, we pre-train the neural bias model using stochastic gradient descent to find pseudo-maximum-likelihood estimates of the weights. These parameters are then used in the initialization of the neural bias model in the HMCLET to help prevent a long burn-in. This option is available in this case since we have both the true density field and halo catalogue generated from that field. However, the pre-training would not be possible when constraining the network on real data, where the density field is not available. In this case, the burn-in of the HMCLET would be much longer and more computationally demanding.

[3] High-resolution results over the whole simulation will be reserved for future papers studying optimal forms of the neural bias model and studies into direct likelihood estimation. 64^3 is more than sufficient for indicating the methodology of the techniques presented in this paper.

[4] https://github.com/tomcharnock/neural_bias_model

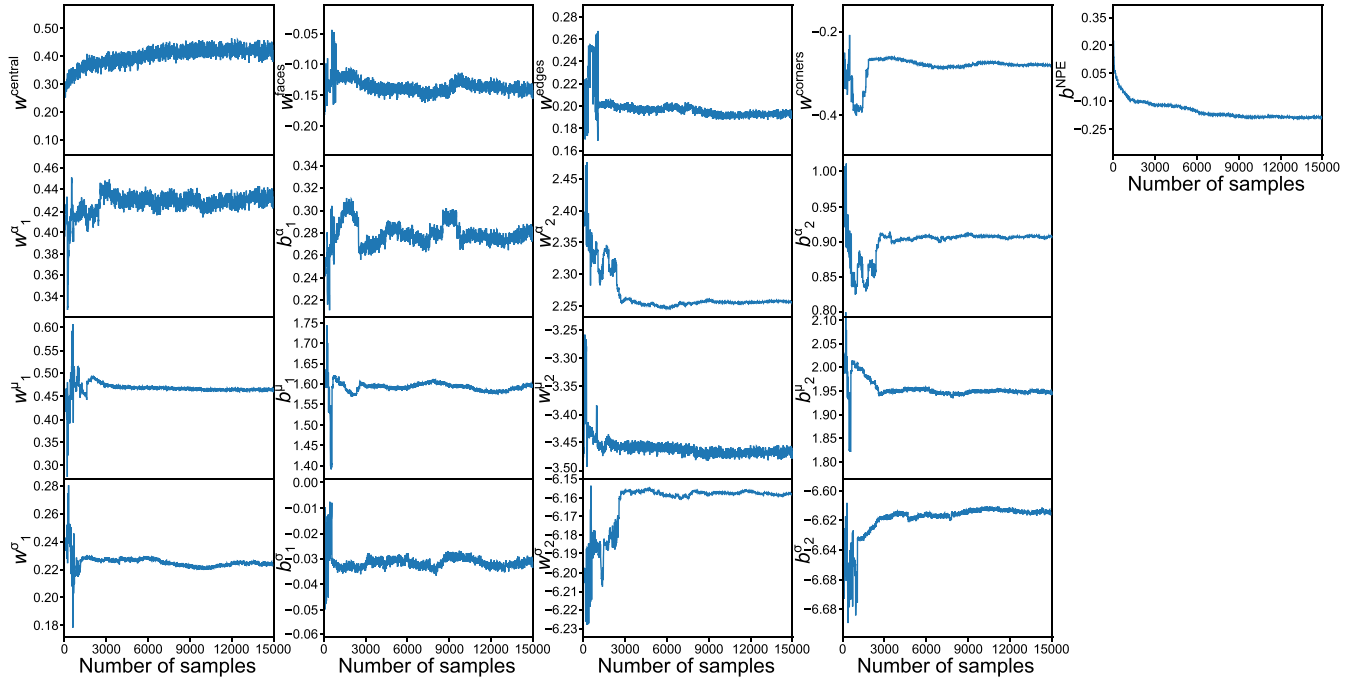

Figure 3. The Markov chain of sampled parameters of the neural bias model. The top row shows, from left to right, the values of the sampled weights of the centre, face, edge, and corner of the $\ell = 0$ convolution kernel, respectively. The final subplot on the top row shows the overall bias parameter of the NPE. The lower three rows show the sampled parameter values of the MDN. The second row shows the weight and bias that parametrize the amplitude of the first and then the second distribution of the mixture of Gaussians. Likewise, the third and fourth rows show the weights and biases parametrizing the mean and standard deviation of the first and second distributions, respectively. We can see that the chain wanders quite wildly until around ∼3000 samples, at which point the chain is properly burnt in and the samples are really being drawn from the posterior distribution.

5.3 Assessment of the model

First, we consider the sampling of the parameters of the neural bias model. The obtained halo mass distribution function is shown in Fig. 2, and the values of the sampled parameters of the model as a function of the Markov chain are shown in Fig. 3.

In Fig. 2, the observed number density of haloes from the central VELMASS simulation halo catalogue is plotted using diamonds (with dashed lines connecting them to help visualization). From top to bottom in number density, the different colours represent decreasing density environments; i.e. voxels with higher densities have a larger halo number density in those voxels. We analyse the effectiveness of the neural bias model by providing it with density patches (of size 3^3 voxels) with an average density equivalent to the values in individual voxels to compare to the halo mass distribution function from the halo catalogue. These 3^3 volumes are randomly drawn to provide a variety of different local patches. The mean result of the neural bias model from the Markov chain is shown using a solid line, and the filled area represents the 1σ deviations from the samples of parameter values of the model. For completeness, we also show the initial values of the parameters of the neural bias model, obtained using stochastic gradient descent, via the dotted line. We can see that there is extremely good agreement between the observed halo number density and that obtained using the neural bias model. In particular, we notice that the two components of the MDN fit the complexities of the halo mass distribution function well. The sampling is also well contained about the observed data, and improved over the initial neural bias model parameter values. The fact that the response to different density environments can be seen provides us with confirmation that the non-linearity of the tracer bias is built into our neural bias model.

Turning our attention to Fig. 3, we can see the sampling of the parameters of the neural bias model. The top row shows the free parameters of the NPE, with the first four subplots showing the weights of the centre, face, edge, and corner of the $\ell = 0$ kernel, respectively. The fifth subplot is the overall bias to the NPE that sets the response scale for activated feature map that will be passed to the mixture of Gaussians. The second, third, and fourth rows show the weight and bias values for the amplitude, mean, and standard deviation of the mixture of Gaussians, respectively. The first two columns show the weight and bias for the first Gaussian of the mixture and the third and fourth columns show the weight and bias for the second Gaussian. Overall, we can see that the values of the samples vary a lot during the first ∼3000 samples, after which burn-in ends and the HMCLET starts to truly sample well from the posterior. The variation in the samples during burn-in occurs even though we pre-initialize the parameters of the neural bias model using stochastic gradient descent. This happens because the rescaling of the momenta in the HMCLET is being learned via the QNHMC, and because the initial density field is not yet conditioned on the observed halo catalogue. Interestingly, we also see that there is some change between the initial and average parameter values. This could be caused by the stochastic gradient descent not properly finding the minimum of the log-likelihood, but which is achieved by the HMCLET. It is more likely to be due to the fact that the density field used to constrain the parameters using stochastic gradient descent is a full *N*-body simulation, while only the LPT field is used in the density sampler, showing that the neural bias model is able to adapt to the missing information due to using only the approximate evolution. We can see that our choice of prior width on the parameter values has not affected the posterior since none of the parameters have vanished, and in fact can be relatively large (in the case of

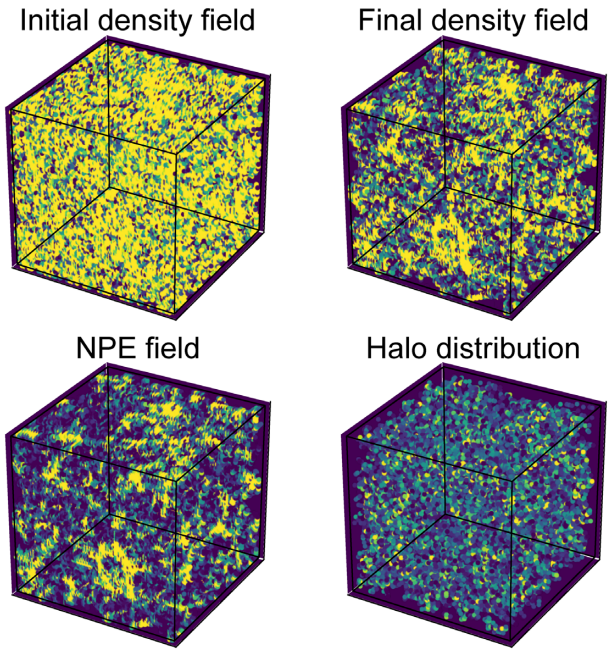

Figure 4. 3D projections of relevant fields. The upper left box shows the 3D projection of the initial dark matter distribution. The upper right box shows the same projection of the dark matter distribution evolved to a scale factor $a = 1$ using LPT. The lower left box shows the output of the NPE, $\boldsymbol{\psi}$, and, for completeness, we show the logarithm of the mass distribution of the halo catalogue in the lower right box. The three density boxes all use the same colour scale. We can see that the production of the non-linear features of evolution by comparing the initial and final density fields (in the top row), while we see the enhancing effect of the non-linear structures due to the NPE in the lower left box. Note the stochastic nature of the halo distribution obtained from the observed halo catalogue compared to the field obtained from the NPE.

the weights and biases for the mean and standard deviation of the second distribution).

By looking at the average samples in the first four subplots on the top row of Fig. 3, we can introspect our physical neural engine. In this case, since we only have one kernel, it is very easy to see the effect of the kernel on the density field. This type of kernel resembles that of a contrast-increasing kernel, which enhances dense regions and washes out underdense areas. In fact, if we look at Figs 4 and 5 we can see, qualitatively, the effect of the NPE. In Fig. 4, the upper left box shows the inferred initial conditions of the dark matter density field, which is then evolved forward using LPT to obtain the box on the upper right. The lower left box then shows the output of the NPE. The lower right box shows the distribution of mass from the halo catalogue, for completeness. We can see the aforementioned increase in contrast of the density field that the NPE provides. The existence of non-zero value for the faces, edges, corners, and central part of the $\ell = 0$ kernel shows us that beyond-local information from the density field is important for the fitting of the halo mass distribution function. The neural bias model therefore makes use of information from the surrounding regions of each voxel to improve the quantification of the number density of haloes with a given mass in each voxel. Likewise, in Fig. 5 we can see that a slice of the average final density field well represents the same slice from the VELMASS simulation with very little variance in the regions of low mass, as can be seen in the bottom right subplot. It should be remembered that the output of the NPE is just a set of summaries that are the most informative about the distribution of haloes with a certain mass in a given environment. The average NPE summary field is quite pixelated in each slice since it contains information about the abundance of haloes from each the density environment of the neighbouring slices.

The halo distribution should be a stochastic sampling of the inferred halo mass distribution function provided by the neural bias model. In the current design, this sampling would be described via a Poisson distribution. Deviations between such a realization and the halo distribution from the catalogue could be due to the nature of the true likelihood that is expected to be non-Poissonian.

Via the BORG algorithm, we also sampled the initial conditions of the dark matter density field (seen in the upper left box of Fig. 4). We show in Fig. 6 the power spectrum of the initial conditions. The orange dashed line is the prior power spectrum from which the initial density field is drawn. The blue transparent lines show the initial power spectrum of the dark matter density field from the posterior samples inferred via BORG. We can see that the inferred power spectra are consistent with the prior, and there are no spurious features. This indicates that the sampling of the density field is self-consistent with the prior provided and the data used to generate the halo catalogue.

As a simple test of how well the inference using the neural bias model, $\mathcal{M}_{\rm Neural\,bias}$, works, we compare it to the bias model of Neyrinck et al. (2014), $\mathcal{M}_{\rm Neyrinck}$, where the Poisson rate for the realization of the halo field is given by

$$\lambda = \bar{N}(1+\delta_{\rm LPT})^{\alpha}\exp\left[-A(1+\delta_{\rm LPT})^{-\varepsilon}\right]. \tag{25}$$

This model is superior to a simple linear bias model since it enhances the contrast between dense regions and voids. There are four free parameters, $\boldsymbol{\theta}_{\rm Neyrinck} = \{\bar{N}, \alpha, A, \varepsilon\}$. By evaluating the value of the normalized likelihood given the VELMASS halo catalogue and a single realization of the $\delta_{\rm LPT}$ field at the maximum-likelihood values of the parameters, $\hat{\boldsymbol{\theta}}_{\rm Neyrinck}$, and comparing it to the normalized likelihood for the neural bias model at the maximum-likelihood values of its parameters, $\hat{\boldsymbol{\theta}}_{\rm Neural\ bias}$, we can get a sense of how the neural bias model performs in comparison to the Neyrinck model. We find that the relative likelihood ratio (Neyman & Pearson 1933) is

$$B(\mathcal{M}_{\rm Neural\,bias}, \mathcal{M}_{\rm Neyrinck}) = \frac{\mathcal{L}(d|\hat{\boldsymbol{\theta}}_{\rm Neural\,bias}, \mathcal{M}_{\rm Neural\,bias})}{\mathcal{L}(d|\hat{\boldsymbol{\theta}}_{\rm Neyrinck}, \mathcal{M}_{\rm Neyrinck})} \approx \exp[2000], \tag{26}$$

which is decisive. Note that we are not suggesting the neural bias model is better than the Neyrinck model since we are not comparing it to a full reconstruction using the bias model of Neyrinck. Instead, we just use this test as a simple comparison to make sure the results of the inference make sense. What we have found is that there is a great potential for this method over classical biasing techniques. A test comparing many models, including an optimal neural bias model, is in preparation using realistic data.

6 CONCLUSIONS

We have presented a neural bias model: a physically motivated neural network-like architecture that maps dark matter density fields to the halo mass distribution function. In doing so, we have used a swathe of new techniques including novel architectures, such as the multipole expansion in convolutional kernels, and well-adapted sampling methods (such as the QNHMC) to provide a Bayesian interpretation of the parameters of the neural bias model.

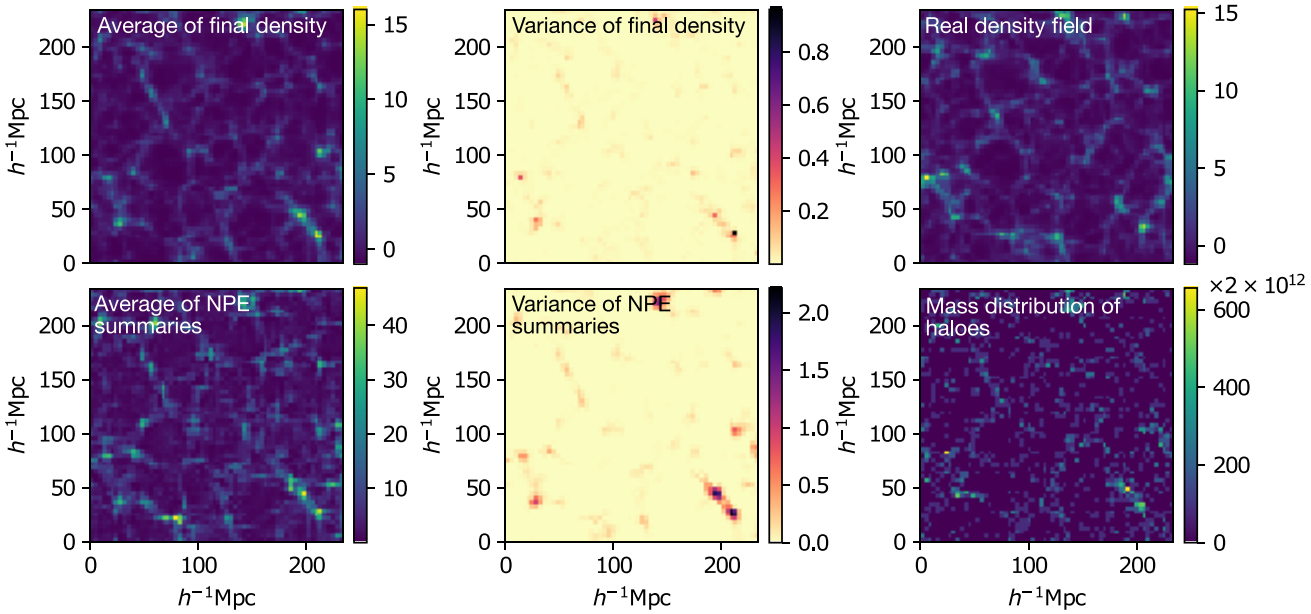

Figure 5. *Left*: A slice of the density field evolved to a scale factor $a = 1$ using LPT averaged over the MC samples from steps 3000 to 15 000 on the top and the same for the output of the NPE. *Middle*: The variance over the same samples of the equivalent slices of the final density field and the summaries from the NPE. *Right*: On the top is a slice of the density field from the VELMASS dark matter simulation and on the bottom is the mass distribution of haloes in the same slice. We can see that the NPE summaries are pixelated in this slice since it is drawing in information from neighbouring slices to be most informative about the stochastic nature of the halo distribution. The average final dark matter density field represents very well the real density field. The variance is small in low-density regions where there is less mass in haloes than in the high-mass regions.

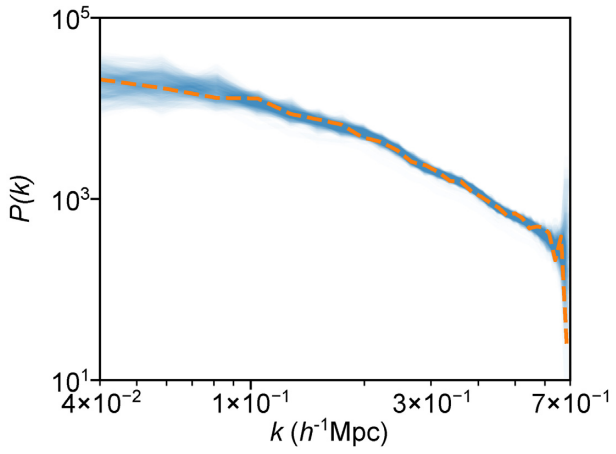

Figure 6. The power spectrum of initial conditions of the dark matter density field. The blue transparent lines show the power spectrum from the posterior samples. The orange dashed line shows the power spectrum of the prior initial dark matter distribution. We can see that the inferred power spectra are consistent with the prior initial power spectrum without any spurious features.

Most importantly, we have shown how physical principles allow us to build extremely efficient neural networks whose parameters can be sampled to provide a truly Bayesian network. The neural bias model becomes part of the forward physical model meaning that the posterior is only conditional on the architecture in the same way as the model that describes the data. No training data are necessary since the weights are inferred directly from the observed data.

We have found that an exceptionally simple neural bias model constructed using a single, rotationally symmetric convolutional kernel and an MDN consisting of a mixture of two Gaussians can effectively model the halo mass distribution function. This neural bias model contains the non-linear response of different density environments and is able to make use of the information from neighbouring patches of the Universe to better predict the abundance of haloes of given masses. Furthermore, the parameters of the model have been inferred using only forward simulations of the dark matter density field and the observed halo catalogue, providing us with a completely Bayesian interpretable network. The simplicity of the model has also allowed us to introspect the neural bias model to see that the kernel enhances dense regions since this informs the halo mass distribution function about where haloes are more likely to be abundant.

This work is proposed as the initial work in a suite of follow-up studies including using Bayesian optimization routines to find the optimal architecture for the NPE and the number and types of distribution for the MDN that allow us to extract maximal information about the halo mass distribution function from the density field. We shall also study the effects of the likelihood that we use to evaluate the density field given the halo catalogue via the replacement of the halo mass distribution function with a neural density estimator emulating the unknown likelihood. This will allow us to search beyond-Poissonity likelihoods. The culmination of this suite of works will be to analyse real cosmological survey data using BORG with a completely agnostic neural bias model, which can be marginalized out to provide constraints on the initial distribution of dark matter, coupled with cosmological parameters, independent of the unknown astrophysics that dictates the tracer bias between the dark matter distribution we use and the observable Universe.

7 ENVIRONMENTAL IMPACT

This study has made use of 103 single core days and 2 single GPU days on a high-performance computing cluster and 90 single core days and 8 full GPU days on a 850-W workstation loaded with an NVIDIA Quadro P6000. This amounts to approximately 1100 kWh, including cooling and data storage. In the Paris metropolitan, this would be equivalent to approximately 55 kg of CO_2.

We have also reused simulations from the VELMASS suite. These simulations were created for the purpose of being a general tool for a wide variety of projects. Its longevity reduces its single-use cost dramatically. This suite of simulations took 23 000 single core days at Occigen facility managed by CINES. The amounts to approximately 5500 kWh and is equivalent to 275 kg of CO_2 in the Hérault metropolitan. The VELMASS suite is stored at l'Institut d'Astrophysique de Paris at a cost of around 260 kWh yr^{-1}, which is approximately 13 kg of CO_2 per year.

All values have been approximated using the Parliamentary Office of Technology document on Carbon Footprint of Electricity Generation and according to l'Agence Internationale de l'Énergie. Exact figures were not available for the power consumption of the computing facilities, and as such a generous approximation has been considered. This is because figures are generally represented in terms of running costs and not in terms of power usage or environmental impact.

ACKNOWLEDGEMENTS

This work was supported by the ANR BIG4 grant ANR-16-CE23-0002 of the French Agence Nationale de la Recherche. TC wishes to thank NVIDIA for the Quadro P6000 used in this work. This work has made use of the Horizon Cluster hosted by Institut d'Astrophysique de Paris. This work has been done within the activities of the Domaine d'Intérêt Majeur (DIM) 'Astrophysique et Conditions d'Apparition de la Vie' (ACAV), and received financial support from Région Île-de-France. The work was granted access to the HPC resources of CINES (Centre Informatique National de l'Enseignement Supérieur) under the allocation A0020410153 and A0040410153 made by GENCI for the VELMASS simulations This work was done as part of the AQUILA consortium.[5] MJH was supported by NSERC (Canada). *Authors' contributions*: TC designed the neural bias model, developed the multipole kernel module, ran the fitting algorithm, contributed to the development of the HMCLET, and wrote the bulk of the content of the paper. GL was the main developer of the HMCLET and ran the VELMASS simulation suite. BDW first proposed the idea of including a neural network and sampling its parameters as part of the BORG algorithm, allowing one to be able to marginalize out uncertainties in the bias model. SSB and MJH proposed the idea at the basis of the likelihood for halo distribution. JJ proposed the use of the QNHMC algorithm. GL and JJ are the main developers of the BORG[3] software used in this work. TC, BDW, and GL are the main proponents of developing neural networks using physical principles.

REFERENCES

Abadi M. et al., 2016, preprint (arXiv:1603.04467)
Bardeen J. M., Bond J. R., Kaiser N., Szalay A. S., 1986, ApJ, 304, 15
Behroozi P. S., Wechsler R. H., Wu H.-Y., 2013, ApJ, 762, 109
Bezanson J., Edelman A., Karpinski S., Shah V. B., 2017, SIAM Rev., 59, 65
Bishop C. M., 1994, Available at:http://publications.aston.ac.uk/id/eprint/373/
Duane S., Kennedy A. D., Pendleton B. J., Roweth D., 1987, Phys. Lett. B, 195, 216
Eisenstein D. J., Hu W., 1999, ApJ, 511, 5
Elsner F., Schmidt F., Jasche J., Lavaux G., Nguyen N.-M., 2020, J. Cosmol. Astropart. Phys., 1, 029
Feng Y., Seljak U., Zaldarriaga M., 2018, J. Cosmol. Astropart. Phys., 2018, 043
Fu T., Luo L., Zhang Z., 2016, UAI'16: Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence. AUAI Press, Arlington, Virginia, US
Goodfellow I., Bengio Y., Courville A., 2016, Deep Learning. MIT Press, Cambridge
Hahn O., Abel T., 2011, MNRAS, 415, 2101
Jasche J., Lavaux G., 2017, A&A, 606, A37
Jasche J., Lavaux G., 2019, A&A, 625, A64
Jasche J., Wandelt B. D., 2013, MNRAS, 432, 894
Jasche J., Leclercq F., Wandelt B. D., 2015, J. Cosmol. Astropart. Phys., 2015, 036
Kaiser N., 1984, ApJ, 284, L9
Kitaura F. S., 2013, MNRAS, 429, L84
Kitaura F. S., Enßlin T. A., 2008, MNRAS, 389, 497
Kitaura F. S., Yepes G., Prada F., 2014, MNRAS, 439, L21
Lavaux G., Jasche J., 2016, MNRAS, 455, 3169
Liu D. C., Nocedal J., 1989, Math. Program., 45, 503
Malmaud J., White L., 2018, J. Open Source Softw., 3, 1002
Mo H. J., White S. D. M., 1996, MNRAS, 282, 347
Neyman J., Pearson E. S., 1933, Phil. Trans. R. Soc. A, 231, 289
Neyrinck M. C., Aragón-Calvo M. A., Jeong D., Wang X., 2014, MNRAS, 441, 646
Nocedal J., Wright S., 2006, Numerical Optimization. Springer Series in Operations Research and Financial Engineering. Springer, New York
Peebles P. J. E., 1980, The Large-Scale Structure of the Universe. Princeton Univ. Press, Princeton, NJ
Planck Collaboration XIII, 2016, A&A, 594, A13
Press W. H., Schechter P., 1974, ApJ, 187, 425
Ramanah D. K., Charnock T., Lavaux G., 2019a, Phys. Rev. D, 100, 043515
Ramanah D. K., Lavaux G., Jasche J., Wandelt B. D., 2019b, A&A, 621, A69
Saito S., Baldauf T., Vlah Z., Seljak U., Okumura T., McDonald P., 2014, Phys. Rev. D, 90, 123522
Sheth R. K., Tormen G., 2004a, MNRAS, 349, 1464
Sheth R. K., Tormen G., 2004b, MNRAS, 350, 1385
Springel V., 2005, MNRAS, 364, 1105
Verlet L., 1967, Phys. Rev., 159, 98
Wang H. et al., 2016, ApJ, 831, 164

APPENDIX: MASS MATRIX FOR QNHMC

The new set of ODEs introduced by the QNHMC yields substantial modifications to the HMC prescription for the optimal mass matrix to sample the parameter space with low rejection rate. This can be seen by considering a Gaussian posterior distribution with covariance $\mathbf{C}$; then

$$\mathcal{V}(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^T\mathbf{C}^{-1}\boldsymbol{\theta}\ . \tag{A1}$$

The approximate inverse Hessian of $\mathcal{V}(\boldsymbol{\theta})$, $\mathbf{B}$, should satisfy $\mathbf{B} \simeq \mathbf{C}$. In this case, equations (18) and (19) become

$$\dot{\boldsymbol{\theta}} = \mathbf{C}\mathbf{M}^{-1}\boldsymbol{p}, \tag{A2}$$

[5]https://www.aquila-consortium.org/

$$\begin{aligned}\dot{\boldsymbol{p}} &= -\mathbf{C}\mathbf{C}^{-1}\boldsymbol{\theta} \\ &= -\boldsymbol{\theta}. \end{aligned} \tag{A3}$$

These two equations can be combined to form a single equation

$$\ddot{\boldsymbol{\theta}} + \mathbf{C}\mathbf{M}^{-1}\boldsymbol{\theta} = \mathbf{0}\ . \tag{A4}$$

To numerically integrate the above equation with a leapfrog integrator in an optimal way, it is best to choose a mass matrix satisfying

$$\mathbf{M} = \mathbf{C}\ . \tag{A5}$$

In most practical cases, we choose a diagonal mass matrix with coefficients that are close to the expected width of the posterior distribution.

This paper has been typeset from a TEX/LATEX file prepared by the author.