Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **509,** 2551–2565 (2022) https://doi.org/10.1093/mnras/stab3114
Advance Access publication 2021 October 28

Multifidelity emulation for the matter power spectrum using Gaussian processes

Ming-Feng Ho,[1]★ Simeon Bird[1]★ and Christian R. Shelton[2]

[1]*Department of Physics & Astronomy, University of California, Riverside, 900 University Ave., Riverside, CA 92521, USA*
[2]*Department of Computer Science & Engineering, University of California, Riverside, 900 University Ave., Riverside, CA 92521, USA*

Accepted 2021 October 22. Received 2021 October 22; in original form 2021 May 13

ABSTRACT

We present methods for emulating the matter power spectrum by combining information from cosmological *N*-body simulations at different resolutions. An emulator allows estimation of simulation output by interpolating across the parameter space of a limited number of simulations. We present the first implementation in cosmology of multifidelity emulation, where many low-resolution simulations are combined with a few high-resolution simulations to achieve an increased emulation accuracy. The power spectrum's dependence on cosmology is learned from the low-resolution simulations, which are in turn calibrated using high-resolution simulations. We show that our multifidelity emulator predicts high-fidelity (HF) counterparts to percent-level relative accuracy when using only three HF simulations and outperforms a single-fidelity emulator that uses 11 simulations, although we do not attempt to produce a converged emulator with high absolute accuracy. With a fixed number of HF training simulations, we show that our multifidelity emulator is $\simeq$100 times better than a single-fidelity emulator at $k \leq 2\,h\mathrm{Mpc}^{-1}$, and $\simeq$20 times better at $3 \leq k < 6.4\,h\mathrm{Mpc}^{-1}$. Multifidelity emulation is fast to train, using only a simple modification to standard Gaussian processes. Our proposed emulator shows a new way to predict non-linear scales by fusing simulations from different fidelities.

Key words: methods: statistical – cosmology: theory – methods: numerical.

1 INTRODUCTION

Current and next-generation large-scale structure surveys, such as DES[1] (Abbott et al. 2020), LSST (Rubin Observatory[2]; Tyson 2002), EUCLID[3] (Amendola et al. 2018), DESI[4] (DESI Collaboration 2016), and the *Roman Space Telescope* (WFIRST; Spergel et al. 2013) will probe gravitational clustering and galaxy formation at small scales with high accuracy. Thus, the future of cosmology relies on exploiting the information in non-linear structure formation at small scales, where numerical *N*-body simulations must be used to give accurate theoretical predictions.

Cosmological linear perturbation theory provides accurate analytic predictions on the clustering of mass up to $k \sim 0.1\,h\mathrm{Mpc}^{-1}$. Despite the success of the standard model of cosmology, several fundamental physics puzzles are still unanswered: the accelerated expansion of the Universe (Caldwell & Kamionkowski 2009), the nature of dark matter (Feng 2010), and the sum of the neutrino masses (Wong 2011). To answer these questions and constrain cosmological parameters using future surveys, theoretical predictions from numerical simulations must be accurate on smaller scales than are accessible to linear theory. As a primary summary statistic, the matter power spectrum needs to be at percent-level precision for $k \lesssim 10\,h\,\mathrm{Mpc}^{-1}$ (Schneider et al. 2016).

Modelling non-linear gravitational clustering is done using *N*-body simulations, where a dark matter fluid is sampled by macro-particles and evolved using a smoothed gravitational force. Each macro-particle is representative of an ensemble of microscopic dark matter particles. Generations of computational physicists have improved the accuracy of the gravitational evolution, and created quicker and more scalable algorithms to drive the mass resolution of the simulations ever higher (Barnes & Hut 1986; Greengard & Rokhlin 1987; Hockney & Eastwood 1988; Couchman, Thomas & Pearce 1995; Dehnen 2002).

The mass resolution necessary to robustly predict the power spectrum at $k \sim 10\,\mathrm{Mpc\,h}^{-1}$ pushes the computational limits of contemporary supercomputers. To adequately sample a high-dimensional input parameter space with Markov chain Monte Carlo (MCMC), millions of samples are needed, while a limited number (at best a few hundred to a few thousand) of high-fidelity (HF) simulations are computationally possible.

An efficient way to perform accurate cosmological inference with a limited number of simulations is to use *emulators*. Emulators are flexible statistical models, usually built with Gaussian processes, which learn the mapping from input cosmological parameters to summary statistics. This reduces the number of costly forward simulations by effectively interpolating the function outputs.

Emulators have been applied extensively in the field of cosmological inference. Heitmann et al. (2006) and Habib et al. (2007) proposed a cosmic calibration project to make percent-level predictions on

★ E-mail: mho026@ucr.edu (MH); sbird@ucr.edu (SB)
[1]https://www.darkenergysurvey.org
[2]https://www.lsst.org
[3]https://sci.esa.int/web/euclid
[4]https://www.desi.lbl.gov

the matter power spectrum using a Bayesian emulator. Heitmann et al. (2009), Lawrence et al. (2010), and Heitmann et al. (2014) implemented this cosmic emulator in their Coyote Universe suite using 37 high-resolution simulations. Heitmann et al. (2016) and Lawrence et al. (2017) designed the Mira-Titan Universe suite to train emulators to make precise theoretical predictions using 36 simulations. The latest Euclid preparation (Euclid Collaboration 2020) runs 250 simulations (3000^3 particles) to prepare their emulator for the matter power spectrum. Besides Gaussian processes, Agarwal et al. (2014) used a neural network to build a cosmic emulator from 6380 *N*-body simulations spanning 580 cosmologies.

Beyond the matter power spectrum, emulators have been trained to predict the halo mass function (Bocquet et al. 2020), the concentration–mass relation for dark-matter haloes (Kwan et al. 2013), the galaxy power spectrum (Kwan et al. 2015), the galaxy correlation function (Zhai et al. 2019), the halo bias (McClintock et al. 2019), weak-lensing peak counts (Liu et al. 2015), the cosmic shear covariance (Harnois-Déraps, Giblin & Joachimi 2019), weak-lensing voids (Davies et al. 2020), the 21 cm signal (Kern et al. 2017), and the Lyman-α 1D flux power spectrum (Bird et al. 2019). They also have been used for inferring beyond-ΛCDM cosmologies (Giblin et al. 2019; Pedersen et al. 2020) and $f(R)$ gravity cosmologies (Ramachandra et al. 2020).

While all these emulators successfully predict summary statistics using HF simulations, one question which remains is how to minimize the number of necessary training simulations to achieve a given accuracy. Here, we demonstrate that building cosmological emulators from simulations can be improved with multifidelity models. Multifidelity models (Kennedy & O'Hagan 2000) minimize the computational cost by combining the predictive power of simulations at different resolutions. They fuse the expensive but accurate *HF* data with cheaply obtained *low fidelity* approximations. One standard model used by the multifidelity emulation is a *multioutput Gaussian process* (Bonilla, Chai & Williams 2008). A multioutput Gaussian process (multioutput GP) generalizes a single-output GP to multiple outputs, while building a cross-covariance function to model the shared information between outputs. In this paper, low fidelity (LF) and HF correspond to simulations at different resolutions. HF simulations have a finer mass resolution while HF simulations have a coarser mass resolution.

To train the multifidelity emulator using as few high-resolution simulations as possible, we also propose a method for selecting HF training samples, based on minimizing the loss computed among the LF simulations. By optimizing the LF emulator's loss, we show that one can efficiently train a multifidelity emulator by avoiding worst-case combinations of the HF training samples.

Computational astrophysicists have used methods similar to multifidelity modelling to minimize the cost of performing high-resolution simulations (Lukić et al. 2015; Chartier et al. 2020). A notable example is Richardson extrapolation (Richardson 1911), a numerical method to improve a simulation's accuracy by combining a sequence of simulations with varied spatial resolutions and fixed cosmologies. More recently, generative adversarial networks (GAN) have been used to produce high-resolution density fields (Kodi Ramanah et al. 2020) and particle displacements (Li et al. 2020) from low-resolution (but larger volume) input data. In principle, such 'super-resolution' simulations could be implemented as a multifidelity emulator's HF training set, allowing an emulator to be built to a scale not directly accessible to simulations.

Rogers et al. (2019) and Leclercq (2018) proposed using Bayesian optimization to improve emulator accuracy by a sequential choice of new simulation points designed to globally optimize the emulator function. Similar approaches to iterative selection of training data in a cosmological parameter space have been presented by Takhtaganov et al. (2019) and Pellejero-Ibañez et al. (2020). Computer scientists and engineers, including Huang et al. (2006), Forrester, Sóbester & Keane (2007), Lam, Allaire & Willcox (2015), Poloczek, Wang & Frazier (2016), and McLeod, Osborne & Roberts (2017), have extensively studied combining multifidelity methods with Bayesian optimization.[5] Multifidelity Bayesian optimization arises when a cheaper approximation to the object function exists.

We present a multifidelity emulator for the matter power spectrum, as output by the cosmological simulation code MP-GADGET (Springel & Hernquist 2003; Feng et al. 2018a). In this paper, we target percent level *relative accuracy*: how well our emulators can reproduce the matter power spectra at our highest fidelity. We defer producing an emulator which allows percent level accurate reconstruction of observations or a hypothetical ideal simulation to future work. The main goal of this paper is to demonstrate that our multifidelity techniques can be used to reduce the computational budget required for an emulator.

We use two fidelities in a $256\,\mathrm{Mpc\,h^{-1}}$ box: a fast but low-resolution version with 128^3 dark-matter particles and a slow but high-resolution version with 512^3 particles. Even with only three HF simulations and 50 HF simulations, we show that we can predict the high-resolution matter power spectrum at percent-level accuracy on average at $k \leq 6.4\,h\,\mathrm{Mpc}^{-1}$ at $z = 0$, with a total computational cost $\lesssim$4 HF simulations. Although we only show our application to the matter power spectrum, the methods presented in this paper could apply to other summary statistics, e.g. the halo mass function or the Lyman-α 1D flux power spectrum.

van Daalen et al. (2011) showed that the lack of AGN feedback affects a dark matter-only simulation significantly (compared to the error requirements of upcoming surveys) at $k > 0.1\,h\,\mathrm{Mpc}^{-1}$. Furthermore, baryon cooling can alter the power spectrum at $k \sim 10\,h\,\mathrm{Mpc}^{-1}$ (White 2004). However, as techniques exist to model this effect in post-processing (Schneider et al. 2020), we defer extending our technique to hydrodynamical simulations including AGN feedback to future work. Here, we validate that a multifidelity emulator is useful in the simplest case: dark matter-only *N*-body simulations.

We build two types of multifidelity emulators. One uses the linear autoregressive model of Kennedy & O'Hagan (2000; first-order autoregressive model, AR1), which we will call the "linear multifidelity model." The second multifidelity emulator uses the non-linear fusion model of Perdikaris et al. (2017) (nonlinear autoregressive Gaussian process, NARGP), and which we call the "non-linear multifidelity emulator."[6] Kennedy & O'Hagan (2000) model the scaling factor between fidelities as a scalar, while Perdikaris et al. (2017) allow the scaling factor to depend on input parameters. Our implementation of AR1 and NARGP is based on EMUKIT (Paleyes et al. 2019),[7] an open-source package for emulation and decision making under uncertainty, with the modifications mentioned above.[8]

[5] Frazier (2018) has a subsection that provides a short review on multifidelity Bayesian optimization.

[6] AR1 and NARGP are acronyms used in Perdikaris et al. (2017) and Cutajar et al. (2019). In this paper, AR1 and linear multifidelity emulator are interchangeable, and NARGP and non-linear mutlifidelity emulator are interchangeable.

[7] https://github.com/EmuKit/emukit

[8] For a detailed comparison between AR1 and NARGP, see Cutajar et al. (2019). An example code for the comparison between AR1 and NARGP can be found in Emukit's examples.

In Section 2, we briefly describe the simulation code, MP-GADGET, for training the emulator. We recap the general formalism of a single-fidelity GP emulator in Section 3. Section 4 describes the formalism of a multifidelity emulator (`MFEmulator`). We explain our sampling strategy in Section 5. Section 6 shows the results, with comparisons between multifidelity emulation and single-fidelity emulation. We summarize the runtime for the MP-GADGET simulations in Section 7. We conclude with a summary of key contributions and potential applications of our work in Section 8. Our code for multifidelity emulation in the matter power spectrum is publicly available at https://github.com/jibanCat/matter_multi_fidelity_emu.

2 SIMULATIONS

We prepare our training set by running dark matter-only simulations using the massively parallel N-body code MP-GADGET (Feng et al. 2018a).[9] MP-GADGET is a publicly available N-body + Hydro cosmological simulation code derived from GADGET3 (Springel & Hernquist 2003). It is parallelized using a hybrid OpenMP/MPI strategy and has successfully performed a hydrodynamical simulation using all 8 032 *Frontera* nodes, a total of 449 792 cores, demonstrating its good scalability properties. The gravitational forces are computed using a Fourier transform-based particle-mesh algorithm on large scales and a Barnes-Hut tree on small scales.

We initialize our simulations from the linear power spectrum produced by CLASS (Lesgourgues 2011) at $z = 99$ using the Zel'dovich approximation (Zel'Dovich 1970). The dark matter particles then evolve through gravitational dynamics. The matter power spectra are computed from the output snapshots of MP-GADGET, and used as our emulation targets. In this paper, we fix the initial condition (IC) noise in the nodes and change only the cosmology for the emulator training. We do not use the "paired and fixed" technique (Angulo & Pontzen 2016), but it would be easy to do so using only low resolution simulations as these pairings are designed to remove variance on large scales.

The matter power spectrum, $P(k)$, is a compressed summary statistic of the over-density field, $\delta(x)$, evaluated as an angle average of the Fourier-transformed overdensity field:

$$P(|\boldsymbol{k}|) = \langle \hat{\delta}^*(k)\hat{\delta}(k)\rangle, \tag{1}$$

$$\hat{\delta}(\boldsymbol{k}) = \int \mathrm{d}^3\boldsymbol{r}\delta(\boldsymbol{r})e^{-2\pi i\boldsymbol{k}\cdot\boldsymbol{r}}. \tag{2}$$

We measure the power spectrum with a cloud-in-cell mass assignment, which is deconvolved. The Fourier transform is taken on a mesh same as the PM grid of the simulation, which has a resolution of two times the mean inter-particle spacing.

For a multifidelity problem, our data are from simulations at different resolutions. Since low-resolution simulations are cheaper to obtain (but are only approximations to the high-resolution results), we typically have a limited number of HF data and many LF approximations.

To make the text of this section consistent with the following sections, we provide some notation to bridge the terminology, summarized in Table 1. We have data from s different fidelities (simulation resolutions). For each fidelity, we have pairs of inputs and outputs $\mathcal{D}_t = \{x_{i,t}, y_{i,t}\} = \{\boldsymbol{x}_t, \boldsymbol{y}_t\}$, where $t = 1, \ldots, s$ denotes the fidelity level from low to high, and $i = 1, \ldots, n_t$, where n_t is the number of data pairs at fidelity t and i indexes each individual simulation. The data pairs $\mathcal{D}_t = \{\boldsymbol{x}_t, \boldsymbol{y}_t\}$ for our emulation setup are the cosmological parameters of the simulations and the power spectrum outputs. Here, we have $s = 2$ for two mass resolutions: 128^3 and 512^3 dark matter-only simulations. We will denote 128^3 as low-resolution (LR, $t = 1$) and 512^3 as high-resolution (HR, $t = 2$).

[9] https://github.com/MP-Gadget/MP-Gadget/

Table 1. Notations and definitions.

Notation	Description
HR	High-resolution simulation, 512^3 particles
LR	Low-resolution simulation, 128^3 particles
$x_{i,t}$	Input cosmological parameters at ith simulation at fidelity t
$y_{i,t}$	Matter power spectrum at ith simulation at fidelity t, at log scale
n_t	Number of simulations at fidelity t
$N_{\mathrm{ptl,\,side}}$	Number of particles per box side

Each fidelity will have a different number of simulations, n_t. Practically, the number of LR simulations will be much larger than the number of HR simulations, $n_1 > n_2$. The compute time for LR ($N_{\mathrm{ptl,\,side}} = 128$) is $\sim$20 and $\sim$2 000 core hours for HR ($N_{\mathrm{ptl,\,side}} = 512$). We will empirically show that we only need 3 HR and 50 LR to train a multifidelity emulator with an average emulator error per k smaller than 1 per cent.

We do not emulate the matter power spectrum across redshifts, conditioning on a given redshift bin z_0. We generally focus on $z_0 = 0$, but will discuss multifidelity emulators at $z_0 = 1$ and $z_0 = 2$ in Section 6.4.2.

2.1 Latin hypercube sampling

As Heitmann et al. (2009) mentioned, a space-filling Latin hypercube design is well suited for GP emulators of the matter power spectrum. For a training set with d-dimensional inputs and N simulations, an N^d grid is created first, and simulations are placed on this grid so that only one simulation is present in any row or column. The Latin hypercube design improves on random uniform sampling by ensuring that the chosen points do not crowd together in any subspace.

We apply a Latin hypercube design on the input parameter space, $\{h, \Omega_0, \Omega_b, A_s, n_s\}$. We vary the ΛCDM cosmological parameters $\{h, \Omega_0, \Omega_b, A_s, n_s\}$, which are the Hubble parameter $h = H_0/(100\,\mathrm{km\ s^{-1}\,Mpc^{-1}})$, the total matter density Ω_0, the baryon density Ω_b, primordial amplitude of scalar fluctuations A_s, and the scalar spectral index n_s. We use the same set of ΛCDM cosmological parameters as Euclid Collaboration et al. (2020), allowing us to compute the relative errors of our simulations with respect to EuclidEmulator2.

We use bounded uniform priors for the input parameters:

$$\begin{aligned} h &\sim \mathcal{U}[0.65, 0.75];\\ \Omega_0 &\sim \mathcal{U}[0.268, 0.308];\\ A_s &\sim \mathcal{U}[1.5\times10^{-9}, 2.8\times10^{-9}];\\ n_s &\sim \mathcal{U}[0.9, 0.99];\\ \Omega_b &\sim \mathcal{U}[0.0452, 0.0492]. \end{aligned} \tag{3}$$

The dark energy density is $\Omega_\Lambda = 1 - \Omega_0$. The prior volume surrounds the WMAP 9-yr cosmology (Hinshaw et al. 2013). The code to handle the simulation input files and Latin hypercube design is publicly available at https://github.com/jibanCat/SimulationRunnerDM.

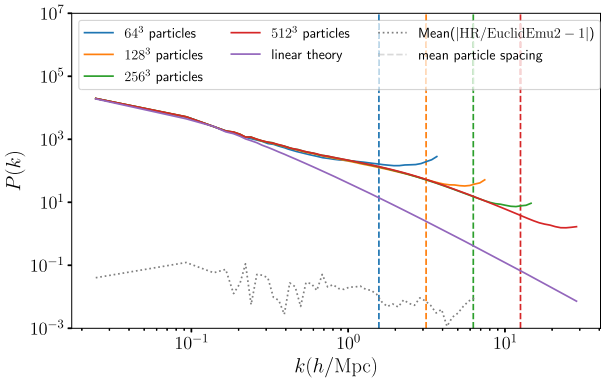

Figure 1. The matter power spectrum output by MP-GADGET at different mass resolutions. The vertical dash lines indicate the mean particle spacing k_{spacing} for a given mass resolution. *(Blue):* the matter power spectrum from a dark-matter only MP-GADGET simulation with 64^3 particles. *(Orange):* the matter power spectrum from MP-GADGET with 128^3 particles. *(Green):* the matter power spectrum from MP-GADGET with 256^3 particles. *(Red):* the matter power spectrum from MP-GADGET with 512^3 particles. *(Purple):* linear theory power spectrum. The cosmology parameters are $h = 0.675$, $\Omega_0 = 0.278$, $\Omega_b = 0.0474$, $A_s = 1.695 \times 10^{-9}$, and $n_s = 9.405 \times 10^{-1}$. The dotted line shows the relative error of HR (512^3 simulations) compared with EuclidEmulator2 (Euclid Collaboration 2020), averaged over four different cosmologies.

2.2 Preprocessing of the simulated power spectrum

A numerical simulation is constrained by its box size and number of particles. The mass resolution limits the smallest scale (the highest k) of the power spectrum. Thus, HF simulations can model smaller scales, not fully resolved in LF simulations, as shown in Fig. 1.

For k larger than the mean particle spacing, $P(k)$ differs substantially from the resolved value, due to artefacts of the macro-particle sampling. The scale of the mean particle spacing is

$$k_{\mathrm{spacing}} = 2\pi \frac{N_{\mathrm{ptl,side}}}{L_{\mathrm{box}}}, \tag{4}$$

where $N_{\mathrm{ptl,\,side}}$ is the number of particles per side of the box. For instance, if we have 512^3 particles in the box, then $N_{\mathrm{ptl,\,side}} = 512$. L_{box} is the size of the simulation box in units of $\mathrm{Mpc\,h^{-1}}$.

We use the same set of matter power spectrum k bins for all fidelities. The available information at small scales is sparse for the LF spectrum. To resolve the issue, we fix the k bins to HF and linearly interpolate the LF power spectrum in a $\log_{10}$ scale, $\log_{10}P(k)$, on to the HF k bins. The maximum k is set to be $\simeq 6.4\,h\mathrm{Mpc}^{-1}$ when using $N_{\mathrm{ptl,\,side}} = 128$ as our LF training set. However, in practice, we found that 128^3 and 512^3 simulations shared similar k bins with small offsets at small scales.

We do not model the HF spectrum with k larger than the maximum k of the LF spectrum:

$$\max k_{t=2} = \max k_{t=1}, \tag{5}$$

where t indicates the fidelity level and $t = 2$ is the highest fidelity. If we do not have any data at a given k from LF, we cannot extract the correlations between fidelities without other more significant assumptions. In other words, the maximum k we can model is limited by the data available from the LF simulations, which always have a lower maximum k than HF simulations. We note that it is possible to get a higher maximum k by particle folding or by increasing the size of the PM grid size used for estimating the power spectrum, although we do not do that here.

We do model the LF $P(k)$ even on scales smaller than the mean particle spacing, $k > k_{\mathrm{spacing}}$. We made this particular decision because we have a prior belief that even though $P(k > k_{\mathrm{spacing}})$ is highly biased, it still captures some information about how $P(k)$ depends on cosmological parameters. Thus, we should be able to exploit the correlations between fidelities and improve the emulator accuracy at those scales.

To summarize, we:

(i) Use the same set of k bins across different fidelities.
(ii) Preserve all available $P(k)$ from LF, even scales smaller than the simulation's mean particle spacing.

3 SINGLE-FIDELITY EMULATORS

Here, we briefly recap how we train a single-fidelity emulator. Readers familiar with this material may wish to skip to Section 4. The notation we use in this section follows those of Perdikaris et al. (2017) and Cutajar et al. (2019). Consider a supervised learning problem, in which we wish to learn the mapping relation, f, between a set of input and output pairs $\mathcal{D} = \{x_i, y_i\} = \{\boldsymbol{x}, \boldsymbol{y}\}$, where $i = 1, \ldots, n$:

$$y = f(x), \quad \text{with } x \in \mathbb{R}^d, \tag{6}$$

where d is the dimension of the input space. A GP (Rasmussen & Williams 2005) is a probabilistic framework modelling the observations, $\boldsymbol{y}$, as drawn from a noisy realization of a single random function f with a likelihood $p(\boldsymbol{y} \mid f)$. It models the distribution over f

$$p(f) = \mathcal{GP}(f; \mu, K), \tag{7}$$

with μ the GP mean prior function, which is usually assumed to be a zero mean prior, and K the covariance kernel function specified by a vector of hyperparameters, $\boldsymbol{\theta}$. For a given set of inputs, x_1, x_2, …, x_n, the kernel function evaluated on these points produces a symmetric, positive-definite covariance matrix $K_{ij} = K(x_i, x_j; \boldsymbol{\theta})$ with $K \in \mathbb{R}^{n\times n}$.

The choice of the covariance kernel depends on our prior knowledge about the data. The hyperparameters of a chosen kernel are optimized by maximizing the marginal log-likelihood:

$$\log p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = -\frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\boldsymbol{y}^\top \mathbf{K}^{-1}\boldsymbol{y} - \frac{n}{2}\log 2\pi. \tag{8}$$

For an emulator, the main purpose is to predict an output $f_* = f(x_*)$ from a new input point x_*, given the provided data $\mathcal{D}$.

$$\begin{aligned} p(f_* \mid \mathcal{D}, x_*) &= \mathcal{N}\left(f_* \mid \mu_*(x_*), \sigma_*^2(x_*)\right), \\ \mu_*(x_*) &= \boldsymbol{k}_{*n}\mathbf{K}^{-1}\boldsymbol{y}, \\ \sigma_*^2(x_*) &= K(x_*, x_*) - \boldsymbol{k}_{*n}\mathbf{K}^{-1}\boldsymbol{k}_{*n}^\top, \end{aligned} \tag{9}$$

where μ_* is the posterior mean and σ_* is the standard deviation of the uncertainty in the estimate of the predictions. The vector $\boldsymbol{k}_{*n}$ is the covariance between the new point and trained data, $\boldsymbol{k}_{*n} = [K(x_*, x_1), \ldots, K(x_*, x_n)]$.

3.1 Cosmological emulators

Consider we have a set of dark matter-only simulations with fixed box size and mass resolution. At each redshift bin z_0, we can compute the matter power spectrum, $P(k, z = z_0)$, given a set of input parameters. We will use the log power spectrum, $\log_{10}P(k, z = z_0)$, as our training data.

The training data, $\mathcal{D} = \{x_i, y_i\}$, are defined as

$$x_i = [h_i, \Omega_{0i}, \Omega_{bi}, A_{si}, n_{si}];$$

$$y_i = \log_{10} P(k, z = z_0),$$

where $i = 1, \ldots, n$ indicates the ith simulation we run with this specific set of input parameters.

The rest of the modelling is choosing an appropriate covariance function $K(x, x')$. We use a squared exponential kernel and use automatic relevance determination (ARD) weights for each input dimension. ARD assigns each input dimension, x_i, a separate hyperparameter, w_i:

$$K(x, x'; \boldsymbol{\theta}) = \sigma^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{d} w_i \left(x_i - x_i'\right)^2\right) \tag{10}$$

where $i = 1, \ldots, d$ indicates the dimension of the input space $x \in \mathbb{R}^d$. σ^2 is the variance parameter for the squared exponential kernel, $\{w_i\}_{i=1}^d$ are the ARD weights. $\{w_i\}_{i=1}^d$ are inverse length scales, which define the degree of smoothness at a given input dimension. We note that we assign independent hyperparameters, $\boldsymbol{\theta} = \{\sigma^2, w_1, \ldots, w_d\}$, for each k mode.[10] A larger w_i corresponds to a smaller length scale, reflecting that the learned function varies more in the ith dimension. On the other hand, a smaller w_i implies a larger length scale, indicating that the learned function is smoother along the ith dimension. ARD allows each dimension of the learned function to have a different degree of smoothness.

We do not decompose the power spectrum into principle components for training the emulators, as described by Heitmann et al. (2006) and Habib et al. (2007) because we want to compare single-fidelity emulators to the multifidelity emulators, and an `MFEmulator` only has a limited number of high-resolution simulations available. In our default case, we only have three high-resolution simulations for an `MFEmulator`, and it is not sensible to perform dimension reduction on three power spectra.

To ensure that our single-fidelity emulator is not unfairly disadvantaged in the comparison with our multifidelity emulator by poorly constrained hyperparameters, we built a single-fidelity emulator that shared kernel parameters across all k modes and empirically verified that it had a similar performance.

4 MULTIFIDELITY EMULATOR

In this section, we describe how we train a multifidelity emulator. We outline the modelling assumptions in Section 4.1. Section 4.2 describes the formalism of the linear multifidelity emulator proposed by Kennedy & O'Hagan (2000), a multioutput GP with a linear correlation between fidelities. Section 4.3 outlines the non-linear multifidelity emulator of Perdikaris et al. (2017), which models the correlation between fidelities as a function of cosmological parameters. We follow the notation and formalism of Kennedy & O'Hagan (2000), Perdikaris et al. (2017), and Cutajar et al. (2019).

4.1 General assumptions

Here, we outline our modelling assumptions, following the assumptions made in Kennedy & O'Hagan (2000):

(i) *Correlations between the code fidelities:* For an N-body simulation, the simulation cost depends on the mass resolution. We assume a simulation with a low-mass resolution can approximate a simulation with a high-mass resolution. The matter power spectrum from different fidelities is strongly correlated at large scales since all fidelities are resolved and the mass resolution has negligible effects. At small scales, however, we expect different fidelities are only weakly correlated.

(ii) *Smoothness:* For an emulation problem, we assume that neighbouring inputs give similar outputs. For example, suppose two sets of input parameters to MP-GADGET are close to each other. In that case, we assume that an N-body simulation will provide a similar outcome.

(iii) *The prior belief on each fidelity is a Gaussian process:* We assume a prior belief that the mapping from code input to output is a GP for each fidelity.

The first assumption is the core assumption of a multifidelity emulator. Different levels of the same code are simulating the same physical reality. It is thus reasonable to assume that different code fidelities should correlate at some level. However, a naive simulation, for example, $N_{\rm ptl,\,side} = 16$ could only barely approximate a HR with $N_{\rm ptl,\,side} = 512$. Therefore, we should also assume the correlation between fidelities depends on the distance between two fidelities in the dimension of mass resolution.

There is thus a trade-off between the strength of correlation and the computational expense: for example, a simulation with $N_{\rm ptl,\,side} = 256$ provides more information about a HR ($N_{\rm ptl,\,side} = 512$), but running a 256^3 simulation is eight times most expensive than running a LR ($N_{\rm ptl,\,side} = 128$).

One can select an optimal choice of simulation cost by balancing the computational time and the emulation accuracy. Here, we choose $N_{\rm ptl,\,side} = 128$ for our LF simulations because:

(i) The maximum k is $\simeq 6.4\,h\,{\rm Mpc}^{-1}$, which includes enough non-linear scales to test the emulation accuracy;
(ii) A 128^3 simulation is 64 times cheaper than a HR, and thus the resolution difference between $N_{\rm ptl,\,side} = 128$ and $N_{\rm ptl,\,side} = 512$ is large enough to demonstrate whether simulations with lower costs can accelerate the training of an emulator.

In Section 6.4.1, we will show our method is applicable to simulations with different resolutions, $N_{\rm ptl,\,side} = 64$ and 256. Empirically, we found that using $N_{\rm ptl,\,side} = 256$ as LF is similar to $N_{\rm ptl,\,side} = 128$, while $N_{\rm ptl,\,side} = 64$ gives a worse emulation accuracy.

The second assumption, the smoothness assumption, is the general assumption of a GP emulator. A GP emulator will have poor accuracy if the code does not behave similarly with similar input. The smoothness assumption is also the assumption behind the Latin hypercube sampling scheme (for a detailed discussion, see Heitmann et al. 2009).

A multifidelity emulation could, in principle, be implemented using other models (see Peherstorfer, Willcox & Gunzburger (2018) for different data-fitting models for surrogates). We chose to use GPs simply because their Bayesian approach supports uncertainty quantification and there is a well-developed community around GP emulation.

4.2 Linear multifidelity emulator (AR1)

We have multifidelity data $\mathcal{D}_t$ as described in Section 2. A multifidelity emulator is essentially inferencing the highest fidelity model conditioned on data from all model fidelities. The final goal of a

[10] Takhtaganov et al. (2019) refers to this approach as the many single-output approach (MS).

multifidelity emulator is to find a mapping relation f such that, from an arbitrary input vector x_*, we can always find the highest fidelity code output:

$$y_{s,*} = f(x_*). \tag{11}$$

As described by Kennedy & O'Hagan (2000), a linear autoregressive model can be applied in a multifidelity setting by assuming a hierarchical order between fidelities:

$$f_t(x) = \rho_t\, f_{t-1}(x) + \delta_t(x), \tag{12}$$

where f_t is the function emulated by a GP at t fidelity and f_{t-1} is the function emulated at the previous fidelity level $(t-1)$. The linear component of equation (12) is ρ_t, which models the correlation between fidelities as a linear relation. δ_t is a GP modelling the bias term:

$$\delta_t \sim \mathcal{GP}(\mu_{\delta_t}, K_t). \tag{13}$$

We modify equation (12) so inference is performed on each k bin independently. For $k = k_j$, we have independent kernel and scaling parameters for each $k = k_j$ mode. For simplicity, we will drop the $k = k_j$ notation in the rest of the paper:

$$f_t(x) = \rho_t(f_{t-1}(x) - \mu_{t-1}) + \delta_t(x). \tag{14}$$

The mean of the bias term, μ_{δ_t}, is assumed to be the zero function. For the LF part, we subtract the sample mean of the logarithm training power spectra, $\log_{10}P(k)$, and model the LF part of the power spectra as a zero mean GP:

$$(f_1(x) - \mu_1) \sim \mathcal{GP}\left(0, K_1\left(x_1, x_1'; \boldsymbol{\theta}_1\right)\right). \tag{15}$$

As shown in Fig. 1, the LF power spectrum is biased high. We pass variations of the LF power spectrum around its mean to the next fidelity to avoid passing biased outputs. In practice, we found that this slightly improves emulation accuracy for multifidelity models.

For the highest fidelity bias function, $\delta_s(x)$, we model the power spectrum using a zero mean GP without subtracting the sample mean. We do not have enough points at the highest fidelity for the sample mean to be a good estimate of the true mean. Except for $t = 1$, $f_t(x)$ is completely determined by $f_{t-1}(x)$, $\delta_t(x)$, and ρ_t.

As mentioned by Kennedy & O'Hagan (2000), there is a Markov property implied in the covariance structure of equation (12):

$$\mathrm{cov}\left\{f_t(x),\, f_{t-1}(x') \mid f_{t-1}(x)\right\} = 0, \tag{16}$$

which is true for all $x \neq x'$. Equation (16) indicates that if we have $f_{t-1}(x)$, then other input parameters $f_{t-1}(x')$ do not contribute to training $f_t(x)$.

The Markovian property also suggests that an efficient training set $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_s\}$ for a multifidelity GP is a nested structure:

$$\boldsymbol{x}_1 \subseteq \boldsymbol{x}_2 \subseteq \cdots \subseteq \boldsymbol{x}_s. \tag{17}$$

The above notation says that, given an input point x at fidelity t, there must be an input x in its lower fidelity u, where $u < t$ and t, $u \in \{1, 2, \cdots, s\}$. The reason for using a nested experimental design is that since we have $\boldsymbol{x}_{t-1} \subseteq \boldsymbol{x}_t$, we can immediately get an accurate posterior $f_{t-1}(x)$ at the x location without interpolating at the $t-1$ level. However, in practice, we found that our multifidelity emulators performed well even without a nested design in the input space.[11]

[11]Without a nested design in input space, we found, for a multifidelity emulator using 50 LR and 3 HR, the non-nested one is only 5 per cent worse than the nested one on the relative errors.

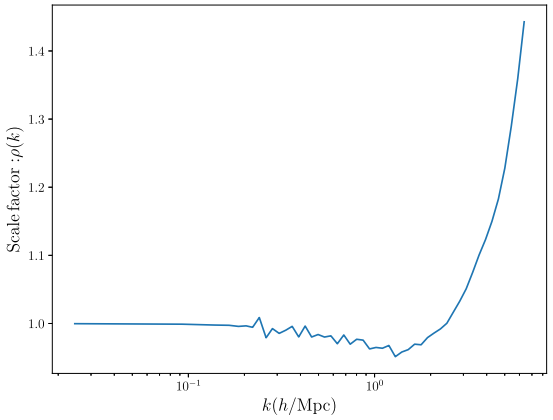

Figure 2. The learned scale factor between fidelities in the linear multifidelity model, ρ, as a function of k. This scale factor is learned from 50 LF simulations and 3 HF simulations.

At a given fidelity t, the posterior at a test input x_* could be written as

$$p(f_{*t} \mid \mathcal{D}, x_*) = \mathcal{N}\left(f_{*t}; \mu_{*t}(x_*), \sigma^2_{*t}(x_*)\right), \tag{18}$$

where we denote predictions from new inputs as subscript $*$. The predictive mean and variance are

$$\begin{aligned}\mu_{*t} &= \rho_t \cdot \mu_{*t-1}(x_*) + \mu_{\delta_t} \\ &\quad + \boldsymbol{k}_{*n_t}\mathbf{K}_t^{-1}[\boldsymbol{y}_t - \rho_t \cdot \mu_{*t-1}(\boldsymbol{x}_t) - \mu_{\delta_t}]; \\ \sigma^2_{*t} &= \rho_t^2 \cdot \sigma^2_{*t-1}(x_*) + K(x_*, x_*) - \boldsymbol{k}_{*n_t}\mathbf{K}_t^{-1}\boldsymbol{k}^{\top}_{*n_t},\end{aligned} \tag{19}$$

where $\boldsymbol{k}_{*n_t} = [K_t(x_*, x_1), \ldots, K_t(x_*, x_{n_t})]$ is a vector of covariance between the new location and the training locations at fidelity t. $K_t = K_t(\boldsymbol{x}_t, \boldsymbol{x}_t')$ is the covariance matrix of training locations at fidelity t.

4.2.1 *Covariance kernel*

For a linear multifidelity emulator, we place an independent squared exponential kernel on each k_j. The mathematical form of the kernel is same as equation (10).

Having ARD weights means we assign different length scales to each dimension so that the kernel can be trained anisotropically. We found that using ARD in the highest fidelity did not improve the model's accuracy. Thus, we decided to assign an isotropic kernel for δ_s. For a two-fidelity emulator ($s = 2$), we have six hyperparameters in LF for each k bin; five of them are the length scale parameters; and one is the variance parameter. We have three hyperparameters for each k bin in HF, with one scale factor ρ_t between fidelities, one variance parameter, and one length scale parameter. We have 49 bins in k, so the total number of trainable hyperparameters is 441.

Fig. 2 shows the learned scale factor, ρ.[12] ρ is roughly unity at large scales $k \leq 2\,h\mathrm{Mpc}^{-1}$, but its value increases dramatically after $k > 2\,h\,\mathrm{Mpc}^{-1}$. Non-linear physics becomes important and the LF simulations become less reliable at small scales, making the relationship between fidelities non-trivial. We want to emphasize that the scale factor, ρ, is learned from the multifidelity emulator. We did not enforce ρ to be a specific shape during the training. Because

[12]The multifidelity scale factor shown Fig. 2 is ρ_2, which is ρ_t when $t = 2$. For simplicity, we use ρ to refer to ρ_2 for our multifidelity emulators.

we learn the mapping from LR to HR using the training data, it is expected that LR runs deviate from HR power spectra. The purpose of multifidelity emulation is to correct these deviations.

4.3 Non-linear multifidelity emulator (NARGP)

The linear multifidelity model in equation (12) assumes the scale factor ρ_t is independent of input parameters, x, and so does not model the cosmological dependence of the scale factor ρ_t. The non-linear multifidelity model proposed by Perdikaris et al. (2017) drops this assumption, allowing the scale factor, $\rho_t(\cdot)$, to be a function of both input cosmology and output from the previous fidelity. As for the linear multifidelity model, we model the non-linear multifidelity GP independently for each k:

$$f_t(x) = \rho_t(x, f_{t-1}(x) - \mu_{t-1}) + \delta_t(x), \tag{20}$$

where $\rho_t(\cdot)$ is a function of both input parameters x and the previous fidelity's output. $\rho_t(\cdot)$ is modelled as a GP equation (20) results in a more complicated distribution over f_t, a deep GP (Damianou & Lawrence 2013). To avoid added computational and statistical complexity, we follow the same approximation as Perdikaris et al. (2017) and replace f_{t-1} in equation (20) with its posterior, f_{*t-1}. The result is a regular GP,

$$f_t \sim \mathcal{GP}(0, K_t), \tag{21}$$

whose kernel can be furthermore decomposed:

$$K_t(x, x') = K_{t_\rho}(x, x'; \boldsymbol{\theta}_{t_\rho}) \cdot K_{t_f}(f'_{*t-1}(x), f'_{*t-1}(x'); \boldsymbol{\theta}_{t_f}) + K_{t_\delta}(x, x'; \boldsymbol{\theta}_{t_\delta}), \tag{22}$$

where $f'_{*t-1} \equiv f_{*t-1}(x) - \mu_{t-1}$ for simplicity. The first kernel K_{t_ρ} models the cosmological dependence of the scale factor ρ. Next, K_{t_f} models the covariance of the output passing from the previous fidelity to the current level. The final term K_{t_δ} models the model discrepancy between fidelities. For the lowest fidelity, the matter power spectrum is only modelled with K_{t_δ}.

Each kernel in equation (22), $(K_{t_\rho}, K_{t_f}, K_{t_\delta})$, is modelled as a squared exponential kernel. Suppose we assign a different length scale parameter for each x dimension. K_{t_ρ} will have $d + 1$ hyperparameters, K_{t_f} will have two hyperparameters, and K_{t_δ} will have $d + 1$ hyperparameters. As for the linear emulator, we found no improvement in accuracy in practice by using ARD for the HF model. Thus, we have two hyperparameters for each kernel in HF and $d + 1$ hyperparameters for LF. To be explicit, in the HF model, K_{t_ρ} has two hyperparameters, K_{t_f} has two hyperparameters, and K_{t_δ} has two hyperparameters. For $d = 5$, we have six hyperparameters for LF and six for HF models at each k bin.

4.3.1 *Halo Model Interpretation*

The formulation of our multifidelity emulator bears a marked resemblance to the equations which form the basis of HALOFIT (Smith et al. 2003), and are themselves motivated by the halo model (Peacock & Smith 2000; Seljak 2000). This correspondence allows us to provide a physical interpretation of our results. In the halo model, matter clustering is schematically divided into two components: a two-halo term and a one-halo term. The two-halo term arises from correlations between haloes on large scales, while the one-halo term, which has a weaker dependence on cosmology, is sensitive to the density profile inside each halo. We can model this by splitting the non-linear power spectrum

$$P_{NL}(k) = P_{\rm Q}(k) + P_{\rm H}(k). \tag{23}$$

The quasilinear term $P_{\rm Q}(k)$ is a two-halo term, while $P_{\rm H}(k)$ is a one-halo term. The two-halo term can be modelled by the linear theory power spectrum filtered by a window function $W(M, k)$:

$$P_Q(k) = P_{\rm L}(k) \left(\int W(M, k) {\rm d}M \right)^2. \tag{24}$$

The window function depends on the halo mass function and halo bias, encodes how virialization displaces the linear matter field, and tends to unity on large scales.

There is a clear connection between this model and the form of our multifidelity emulator. Equations (12; AR1) and (20; NARGP) move between fidelities via two terms: a scaling factor ρ and an additive factor δ_t. The correlations between fidelities are strong on large scales, and so $\rho \to 1$ as $k \to 0$. ρ is analogous to the quasilinear window function, except that it filters not the linear theory power spectrum $P_{\rm L}$, but the LF N-body model $f_{t-1}(x)$. In the context of the halo model, it extrapolates the existing quasilinear halo filtering to include lower mass haloes not included in the LF simulation.

The additive factor δ_t, which is important on small scales, is analogous to the one-halo term. It models the difference in halo shot noise and internal halo profiles between resolutions. Notice that δ_t, like the one-halo term, depends only weakly on cosmology, as evidenced by it requiring only one length-scale hyperparameter.

5 SAMPLING STRATEGY FOR HF SIMULATIONS

In this section, we will describe how we select the training simulations for our multifidelity emulators. We will first describe the nested structure implemented in multifidelity emulators in Section 5.1. Section 5.2 explains how we find the optimal choice of highfidelity training simulations.

5.1 Nested training sets

The proposed sampling scheme for training and testing is shown in Fig. 3. The corresponding output power spectra are shown in Fig. 4. In Fig. 3, the sampling is done using two different Latin hypercubes:

(i) Training simulations: a Latin hypercube with 50 points. HR points are a subset of LR points.

(ii) Testing simulations: another Latin hypercube with 10 points.

(iii) We use the notation "X LR-Y HR emulator" to represent a multifidelity emulator trained on X number of low-resolution simulations and Y number of high-resolution simulations.

The first hypercube with 50 points ensures that we will have a nested experimental design. The second hypercube is to ensure we will not test on the training simulations during the validation phase. In practice, we found that the emulation accuracy roughly converged with ∼30 LR points.

5.2 Optimizing the loss of LF simulations

For a multifidelity problem, we want to minimize the required HF training simulations to achieve a given accuracy. We search for the optimal subset of LR points to simulate at HR by picking the subset that would minimize the LF training set's single-fidelity emulator errors. In our experiments with two fidelities, $s = 2$, there are $\binom{n_1}{n_2}$ possible combinations for $\boldsymbol{x}_2$, which are input parameters for the HF data, $\mathcal{D}_2 = \{\boldsymbol{x}_2, \boldsymbol{y}_2\}$.

Retraining LF-only emulators on all possible subsets of the LF grid is computationally intensive. For example, selecting two samples out

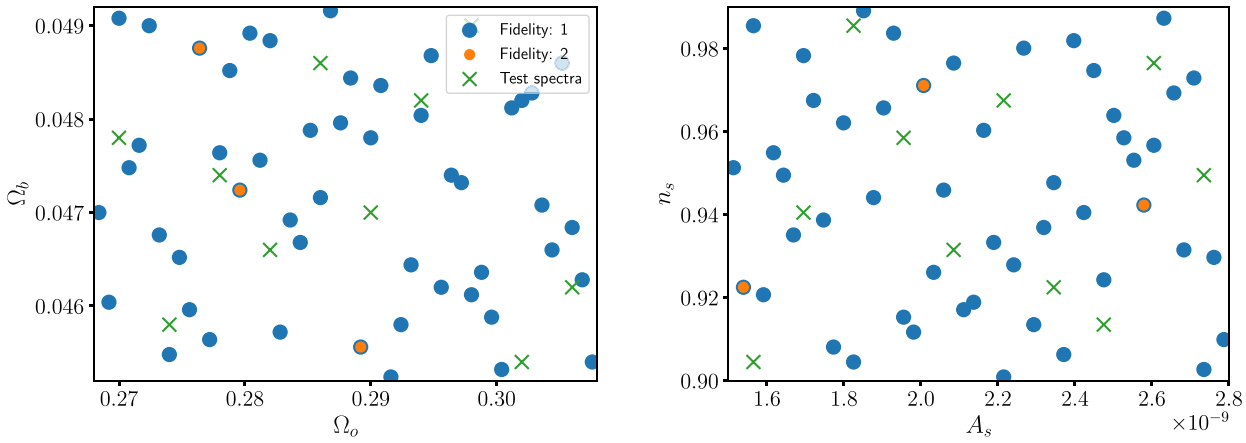

Figure 3. Two 2D cross-sections of the 5D samples of input parameters. The input parameters are designed with a nested structure, $\boldsymbol{x}_1 \subseteq \boldsymbol{x}_2$, between HR and LR. *(Blue):*$\boldsymbol{x}_1$, 50 sampling points in LR. *(Orange):*$\boldsymbol{x}_2$, 3 sampling points in HR. The selection of these three points is chosen by the procedure described in Section 5.2, which minimizes the LR error in the LF-only emulator. *(Green):* 10 points from the HR testing set, which is a different Latin hypercube than $\boldsymbol{x}_1$.

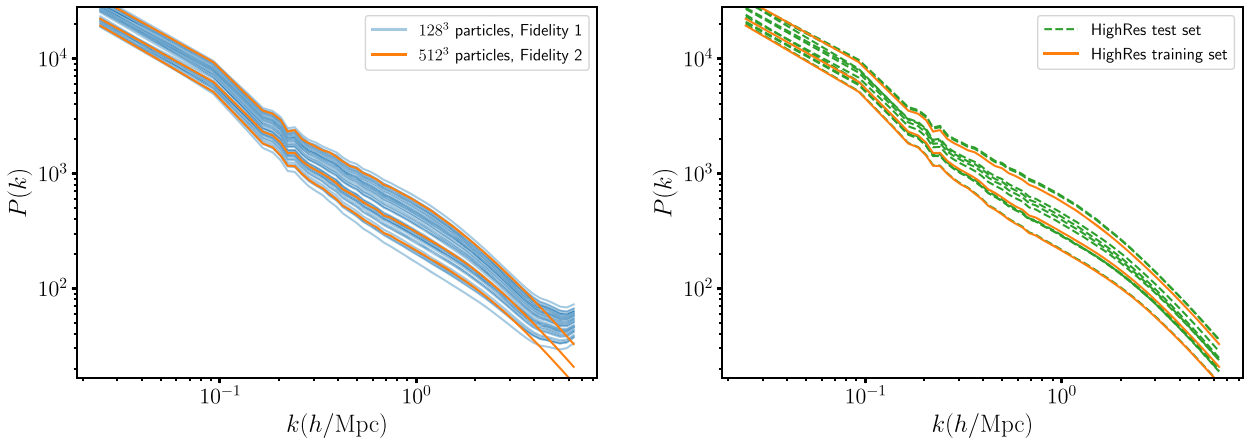

Figure 4. Training (left) and testing (right) data for the multifidelity emulator. *(Left):* 50 LF training simulations (blue) and three HF simulations (orange) used in a 50LR–3HR emulator. A HR is a 512^3 simulation and a LR is a 128^3 simulation. Both HR and LR are in a box with 256 Mpc h^{-1} per side. The 50 LF training simulations are drawn from a 5D Latin hypercube, (h, Ω_0, Ω_b, A_s, and n_s). The three HF simulations are a subset of the LF simulation hypercube. *(Right):* 10 HF test simulations (green dashed) and three HF training simulations (orange).

of 50 points means that we have to train $\binom{50}{2} = 1\,225$ LF emulators. To save computational cost, we employed a greedy optimization strategy. Instead of exploring all possible subsets, we grew the subset one point at a time, fixing the previously chosen points. As a further optimization, we used the same set of kernel hyperparameters for all k bins.

Consider $\mathcal{S}$, a potential $\mathcal{D}_2$ with $\boldsymbol{x}_2 \subset \boldsymbol{x}_1$. We train a LF-only emulator based on equation (8) using the n_2 LF points in $\mathcal{S}$ and get a GP:

$$p(f_* \mid \mathcal{S}, x_*) = \mathcal{N}\left(f_* \mid \mu_*^{(i)}(x_*), \sigma_*^{(i)}(x_*)^2\right), \tag{25}$$

which is the posterior as described in equation (9).

With the trained LF-only emulator in equation (25), we can test this single-fidelity emulator's performance by predicting the rest of the data in the LF Latin hypercube. To evaluate the accuracy, we compute the mean squared error by averaging over the test data:

$$\mathrm{MSE} = \mathbb{E}\left[\left(y_* - \mu_*^{(i)}(x_*)\right)^2\right], \tag{26}$$

where $\{(x_*, y_*)\}$ are the LF data pairs from the rest of the Latin hypercube,

$$\{(x_*, y_*)\} \in \{\mathcal{D}_1 - \mathcal{S}\}. \tag{27}$$

This simply means that we test the single-fidelity emulator on the available data not included in the training subset.

Suppose we repeat the training of single-fidelity emulators until we train all possible subsets in the LF hypercube. We will now have $\binom{n_1}{n_2}$ trained single-fidelity emulators. Each single-fidelity emulator will provide a mean squared error, which is the test error that the emulator generates against the LF hypercube test data. To select the optimally trained emulator, we compute

$$\mathcal{S}^* = \arg\min_{\mathcal{S}^*}\left(\mathbb{E}\left[\left(y_* - \mu_*^{(i)}(x_*)\right)^2\right]\right), \tag{28}$$

where we find the subset $\mathcal{S}^*$ which minimizes the mean squared errors on the test set. We use $\mathcal{S}^*$ as our HF training set $\mathcal{D}_2$ under the

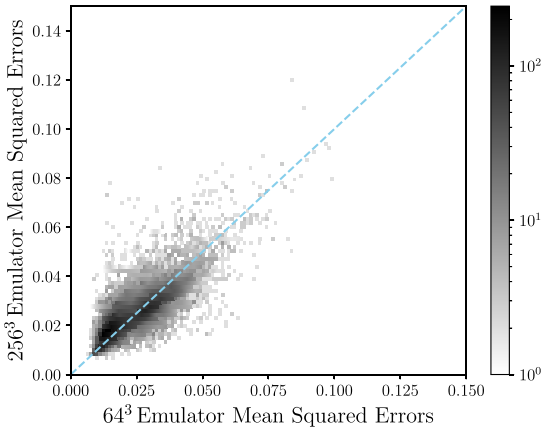

Figure 5. Emulator mean squared errors evaluated from 64^3 emulators and 256^3 emulators. We compute all subsets of three samples from a 50 samples Latin hypercube, $\binom{50}{3} = 19\,600$ subsets in total. Colourbar is in log scale. The blue dashed line represents a perfect linear relationship.

nested experimental design. To be explicit:

$$\boldsymbol{x}_2 = \boldsymbol{x}_{\mathcal{S}^*} \subset \boldsymbol{x}_1, \tag{29}$$

where $\boldsymbol{x}_2$ are the selected HF input points, $\boldsymbol{x}_{\mathcal{S}^*}$ are the input points from the selected subset $\mathcal{S}^*$ (which minimizes the LF emulator mean squared error), and $\boldsymbol{x}_1$ are the LF input points.

This strategy assumes that the effect of a sampling scheme on a LF emulator is the same as that on a corresponding multifidelity emulator. For example, suppose $\Delta\Omega_b$ is crucial for learning how the LF power spectrum y_1 changes for inputs x_1. In that case, we expect that information about $\Delta\Omega_b$ can also effectively change the HF spectrum y_2.

The above assumption could be violated if the power spectra at small scales, which are not included in the LF data, behave very differently from those at large scales. This could happen if the smoothness length scale acts very differently between LF and HF data for a given input dimension. For example, imagine that a parameter, θ, has a small effect on the outcomes of LF simulations, but a large effect on the outcomes of HF simulations.

Fig. 5 shows the mean squared errors computed from 64^3 single-fidelity emulators and 256^3 single-fidelity emulators. First, note that the selection of the training simulations affects the emulator accuracy. Secondly, the LF emulator errors are correlated with their higher fidelity counterparts. This suggests that a LF emulator can serve as a guide for placing HF training simulations. The HR parameter choices used in Section 6 were selected with an earlier version of our model using 64^3 particle simulations. We checked that using either 64^3 or 128^3 for selection gave almost the same emulation accuracy for a non-linear 50 LR-3 HR emulator, though one of the selected samples is different.

In practice, we find the procedure above can prevent us from selecting the HR combination that will give us the worst multifidelity emulation result. Although, we have tested that our procedure works for the matter power spectrum, we would suggest that when emulating a new summary statistic (e.g. the halo mass function), the reader investigates the effectiveness of this method using small test cases. We may in future work and investigate using Bayesian optimization (e.g. Forrester et al. 2007; Lam et al. 2015; Poloczek et al. 2016) to select the optimal HR samples for multifidelity training.

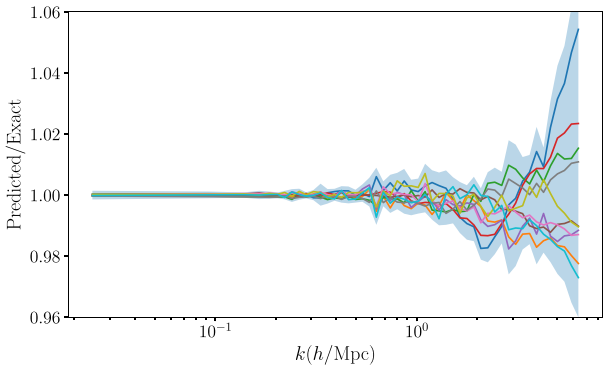

Figure 6. Predicted divided by exact power spectrum from a 50 LR–3 HR emulator using a linear multifidelity method (AR1). Different colours correspond to 10 test simulations spanning a 5D Latin hypercube. The shaded area indicates the worst-case $1-\sigma$ emulator uncertainty. There is one test simulation driving the larger error compared to the non-linear one in Fig. 7.

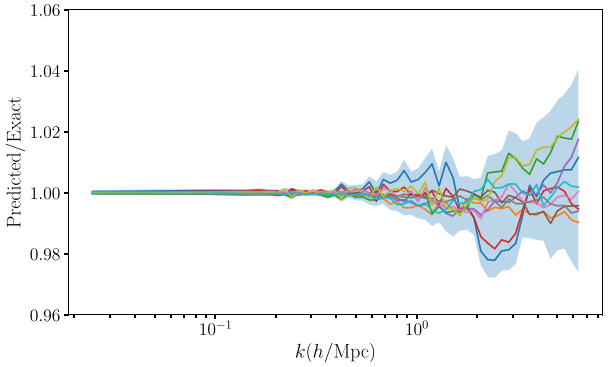

Figure 7. Predicted divided by exact power spectrum from a 50 LR–3 HR emulator using a non-linear multifidelity method (NARGP). Different colours correspond to 10 test simulations spanning a 5D Latin hypercube. The shaded area indicates the worst-case $1-\sigma$ emulator uncertainty. Note that the y-scale in this plot is the same as Fig. 6.

6 RESULTS

This section shows the interpolation accuracy of multifidelity methods and compares our multifidelity emulators to single-fidelity emulators. Section 6.1 compares test set emulator errors for the linear multifidelity emulator (AR1) and non-linear multifidelity emulator (NARGP). Section 6.2 compares a multifidelity emulator to two kinds of single-fidelity emulators: HF only and LF only. We also compare the emulator accuracy as a function of core hours for both multifidelity emulators and single-fidelity emulators.

To test how much a multifidelity emulator can improve with more training simulations, Section 6.3 shows the emulator errors with more LR or HR training simulations. Finally, Section 6.4 checks the performance of the multifidelity method for other emulation settings.

6.1 Comparison of linear and non-linear emulators

Figs 6 and 7 show the predicted power spectrum divided by the exact power spectrum for simulations in the testing set. Both emulators, linear (AR1) and non-linear (NARGP), are trained with 50 LF simulations and 3 HF simulations. We will call these emulators "50 LR–

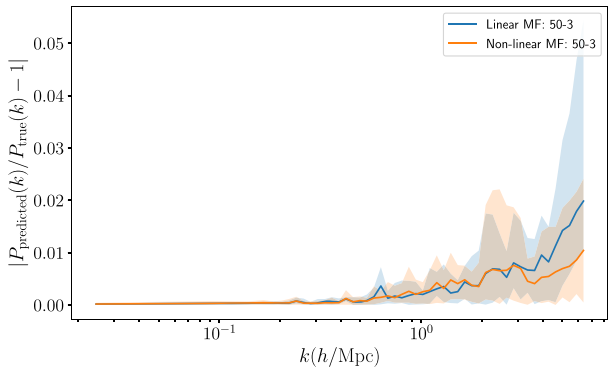

Figure 8. Relative emulator errors from a 50 LR–3 HR emulator using linear multifidelity (blue) and non-linear multifidelity (orange). Solid lines represent the average error from test simulations, $\frac{1}{10}\sum_{i=1}^{10}|\frac{P_{{\rm pred},i}}{P_{\rm true}}-1|$. Shaded areas show the maximum and minimum test errors.

3 HR emulators" for simplicity. A non-linear (linear) multifidelity emulator requires at least 3 (2) HR simulations for training and has $\lesssim 2$ per cent ($\lesssim 5$ per cent) worst-case accuracy per k bin. For a linear multifidelity emulator, the minimum required number of HR simulations is 2, reflecting the lower number of hyperparameters in the kernel.

Fig. 8 shows a comparison between a linear multifidelity emulator and a non-linear multifidelity emulator in relative emulator error. We include linear and non-linear 50 LR–3 HR emulators. We define the relative emulator error:

$$\text{Emulator Error} = \left|\frac{P_{\rm pred}}{P_{\rm true}}-1\right|. \tag{30}$$

$P_{\rm pred}$ is the predicted power spectrum from the multifidelity emulator, and $P_{\rm true}$ is the power spectrum from the HF test simulation.

Fig. 8 shows that the linear 50 LR–3 HR emulator predicts an average error < 1 per cent per k bin for $k \leq 4\ h\,{\rm Mpc}^{-1}$ and < 2 per cent per k bin for $4 < k \leq 6.4\ h\,{\rm Mpc}^{-1}$. The non-linear multifidelity emulator predicts an average error $\lesssim 1$ per cent per k bin, which implies we only need 3 HR to achieve a percent-level accurate emulator using the non-linear multifidelity method. At $k \leq 3\,h\,{\rm Mpc}^{-1}$, both emulators predict mostly the same accuracy, but the non-linear one performs better at smaller scales $k > 3\,h\,{\rm Mpc}^{-1}$.

We found that the non-linear multifidelity emulator outperforms the linear one in all aspects. For simplicity, we will only show the non-linear multifidelity models in the following sections, but we note that a linear multifidelity model is still useful when only two HR simulations are available. We also found that, for the linear model, changing from 50 LR–3 HR emulator to 50 LR–2 HR emulator only slightly degrades the overall accuracy.

6.2 Comparison to single-fidelity emulators

6.2.1 Comparison to HF-only emulators

Fig. 9 shows a comparison between a non-linear 50 LR–3 HR emulator and HF-only emulators. The HF-only emulators are single-fidelity emulators trained solely on HR simulations. The non-linear multifidelity emulator outperforms the single-fidelity emulator with 11 HR at all k modes. It also predicts a worst-case error smaller than the worst-case error from the 11 HR single-fidelity emulator. At $k \leq 2\,h\,{\rm Mpc}^{-1}$, the multifidelity emulator performs much better than the single-fidelity emulators. Since LR simulations can predict

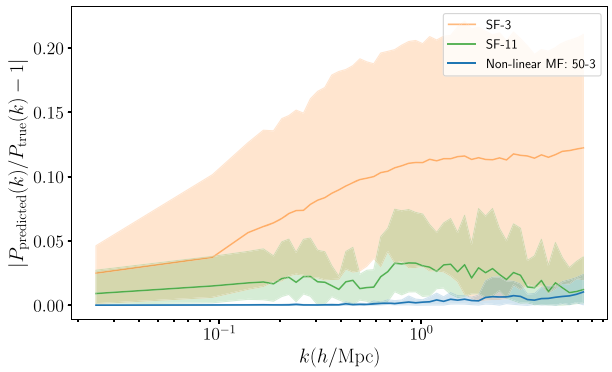

Figure 9. *Non-linear multifidelity emulator (blue)* with 50 LR and 3 HR simulations, compared to *single-fidelity emulators* with 3 HR (orange) and with 11 HR (green). Shaded area indicates the maximum and minimum emulation errors. The computational cost for a 50 LR–3 HR emulator $\simeq$ 9000 core hours while the single-fidelity emulator with 11 HR requires $\simeq$ 25 000 core hours. However, a 50 LR–3 HR emulator still outperforms an 11 HR emulator.

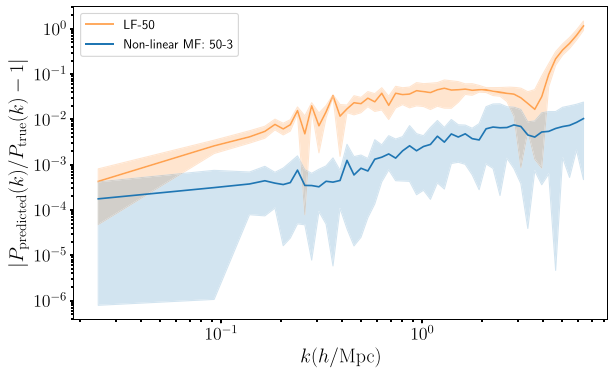

Figure 10. Relative emulator errors between a 50 LF emulator and a non-linear 50 LR–3 HR emulator. Errors are evaluated on 10 HR simulations. Shaded area indicates the maximum and minimum errors. Note that the y-axis is in $\log_{10}$ scale.

accurate power spectrum at large scales $k \leq 2\,h\,{\rm Mpc}^{-1}$, we expect a single-fidelity emulator requires $\sim$50 HR to compete with the 50 LR–3 HR emulator on large scales. A HR is $\simeq$64 times more expensive than a LR, thus the core time for a 50 LR–3 HR emulator is $\simeq$4 HR. The non-linear multifidelity outperforms a single-fidelity 11 HR emulator with $\simeq$3 times lower computational cost.

The error reduction rate is the relative error of a single-fidelity emulator divided by the error of a multifidelity emulator. Both linear and non-linear 50 LR–3 HR emulator show an error reduction rate of $\simeq$100 for $k \leq 0.5\,h\,{\rm Mpc}^{-1}$, $\simeq$100 times better than the single-fidelity counterpart using 3 HR. At smaller scales $k > 3\,h\,{\rm Mpc}^{-1}$, the multifidelity emulators are $\simeq$20 times (non-linear), and $\simeq$10 times (linear) better than their single-fidelity counterpart.

6.2.2 Comparison to LF-only emulators

Fig. 10 shows a single-fidelity emulator trained on 50 LR simulations, compared to a non-linear 50 LR–3 HR emulator. Fig. 10 demonstrates how multifidelity modelling improves the emulator accuracy at each k scale. At $k \lesssim 3\,h\,{\rm Mpc}^{-1}$, multifidelity modelling uses 3 HR to

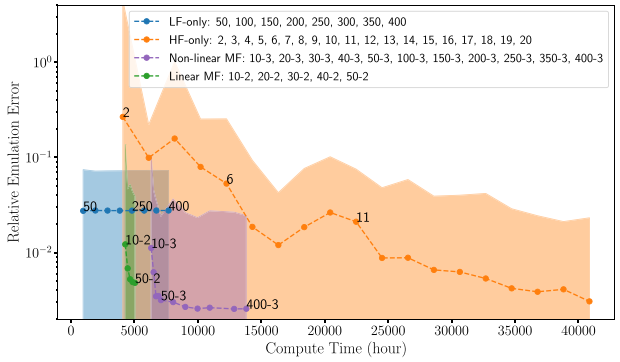

Figure 11. Core hours for running the training simulations versus emulation errors for HF-only emulators (orange) and LF-only emulators (blue), linear multifidelity emulators (AR1) with 2 HR (green), and non-linear multifidelity emulators (NARGP) with 3 HR (purple). The numbers in the labels indicate the number of training simulations used in the emulator. For multifidelity emulators, *X*-*Y*, *X* is the number of low-resolution and *Y* is the number of high-resolution training simulations. The dots show the average errors. The upper shaded areas show the maximum emulator errors among 10 test simulations. The LR samples beyond 100 are drawn from a separate Latin hypercube with 400 samples. For LF-only emulators, we only calculate the relative errors for $k \leq 3$.

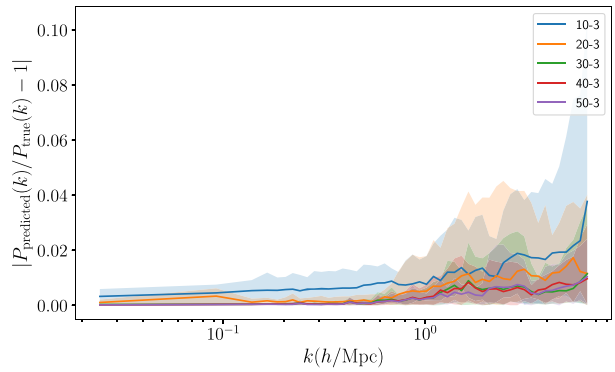

Figure 12. Relative emulator error of non-linear *N* LR–3 HR emulator colour coded with different number of LR training simulations, with $N \in \{10, 20, 30, 40, 50\}$. The same as Fig. 8, solid lines represent the average error from test simulations, $\frac{1}{10}\sum_{i=1}^{10}|\frac{P_{{\rm pred},i}}{P_{\rm true}} - 1|$, and shaded areas show the maximum and minimum test errors.

correct the resolution and reduce the average emulator error from $\lesssim 5$ per cent to ≤ 1 per cent. A LF emulator predicts a biased power spectrum beyond $k = 3\,h\,{\rm Mpc}^{-1}$. However, the multifidelity method can moderately correct the bias and reduce the error to $\lesssim 1$ per cent. Again, the multifidelity technique can use a few HR simulations to calibrate the resolution difference.

6.2.3 Core hours versus emulator errors

Fig. 11 shows the average relative emulator error as a function of core hours for performing the training simulations. The emulator errors shown in Fig. 11 are averaged over all k modes, so each emulator corresponds to a single point in the plot. An ideal emulator will be on the left bottom corner, implying both low cost and high accuracy. The slope of a given emulator in the plot indicates how easily we can improve the emulator with more training data. A steeper (more negative) slope means we can increase the emulator accuracy with a lower cost.

We notice three types of emulators are clustered in separate regions in the plot. The LF-only emulator has the lowest cost and shows no noticeable improvement from increasing training simulations from 50 to 400 LR. The HF-only emulator (HF-only) shows an accuracy improvement with more HR simulations from 3 HR to 11 HR. However, performing one HR requires $\sim$2000 core hours, making the HF-only emulator much more expensive than the other two emulators in the plot.

In Fig. 11, the non-linear multifidelity emulator (NARGP) shows a compute time similar to 3 HR simulations but has better accuracy than the HF-only emulator. It also presents a steeper slope than the HF-only emulator, indicating we can efficiently increase the accuracy using low-cost LR simulations. From 10 LR–3 HR emulator to 50 LR–3 HR emulator, it shows that we can decrease the error from $\sim$0.02 to $\sim$0.003 using an additional $\sim$800 core hours. From 50 LR–3 HR emulator to 400 LR–3 HR emulator, we also see a mild decrease of error but not as steep as 10LR–3HR to 50LR–3HR.

We also include the linear model (AR1) to demonstrate the performance of the multifidelity method when there are only 2 HR available. The linear model also shows a steep improvement slope from 10LR–2HR to 50LR–2HR. However, we notice that the linear model with 2 HR is slightly worse than the non-linear one with 3 HR.

Fig. 11 demonstrates that a multifidelity emulator can provide good accuracy with a much lower cost than HF-only emulators. It also points out that we can efficiently improve the accuracy of a multifidelity emulator using cheap LF simulations.

6.3 Varying the number of training simulations

6.3.1 Effects of more low-resolution training simulations

The benefit of using a multifidelity emulator is that we can improve the emulator accuracy using extra LF simulations. Fig. 12 shows the emulator error colour coded by the number of LR training simulations. With more LR training data, the emulator performance improves at both large and small scales. We only show the non-linear emulator here for simplicity, but we observe a similar trend in the linear emulator. For *N*LR–3HR with $N \in \{10, 20, 30, 40, 50\}$ emulators, the last k bin gives 3.77 per cent, 1.16 per cent, 1.15 per cent, 0.97 per cent, and 1.04 per cent emulator errors, indicating an increase of accuracy with more LR training simulations. Dividing the errors into large and small scales at $k = 1\,h\,{\rm Mpc}^{-1}$, the average emulator errors are 0.65 per cent, 0.22 per cent, 0.10 per cent, 0.09 per cent, and 0.09 per cent for $k \leq 1\,h\,{\rm Mpc}^{-1}$ and 1.60 per cent, 1.04 per cent, 0.60 per cent, 0.61 per cent, and 0.56 per cent for $k > 1\,h\,{\rm Mpc}^{-1}$. The decrease in error is nearly saturated with $\sim$40 LR simulations.

6.3.2 Effects of more high-resolution training simulations

In Fig. 13, we add more HR training simulations to our multifidelity emulator. The 50 LR–N HR emulator with $N \in \{3, 5, 7, 9\}$ shows no improvement in average error with more HR, although the worst case error improves noticeably for the 50 LR–9 HR emulator. One reason may be stochasticity in the training set due to simulation modelling error, which is around 1 per cent, and limits the prediction accuracy. In particular, MP-GADGET simulations with 512^3 particles may not be fully converged on small scales, and this limits the emulator's

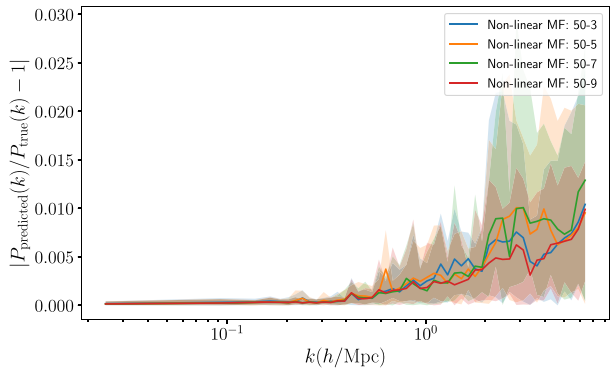

Figure 13. Relative emulator errors from non-linear 50 LR–N HR emulator with $N = 3$ (blue), $N = 5$ (orange), $N = 7$ (green), and $N = 9$ (red) HR training simulations. Solid lines are the average test errors. Shaded areas show the maximum and minimum test errors.

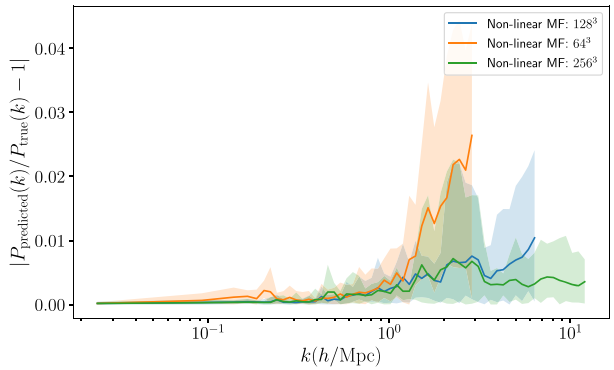

Figure 14. Relative emulator errors for 50 LR–3 HR emulators using different qualities of LR simulations. *(Blue)*: using 128^3 simulations as LF training simulations. *(Orange)*: using 64^3 simulations as LR, which are $\simeq$8 times cheaper than 128^3 simulations. *(Green):* using 256^3 simulations as LR, which are $\simeq$8 times most expensive than 128^3 simulations. Shaded area shows the maximum and minimum errors amongst 10 test simulations.

learning. Another possibility is that prior from 50 LF simulations may be too hard to overcome with only 9 HR simulations.

To improve multifidelity emulator accuracy further, one could build a more complicated model than the one proposed in this paper. The improvement from the linear to the non-linear model shows that different decisions about the scaling factor ρ could better predict the non-linear structure. However, those complicated models will require more HF training simulations. We will leave more complex modelling structures to future work.

6.4 Effect of other emulation parameters

6.4.1 The resolution of LF simulations

We have so far tested multifidelity emulators using 128^3 simulations (LR) as LF and 512^3 simulations (HR) as HF. Fig. 14 shows non-linear 50 LR–3 HR emulators using different mass resolutions, 64^3 and 256^3 simulations, as LF.

A 64^3 simulation is $\simeq$512 times cheaper than a HR but has a smaller maximum k with $\max(k) \simeq 3\,h\,{\rm Mpc}^{-1}$. It produces percent level accuracy for $k \leq 1\,h\,{\rm Mpc}^{-1}$ and has worst-case errors < 5 per cent at

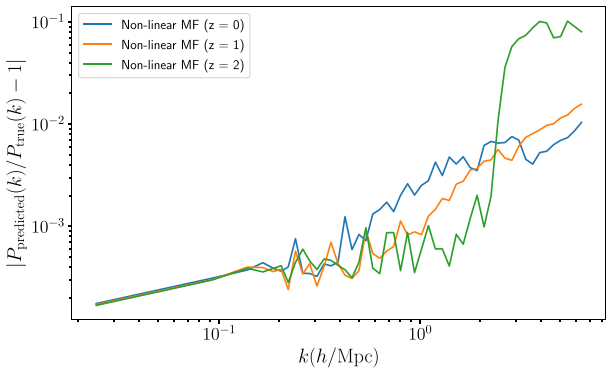

Figure 15. Relative emulator errors for a non-linear emulator at different redshifts, $z \in \{0, 1, 2\}$. Note that the y-axis is in $\log_{10}$ scale. The larger error in the $z = 2$ emulator at $k > 2\,h\,{\rm Mpc}^{-1}$ may be due to a transient near the mean-particle spacing in the LR simulations, see Fig. 16.

small scales $k \geq 1\,h\,{\rm Mpc}^{-1}$. A 256^3 simulation is $\simeq$8 times cheaper than a HR simulation, so the computational cost for a 50 LR–3 HR emulator is $\simeq$9 HR simulations. This emulator mildly outperforms the emulator where LR is 128^3, with an average percent-level emulation until $k \simeq 12\,h\,{\rm Mpc}^{-1}$, but at a substantially increased computational cost.

Fig. 14 demonstrates that one can fuse various qualities of LR with HR simulations to build a multifidelity emulator. Fig. 14 also shows that the multifidelity emulator's accuracy depends on the correlation between LR and HR. A 64^3 simulation is only a rough approximation to its 512^3 counterpart, so the emulator that uses 64^3 simulations as LF is less accurate than the others in Fig. 14.

6.4.2 Emulation at $z = 1$ and $z = 2$

This section examines the performance of a non-linear emulator at higher redshifts, $z = 1$ and $z = 2$. Fig. 15 shows the emulator error of a non-linear 50 LR–3 HR emulator at $z = 0$, 1, 2. The mean error at $z = 1$ is smaller than the $z = 0$ error at $k \leq 2\,h\,{\rm Mpc}^{-1}$ while it is larger for $k > 2\,h\,{\rm Mpc}^{-1}$

. This result shows that it is easier to train the correlation between fidelities at large scales $k \leq 2\,h\,{\rm Mpc}^{-1}$ while harder to train at small scales $k > 2\,h\,{\rm Mpc}^{-1}$. The emulator at $z = 2$ also shows a better performance than $z = 0$ at large scales, $k \leq 2\,h\,{\rm Mpc}^{-1}$, but the error diverges to $\sim$10 per cent on smaller scales, $k > 2\,h\,{\rm Mpc}^{-1}$. The improved performance on large scales may be because at higher redshifts the matter power spectrum is closer to linear theory and so the correlation between fidelities is easier to learn.

Fig. 16 shows the matter power spectrum at $z = 2$, with the same cosmological parameters as Fig. 1 and indicates a potential explanation. At $z = 2$, the LF simulation contains a systematic at the scale of the mean inter-particle spacing, related to the initial spacing of particles on a regular grid. This systematic is a transient and disappears by $z = 0$. However, at redshifts, where it is present, it implies that the LF simulations contain very little cosmological information on scales near their mean interparticle spacing, $k \simeq 3\,h\,{\rm Mpc}^{-1}$ and thus cannot significantly improve the emulation accuracy. It may be possible to improve performance at high redshift with the use of other pre-initial conditions such as a Lagrangian glass (White 1994).

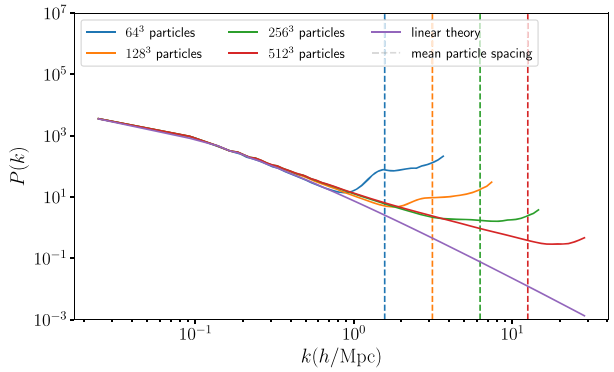

Figure 16. The matter power spectrum at $z = 2$, output by MP-GADGET with different mass resolutions. The vertical dash lines indicated the mean particle spacing k_{spacing} for a given mass resolution. *(Blue):* the matter power spectrum from dark-matter-only MP-GADGET simulation with $N_{\mathrm{ptl,\,side}} = 64$. *(Orange):* the matter power spectrum from MP-GADGET with $N_{\mathrm{ptl,\,side}} = 128$. *(Green):* the matter power spectrum from MP-GADGET with $N_{\mathrm{ptl,\,side}} = 256$. *(Red):* the matter power spectrum from MP-GADGET with $N_{\mathrm{ptl,\,side}} = 512$. *(Purple):* linear theory power spectrum.

7 RUNTIME

We ran our simulations using MP-GADGET on UCR's High Performance Computing Centre and the Texas Advanced Computing Centre. The standard computational setup was 256 MPI tasks per simulation for both HR (512^3 dark matter particles) and LR (128^3 dark matter particles). The runtime was ∼20 core hours for LR and ∼2 000 core hours for HR, with a fixed boxsize 256 Mpc h^{-1}. The computational time for a 64^3 simulation was ∼1.5 core hours with 64 MPI tasks and ∼280 core hours for a 256^3 simulation with 256 MPI tasks.

The computational cost for training a non-linear 50 LR–3 HR emulator (NARGP) was ≃0.5 h and ≃1.6 h for a linear 50 LR–3 HR emulator (AR1) on a single core. For a single-fidelity emulator, it was ≃2 min on one core. The compute time could be further improved by parallelizing the hyperparameter optimization for each k bin. The compute time for optimizing the choice of HR using LF emulators was ∼3 h for selecting 3 HR (on one core). The run time was ≃12 s for evaluating 10 test simulations.

8 CONCLUSIONS

We have presented multifidelity emulators for the matter power spectrum. Multifidelity methods fuse together N-body simulations from different mass resolutions to improve interpolation accuracy. Multifidelity emulators use many LF simulations to learn the power spectrum's dependence on cosmology, correcting for their low resolution by adding a few HF simulations. The result is equivalent in accuracy to a single-fidelity emulator performed entirely with much more costly HF simulations. A multifidelity emulator's physical motivation can be understood using the halo model: LF simulations capture the two-halo term at large scales, while a few HF simulations are used to learn the (almost cosmology independent) one-halo term at small scales.

We have also proposed a new sampling strategy which uses LF simulations as a prior to place HF training simulations. We choose our HF training samples by optimizing the LF emulator's error. In this way, the input parameters at which to run HR simulations can be optimized without knowledge of the HR output. We showed that single-fidelity emulator errors are correlated between different fidelities, indicating that a lower fidelity emulator can serve as a good prior for picking HR simulation points.

Our best multifidelity emulator achieved percent level accuracy using only 3 HR simulations and 50 LR simulations, with a total computational cost ≲4 HR simulations. We showed it outperforms a single-fidelity emulator with 11 HR simulations. We expect that a single-fidelity emulator would require ∼50 HR simulations to compete with the multifidelity one at large scales, $k \leq 2\,h\,\mathrm{Mpc}^{-1}$.

In this paper, we used 128^3 simulations as our LF training sample and 512^3 simulations as HF, with a fixed 256 Mpc h^{-1} box. However, Fig. 14 indicates that our method still has a good performance when extended to other resolutions. We tested our emulator with a series of 10 HR simulations in a Latin hypercube. Two types of multifidelity emulators, linear (AR1) and non-linear (NARGP), are used. We showed that both emulators perform similarly at large scales, while the non-linear one has a better accuracy at small scales.

We focused on $z = 0$, but also investigated higher redshifts. Higher redshift power spectra behave more linearly than at $z = 0$, so it is easier to learn the large-scale correlation between fidelities. However, the LF power spectra are less reliable beyond the mean particle spacing at higher redshifts, inducing some difficulty modelling small scales with $k > 2\,h\,\mathrm{Mpc}^{-1}$.

Our multifidelity emulators could provide percent-level predictions for future space- and ground-based surveys at a minimum computational cost. All current emulators are single-fidelity, training only on expensive HF simulations. A single-fidelity emulator requires at least ∼40 simulations to give percent-level accuracy in a ΛCDM Universe. For example, Heitmann et al. (2009) use 37 simulations to emulate a 5D ΛCDM model. Euclid Collaboration (2020) use ∼200 HF simulations (3000^3 dark matter particles) to achieve the upcoming Euclid mission's desired accuracy in an 8D parameter space.

Our multifidelity methods can also be used to improve the existing single-fidelity emulators. For example, suppose we have run 50 high-resolution simulations to build an emulator. We can perform three additional super high-resolution simulations and combine them to build a super-resolution multifidelity emulator. The choice of these three simulations could be selected via the optimization strategy proposed in this paper. Instead of performing super high-resolution simulations, one could use GAN techniques (see Li et al. 2020) to generate super-resolution simulations and combine them with a multifidelity emulator.

Besides increasing the resolution, multifidelity methods could also be used to decrease the emulation uncertainty of an existing emulator by extending it with many low-resolution simulations. This indicates a low-cost way to enhance current emulators. Multifidelity emulators may make possible efficient expansion of the prior parameter volume. Since HF simulations are only used to calibrate the resolution, they might not need to span the whole parameter space, implying we can expand the sampling range of an existing emulator by extending the LF sampling range. We will leave this technique to future work.

In this work, we have tested our multifidelity emulators with 512^3 resolution and a relatively small box 256 Mpc h^{-1}. In future, we will apply the framework developed here to create a production quality emulator using higher particle load simulations (e.g. 2048^3 particles) in larger boxes. Other summary statistics, including the halo mass function and the cosmic shear power spectrum, could also be emulated using the same framework.

The multifidelity framework may also be extended to hydrodynamical simulations, which are much more costly than their dark matter-only counterparts. No production hydrodynamical emulators

including galaxy formation effects, such as AGN feedback yet exist.[13] However, AGN feedback significantly affects the matter power spectrum at $k > 0.1\,h\,\mathrm{Mpc}^{-1}$ (van Daalen et al. 2011) and pressure forces can affect the power spectrum at $k \sim 10\,h\,\mathrm{Mpc}^{-1}$ (White 2004). Thus practical exploitation of the small-scale information from future surveys will require the development of hydrodynamical emulators. By decreasing the computational cost of an emulator by a factor of ≈3 and still outperforming a single-fidelity emulator, the work presented here makes emulation development substantially more practical.

SOFTWARE

We used the GPY (GPy 2012) package for Gaussian processes. For multifidelity kernels, we moderately modified the multifidelity submodule from EMUKIT (Paleyes et al. 2019).[14] We used the MP-GADGET (Feng et al. 2018a) software for simulations.[15] We generated customized dark matter-only simulations using Latin hypercubes a modified version of SIMULATIONRUNNER.[16]

ACKNOWLEDGEMENTS

We thank the referee for providing insightful suggestions and comments. We thank Martin Fernandez, Phoebe Upton Sanderbeck, Mahdi Qezlou, and Shan-Chang Lin for valuable help and discussions on this project. We thank Cosmology from Home 2020 for providing a valuable place for discussing simulation-based inference during the pandemic. MFH acknowledges funding from a NASA Future Investigators in NASA Earth and Space Science and Technology (FINESST) grant. SB was supported by NSF grant AST-1817256. Computing resources were provided by NSF XSEDE allocation AST180058.

DATA AVAILABILITY

The code to reproduce a 50 LR–3 HR emulator is available at https://github.com/jibanCat/matter_multi_fidelity_emu alongside the power spectrum data.

REFERENCES

Abbott T. M. C. et al., 2020, Phys. Rev. D, 102, 023509
Agarwal S., Abdalla F. B., Feldman H. A., Lahav O., Thomas S. A., 2014, MNRAS, 439, 2102
Amendola L. et al., 2018, Living Rev. Relativ., 21, 2
Angulo R. E., Pontzen A., 2016, MNRAS, 462, L1
Aricò G., Angulo R. E., Contreras S., Ondaro-Mallea L., Pellejero-Ibañez M., Zennaro M., 2020, MNRAS, 506, 4070
Barnes J., Hut P., 1986, Nature, 324, 446
Bird S., Rogers K. K., Peiris H. V., Verde L., Font-Ribera A., Pontzen A., 2019, J. Cosmol. Astropart. Phys., 2019, 050
Bocquet S., Heitmann K., Habib S., Lawrence E., Uram T., Frontiere N., Pope A., Finkel H., 2020, ApJ, 901, 5
Bonilla E. V., Chai K., Williams C., 2008, Advances in Neural Information Processing Systems. NIPS, Curran Associates, Inc., 20

[13] Villaescusa-Navarro et al. (2020) has a neural net emulator trained with 4233 (magneto-)hydrodynamical simulations in a relatively small box, 25 Mpc h^{-1}. Aricò et al. (2020) has an hydro-emulator using baryonification methods for BACCO simulations.
[14] https://github.com/EmuKit/emukit
[15] https://github.com/MP-Gadget/MP-Gadget
[16] https://github.com/sbird/SimulationRunner

Caldwell R. R., Kamionkowski M., 2009, Ann. Rev. Nucl. Part. Sci., 59, 397
Chartier N., Wandelt B., Akrami Y., Villaescusa-Navarro F., 2020, MNRAS, 503, 1897
Couchman H. M. P., Thomas P. A., Pearce F. R., 1995, ApJ, 452, 797, preprint (arXiv:astro-ph/9409058)
Cutajar K., Pullin M., Damianou A., Lawrence N., González J., 2019, Deep Gaussian Processes for Multi-fidelity Modeling, preprint (arXiv:1903.07320)
Damianou A., Lawrence N., 2013, in Carvalho C. M., Ravikumar P., eds, Proceedings of Machine Learning Research Vol. 31, Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics. PMLR, Scottsdale, Arizona, USA, p. 207
Davies C. T., Cautun M., Giblin B., Li B., Harnois-Déraps J., Cai Y.-C., 2020, MNRAS, 507, 2267
Dehnen W., 2002, J. Comput. Phys., 179, 27
DESI Collaboration, 2016, The DESI Experiment Part I: Science,Targeting, and Survey Design, United States, preprint (arXiv:1611.00036)
Euclid Collaboration et al., 2020, MNRAS, 505, 2840
Feng J. L., 2010, ARA&A, 48, 495
Feng Y., Bird S., Anderson L., Font-Ribera A., Pedersen C., 2018a, MP-Gadget/MP-Gadget: A tag for getting a DOI.
Forrester A. I., Sóbester A., Keane A. J., 2007, Proc. R. Soc. A., 463, 3251
Frazier P. I., 2018, A Tutorial on Bayesian OptimizationShow affiliations, preprint (arXiv:1807.02811)
Giblin B., Cataneo M., Moews B., Heymans C., 2019, MNRAS, 490, 4826
GPy since, 2012, GPy: A Gaussian process framework in python. http://github.com/SheffieldML/GPy
Greengard L., Rokhlin V., 1987, J. Comput. Phys., 73, 325
Habib S., Heitmann K., Higdon D., Nakhleh C., Williams B., 2007, Phys. Rev. D, 76, 083503
Harnois-Déraps J., Giblin B., Joachimi B., 2019, A&A, 631, A160
Heitmann K., Higdon D., Nakhleh C., Habib S., 2006, ApJ, 646, L1
Heitmann K., Higdon D., White M., Habib S., Williams B. J., Lawrence E., Wagner C., 2009, ApJ, 705, 156
Heitmann K., Lawrence E., Kwan J., Habib S., Higdon D., 2014, ApJ, 780, 111
Heitmann K. et al., 2016, ApJ, 820, 108
Hinshaw G. et al., 2013, ApJS, 208, 19
Hockney R. W., Eastwood J. W., 1988, Computer Simulation using Particles. Taylor & Francis, Inc., USA
Huang D., Allen T. T., Notz W. I., Miller R. A., 2006, Struct. Multidisc. Optim., 32, 369
Kennedy M., O'Hagan A., 2000, Biometrika, 87, 1
Kern N. S., Liu A., Parsons A. R., Mesinger A., Greig B., 2017, ApJ, 848, 23
Kodi Ramanah D., Charnock T., Villaescusa-Navarro F., Wandelt B. D., 2020, MNRAS, 495, 4227
Kwan J., Bhattacharya S., Heitmann K., Habib S., 2013, ApJ, 768, 123
Kwan J., Heitmann K., Habib S., Padmanabhan N., Lawrence E., Finkel H., Frontiere N., Pope A., 2015, ApJ, 810, 35
Lam R., Allaire D., Willcox K. E., 2015, 56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference. the American Institute of Aeronautics and Astronautics, Inc., Kissimmee, Florida, USA
Lawrence E., Heitmann K., White M., Higdon D., Wagner C., Habib S., Williams B., 2010, ApJ, 713, 1322
Lawrence E. et al., 2017, ApJ, 847, 50
Leclercq F., 2018, Phys. Rev. D, 98, 063511
Lesgourgues J., 2011, preprint (arXiv:1104.2932)
Li Y., Ni Y., Croft R. A. C., Di Matteo T., Bird S., Feng Y., 2020, Proceedings of the National Academy of Science, 118, 2022038118
Liu J., Petri A., Haiman Z., Hui L., Kratochvil J. M., May M., 2015, Phys. Rev. D, 91, 063507
Lukić Z., Stark C. W., Nugent P., White M., Meiksin A. A., Almgren A., 2015, MNRAS, 446, 3697
McClintock T. et al., 2019, preprint (arXiv:1907.13167)
McLeod M., Osborne M. A., Roberts S. J., 2017, preprint (arXiv:1703.04335)
Paleyes A., Pullin M., Mahsereci M., Lawrence N., González J., 2019, in Second Workshop on Machine Learning and the Physical Sciences, NIPS.

Peacock J. A., Smith R. E., 2000, MNRAS, 318, 1144, preprint (astro-ph/0005010)
Pedersen C., Font-Ribera A., Rogers K. K., McDonald P., Peiris H. V., Pontzen A., Slosar A., 2020, J. Cosmol. Astropart. Phys., 2021, 033
Peherstorfer B., Willcox K., Gunzburger M., 2018, SIAM Rev., 60, 550
Pellejero-Ibañez M., Angulo R. E., Aricó G., Zennaro M., Contreras S., Stücker J., 2020, MNRAS, 499, 5257
Perdikaris P., Raissi M., Damianou A., Lawrence N. D., Karniadakis G. E., 2017, Proc. R. Soc. A., 473, 20160751
Poloczek M., Wang J., Frazier P. I., 2016, Advances in Neural Information Processing Systems, Curran Associates, Inc., Multi-Information Source Optimization, 30
Ramachandra N., Valogiannis G., Ishak M., Heitmann K., 2020, Phys. Rev. D, 103, 123525
Rasmussen C. E., Williams C. K. I., 2005, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, Cambridge, MA, USA
Richardson L. F., 1911, Phil. Trans. R. Soc., 210, 307
Rogers K. K., Peiris H. V., Pontzen A., Bird S., Verde L., Font-Ribera A., 2019, J. Cosmol. Astropart. Phys., 2019, 031
Schneider A. et al., 2016, J. Cosmol. Astropart. Phys., 2016, 047
Schneider A., Stoira N., Refregier A., Weiss A. J., Knabenhans M., Stadel J., Teyssier R., 2020, J. Cosmol. Astropart. Phys., 2020, 019
Seljak U., 2000, MNRAS, 318, 203
Smith R. E. et al., 2003, MNRAS, 341, 1311,
Spergel D. et al., 2013, preprint (arXiv:1305.5422)
Springel V., Hernquist L., 2003, MNRAS, 339, 289
Takhtaganov T., Lukic Z., Mueller J., Morozov D., 2019, ApJ, 906, 74
Tyson J. A., 2002, in Tyson J. A., Wolff S., eds, SPIE Conf. Ser. Vol. 4836, Survey and Other Telescope Technologies and Discoveries. SPIE, Bellingham, p. 10
van Daalen M. P., Schaye J., Booth C. M., Dalla Vecchia C., 2011, MNRAS, 415, 3649
Villaescusa-Navarro F. et al., 2020, ApJ, 915, 71
White S. D. M., 1994, preprint (astro-ph/9410043)
White M., 2004, Astropart. Phys., 22, 211
Wong Y. Y. Y., 2011, Ann. Rev. Nucl. Part. Sci., 61, 69
Zel'Dovich Y. B., 1970, A&A, 500, 13
Zhai Z. et al., 2019, ApJ, 874, 95

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