

RECEIVED: January 9, 2025

REVISED: May 27, 2025

ACCEPTED: August 8, 2025

PUBLISHED: September 16, 2025

Effort.jl: a fast and differentiable emulator for the Effective Field Theory of the Large Scale Structure of the Universe

Marco Bonici^{a,b,c,*} Guido D'Amico^{d,e} Julien Bel^f and Carmelita Carbone^{c,g}

^a Waterloo Centre for Astrophysics, University of Waterloo,
Waterloo, ON N2L 3G1, Canada

^b Department of Physics and Astronomy, University of Waterloo,
200 University Ave W, Waterloo, ON N2L 3G1, Canada

^c INAF Istituto di Astrofisica Spaziale e Fisica cosmica di Milano,
Via Alfonso Corti 12, I-20133 Milano, Italy

^d Dipartimento di Scienze Matematiche, Fisiche e Informatiche, Università di Parma,
Viale delle Scienze 7/A, I-43124 Parma, Italy

^e INFN Gruppo Collegato di Parma, Viale delle Scienze 7/A, I-43124 Parma, Italy

^f Aix Marseille Univ, Université de Toulon, CNRS, CPT, Marseille, France

^g ICSC — Centro Nazionale di Ricerca in High Performance Computing, Big Data
e Quantum Computing, Via Magnanelli 2, Bologna, Italy

E-mail: mbonici@uwaterloo.ca, guido.damico@unipr.it,
julien.bel@univ-amu.fr, carmelita.carbone@inaf.it

ABSTRACT: We present the official release of the Effective Field theORy surrogaTe (Effort.jl), a novel and efficient emulator designed for the Effective Field Theory of Large-Scale Structure (EFTofLSS). This tool combines state-of-the-art numerical methods and clever preprocessing strategies to achieve exceptional computational performance without sacrificing accuracy. To validate the emulator reliability, we compare Bayesian posteriors sampled using Effort.jl via Hamiltonian MonteCarlo methods to the ones sampled using the widely-used pybird code, via the Metropolis-Hastings sampler. On a large-volume set of simulations, and on the BOSS dataset, the comparison confirms excellent agreement, with deviations compatible with MonteCarlo noise. Looking ahead, Effort.jl is poised to analyze next-generation cosmological datasets and to support joint analyses with complementary tools.

KEYWORDS: cosmological parameters from LSS, galaxy clustering, Machine learning, Statistical sampling techniques

ARXIV EPRINT: [2501.04639](https://arxiv.org/abs/2501.04639)

*Corresponding author.

Contents

1	Introduction	1
2	The <code>Effort.jl</code> emulator	3
2.1	Neural network architecture and core functionalities	3
2.2	Bias treatment and emulator structure	4
2.3	Preprocessing strategy	5
2.4	Observational effects	6
3	Cosmological inference framework	9
3.1	Probabilistic programming with <code>Turing.jl</code>	9
3.2	The Hamiltonian MonteCarlo sampler	10
3.3	MicroCanonical Hamiltonian Monte Carlo sampler	11
4	Results	11
4.1	Accuracy checks and preprocessing impact	11
4.2	Application to the PT-challenge simulations and BOSS	14
5	Conclusions	17
A	The galaxy power spectrum in the EFTofLSS	21
B	CLASS and pybird settings	22
C	Fixed redshift emulator	23

1 Introduction

In the era of precision cosmology, the analysis of Large Scale Structure (LSS) surveys play a pivotal role in understanding the universe’s contents and dynamics, from its birth to late-time accelerated expansion. Forthcoming datasets from collaborations such as the Dark Energy Spectroscopic Instrument (DESI) [1–3] and Euclid [4] promise to probe the universe’s LSS with unprecedented accuracy. These datasets present opportunities to explore both the standard cosmological model and its potential extensions with unparalleled precision. However, they also impose new computational demands because of the increased dimensionality of the posterior distribution to explore, requiring tools that can efficiently handle the scale and complexity of these analyses.

One promising technique that has emerged to address these computational challenges is emulation. An emulator is a surrogate model that can approximate the outputs of computationally expensive models, but at a significantly lower computational cost. This approach offers a twofold benefit: it accelerates the computation and allows for the application of more advanced optimization and sampling techniques, in particular gradient-based methods.

In recent years, emulators have proven to be highly successful in domains with computationally demanding theoretical models, such as in cosmology.

After the first seminal papers on this topic [5–7], several emulators have been produced in the literature, emulating the output of Boltzmann solvers such as **CAMB** [8] or **CLASS** [9], with applications ranging from the Cosmic Microwave Background (CMB) [10–14], the linear matter power spectrum [11, 15–19], galaxy power spectrum multipoles [17, 19–22], and the galaxy survey angular power spectrum [23–29].

While the acceleration of likelihood evaluations through emulation marks a significant step forward, it is only one facet of the broader efficiency gains that emulators can bring to cosmological analyses. Emulators enable the use of gradient-based samplers, which are particularly well-suited to high-dimensional and complex parameter spaces. These samplers, such as Hamiltonian Monte Carlo (HMC) [30, 31] and its variants, exploit the differentiability of the model to explore parameter spaces more efficiently than gradient-free methods. As a result, they significantly reduce the number of evaluations required to reach chain convergence, especially for complex target distributions, offering substantial improvements in computational performance.

This capability has led to a growing body of work employing gradient-based samplers in cosmology, with applications to CMB and LSS analyses [12, 14, 25, 32–44]. The increasing adoption of these approaches reflects the potential of gradient-based methods to accelerate inference in challenging, high-dimensional models.

The primary method for computing log-likelihood gradients is known as algorithmic differentiation (AD), or automatic differentiation [45–47]. At its core, AD is built on a straightforward concept: any computer program can be interpreted as a sequence of elementary functions, each with a known analytical derivative.¹ The task of an AD engine is to apply the chain rule to combine these derivatives systematically. While AD is capable of differentiating highly complex programs, it can still benefit from higher-level mathematical insights. When custom, hand-written derivatives are provided, significant speed improvements can be achieved. A classic example is matrix multiplication; although this operation can be broken down into individual sums and products, each of which is differentiable, it is far more efficient to express its derivative as another matrix product, which achieves the same result with fewer computational resources [50].

Several codes have already been developed to efficiently compute quantities relevant for galaxy clustering analyses. Notable examples include **Comet** [20], **Matryoshka** [21], and **EmulateLSS** [51]. These frameworks have been successfully applied to a variety of cosmological datasets, demonstrating their effectiveness in this field. However, while all these frameworks are differentiable, none of them has yet been explicitly used in synergy with gradient-based samplers.

In this context, we introduce **Effort.jl**, a **Julia** [52] package that builds on these previous works with a focus on computational performance and integration with gradient-based algorithms. One of the key strengths of **Effort.jl** is its adaptability: observational effects — such as the Alcock-Paczynski (AP) effect and the survey mask — can be either

¹A possible exception is represented by programs with discrete stochastic behavior; in order to deal with this scenario, tailored approaches have been developed as in [48, 49].

incorporated into the data generated to train the emulator or included a posteriori. While the former approach is computationally more efficient, as no post-processing of the emulator output is required, the latter is more flexible as the output can be adapted without the need to retrain the emulator. Furthermore, great care has been devoted to ensure `Effort.jl` compatibility with AD engines, enabling efficient differentiation throughout the entire workflow. While previous codes have laid the groundwork, `Effort.jl` offers a distinct advantage for analyses centered on gradient-based techniques, providing a robust and flexible toolkit tailored to the evolving needs of modern cosmological research.

Through its interface with the probabilistic programming language (PPL) `Turing.jl`,² `Effort.jl` can be used in conjunction with gradient-based sampling and optimization techniques, making it suitable for computing posteriors in Bayesian analyses as well as profile likelihoods (and best-fit points) for frequentist constraints.

The design of `Effort.jl` prioritizes computational efficiency without sacrificing accuracy, enabling the rapid exploration of parameter spaces for the analysis of big cosmological datasets. To validate the performance of `Effort.jl`, we applied it to the PT-challenge simulations [53] and the BOSS data [54], finding no significant deviations from more traditional pipelines that combine `CLASS` or `CAMB` with an EFTofLSS code.

This paper presents the technical details of `Effort.jl`, the methodologies behind its implementation, and its application to both current and future cosmological surveys, with particular emphasis on the upcoming datasets from DESI and Euclid.

This paper is structured as follows: in section 2, we introduce the core features of `Effort.jl`, detailing its architecture, preprocessing strategies, and the implementation of observational effects. Section 3 describes the probabilistic programming framework and the gradient-based sampling methods employed for cosmological inference. In section 4.1, we validate the performance of `Effort.jl` through applications to the PT-challenge simulations and the BOSS dataset, comparing its results against standard pipelines. Finally, in section 5, we present our conclusions and outline potential future developments, including applications to upcoming surveys and extensions of the emulator framework.

2 The `Effort.jl` emulator

In this section, we describe the core features of the `Effort.jl` emulator, highlighting its architecture, preprocessing strategies, training approach, and the implementation of observational effects. `Effort.jl` enables fast computation of the galaxy power spectrum at 1-loop in the EFTofLSS (we refer the reader to appendix A for a brief review) and seamless integration with AD systems. We discuss its efficient handling of bias parameters, its neural network (NN) architecture, and the methods used to incorporate observational effects, ensuring high precision and computational performance.

2.1 Neural network architecture and core functionalities

`Effort.jl`³ is a software package developed in Julia [55], a high-level, high-performance programming language tailored for technical computing. The choice of Julia allows `Effort.jl`

²<https://turinglang.org/>.

³<https://github.com/CosmologicalEmulators/Effort.jl>.

to achieve exceptional performance, with the ability to compute the power spectrum multipoles in approximately $15 \mu\text{s}$.⁴ Additionally, `Julia`'s support for various AD systems plays a crucial role in enabling efficient sampling, a key feature of the use of `Effort.jl` for cosmological parameter inference.

The key functionalities useful for building emulators, such as commands for the instantiation and execution of trained emulators and postprocessing of the emulators output, are abstracted in an external package, `AbstractCosmologicalEmulators.jl`.⁵ This allows us to build other emulators such as `Capse.jl`, a surrogate model for CMB observables [14], with minimal effort.

The `AbstractCosmologicalEmulators.jl` package integrates two different NN libraries, `SimpleChains.jl`⁶ and `Lux.jl`.⁷ `SimpleChains.jl` offers superior performance on central processing units (CPUs), while `Lux.jl` extends compatibility to graphics processing units (GPUs).

Moreover, `AbstractCosmologicalEmulators.jl`'s methods for emulator instantiation via JSON configuration files eliminates the need for pickling or serialization of trained NNs. This approach ensures robustness, portability, and compatibility across different computational environments.

In addition to the NN architecture, `Effort.jl` handles observational effects such as survey window masking and the Alcock-Paczynski (AP) effect. These effects are implemented using tools from the `Julia` ecosystem, ensuring that all steps are compatible with the AD framework.

Thanks to these design choices, `Effort.jl` is easy to train even using standard hardware such as CPUs, making it particularly efficient and accessible, avoiding the need for large-scale hardware setups typically required for more resource-demanding models.

2.2 Bias treatment and emulator structure

`Effort.jl` is designed to emulate the galaxy power spectrum multipoles at 1-loop in the EFTofLSS. The computation is carried out as follows:

$$P_\ell(k; \theta) = \sum_{i,j} b_i b_j \mathcal{P}_{ij,\ell}(k; \theta) + S_\ell(k). \quad (2.1)$$

Here, $P_\ell(k; \theta)$ represents the power spectrum multipoles as a function of wavenumber k and cosmological parameters and redshift θ , which control the cosmological model. The terms $\mathcal{P}_{ij,\ell}(k; \theta)$ are the individual components of the power spectrum multipoles, b_i are biases and counterterms, $S_\ell(k)$ represents the stochastic terms. By treating biases and counterterms analytically, `Effort.jl` (as `pybird` does) decouples the elements depending on cosmological parameters from the other ones, significantly reducing the dimensionality of the emulated space. Regarding the stochastic part, this is constructed analytically by the code and does not require to be emulated.

Other packages, such as `EmulateLSS` [51], incorporate bias parameters directly into the emulation process, which can lead to the need for more complex NNs. In contrast, `Effort.jl`

⁴All the benchmarks of this paper are performed locally using a single core of an i7-13700H CPU.

⁵<https://github.com/CosmologicalEmulators/AbstractCosmologicalEmulators.jl>.

⁶<https://github.com/PumasAI/SimpleChains.jl>.

⁷<https://github.com/LuxDL/Lux.jl>.

follows a strategy similar to **Comet** [20] and **Matryoshka** [21], where bias parameters are handled analytically. This reduces the complexity of the NNs, allowing for faster training and inference, while the small computational overhead of bias inclusion remains negligible.

2.3 Preprocessing strategy

Effort.jl employs a carefully optimized preprocessing pipeline to ensure both accuracy and efficiency in the emulator. The linear matter power spectrum is given by:

$$P_{\text{lin}}(k, z) = A_s k^{-3} \left(\frac{k}{k_0} \right)^{n_s-1} T^2(k, z) D^2(z), \quad (2.2)$$

where A_s and n_s set the initial power spectrum conditions, k_0 is the pivot wavenumber, $T(k, z)$ is the matter transfer function, and $D(z)$ is the scale-independent growth factor.

The P_{11} and counter-term, P_{ct} , components are linear in $P_{\text{lin}}(k, z)$, and therefore scale proportionally with $\mathcal{A}(z) \equiv A_s D^2(z)$, while the loop component is quadratic in $P_{\text{lin}}(k, z)$, making it proportional to $\mathcal{A}^2(z)$, up to the infrared (IR) resummation [56, 57] and the effects of massive neutrinos. To account for this structure, we rescale the P_{11} and P_{ct} terms by $\mathcal{A}(z)$, and the loop term by $\mathcal{A}(z)^2$.

While this rescaling significantly reduces the complexity, it is not perfect due to the IR resummation. To address these residuals, **Effort.jl** combines the analytical rescaling with machine learning, allowing the NNs to learn the remaining non-factorizable dependencies on $A_s D(z)$. This hybrid approach ensures high precision with negligible computational overhead. As shown in [14], this method enhances the emulator’s accuracy.

This rescaling is particularly effective for extended models with additional parameters, which primarily influence the amplitude of the linear matter power spectrum (e.g. modifications to the dark energy equation of state), which primarily influence the amplitude of the linear matter power spectrum. The analytical rescaling efficiently captures their impact on the background evolution.

In **Comet** a similar method is used, known as *evolution mapping*, which exploits parameter degeneracies that primarily affect the amplitude of the linear power spectrum [58]. While **Comet**’s approach is used to reduce the number of input features, **Effort.jl** leverages it primarily to reduce the dynamic range of the output features. This is especially advantageous in cases with scale-dependent growths, such as when accounting for massive neutrinos or considering modified gravity theories [59], where **Effort.jl**’s NN learns the residual effects not captured by the rescaling, offering flexibility in modeling non-linear phenomena. The impact of this approach is assessed and shown in section 4.1.

The growth factor $D(z)$ is thus used to postprocess the output of the NNs, and its computation will be described in the next subsection.

Finally, to enhance numerical stability, **Effort.jl** applies min-max normalization to both input and output features. This ensures consistent scaling across all features, preventing any feature from dominating due to differences in magnitude, which leads to a more stable and efficient NN training process, resulting in improved overall performance [60]. The impact of the normalization has been assessed by retraining the emulator without the normalization procedure. In that case, the emulator performance was greatly worsened, leading

to residuals increased by almost two orders of magnitude, hence showing the importance of the normalization procedure.

2.4 Observational effects

In this subsection, we describe the treatment of key observational effects in `Effort.jl`, including the computation of required quantities such as the growth factor, the inclusion of the Alcock-Paczynski (AP) effect, and the window mask convolution.

2.4.1 Computation of additional quantities

The accurate calculation of background quantities is fundamental for interpreting large-scale structure data. In `Effort.jl`, we compute several key cosmological background quantities: the Hubble factor $H(z)$, the radial comoving distance $r(z)$, the growth factor $D(z)$, and the growth rate $f(z)$. These quantities are essential inputs for the inclusion of the AP effect (and, in the case of $f(z)$, redshift-space distortions), as we will discuss in the next subsection.

There are two main approaches to handling background quantities in cosmological analyses. One approach, adopted by packages such as `Matryoshka` and `CosmoPower` [11], is to emulate the background quantities. Once such an emulator has been trained and validated, it can be effectively used to replace numerical integration routines, speeding up the calculations.

In contrast, `Effort.jl` computes these background quantities dynamically by integrating the relevant equations at runtime. Albeit more time consuming, as the numerical integration is more expensive than using a NN, this approach avoids the need for additional emulators, simplifying the pipeline and reducing dependency on pre-computed models. With a careful implementation, these calculations can be performed efficiently, providing precise results for a little overhead that is amortized when performing the computations for multiple redshifts.

When computing the Hubble factor, `Effort.jl` accounts for contributions from non-relativistic matter, evolving DE, radiation, and massive neutrinos. In particular, for massive neutrinos, we incorporate the transition from relativistic to non-relativistic regimes, following the treatment described in [61, 62]; here we review the most important equations for completeness. The dimensionless Hubble factor $E(z) = H(z)/H_0$ is given by

$$E^2(z) = \Omega_{\gamma,0}(1+z)^4 + \Omega_{c,0}(1+z)^3 + \Omega_{\nu}(z) + \Omega_{\text{DE},0}(1+z)^{3(1+w_0+w_a)} \exp \frac{-3w_a z}{1+z}. \quad (2.3)$$

$\Omega_{\gamma,0}$, $\Omega_{c,0}$, and $\Omega_{\text{DE},0}$ are the abundances of radiation, non-relativistic matter, and DE, respectively, and the DE equation of state is parametrized as $w(a) = w_0 + w_a(1-a)$ [63, 64].

The contribution from massive neutrinos is given by

$$\Omega_{\nu}(a) = \frac{15}{\pi^4} \Gamma_{\nu}^4 \frac{\Omega_{\gamma,0}}{a^4} \sum_{j=1}^{N_{\nu}} \mathcal{F} \left(\frac{m_j a}{k_B T_{\nu,0}} \right), \quad (2.4)$$

where m_j is the mass of each neutrino eigenstate, and $T_{\nu,0}$ is the current temperature of the neutrino background, $\Gamma_{\nu} \equiv T_{\nu,0}/T_{\gamma,0}$ is the neutrino-to-photon temperature ratio today and the function $\mathcal{F}(y)$ is defined as

$$\mathcal{F}(y) \equiv \int_0^{\infty} dx \frac{x^2 \sqrt{x^2 + y^2}}{1 + e^x}. \quad (2.5)$$

The parameter Γ_ν takes into account corrections to the instantaneous decoupling and it is given by

$$\Gamma_\nu^4 = \frac{1}{3} N_{\text{eff}} \left(\frac{4}{11} \right)^{4/3}, \quad (2.6)$$

where we take $N_{\text{eff}} = 3.044$ [65–67].

The comoving distance $r(z)$ is computed as

$$r(z) = c \int_0^z \frac{dz'}{H(z')}. \quad (2.7)$$

We implement two numerical approaches to solve the integral for $r(z)$. First, we employ an adaptive integration Gaussian quadrature scheme, which ensures a high-precision calculation but is more computationally expensive. This method is ideal for cases requiring extremely accurate results. Second, we use a Gauss-Lobatto rule with a fixed number of points (9), which provides sufficient precision for most practical purposes, yielding a relative accuracy better than 10^{-5} compared to the adaptive quadrature scheme.⁸ We further validated our results with `CAMB`, `CLASS`, and `pycc1` [68], finding an agreement up to the fifth digit.

The growth factor $D(a)$ is computed by solving the following second-order differential equation:

$$D''(a) + \left(2 + \frac{E'(a)}{E(a)} \right) D'(a) = \frac{3}{2} \Omega_m(a) D(a), \quad (2.8)$$

where $D' \equiv \frac{dD}{d \ln a}$.

We start solving this equation at high redshift, nominally $z = 138$, deep in the matter domination regime. The appropriate initial conditions in this regime are given by:

$$\begin{aligned} D(z_i) &= a_i \\ D'(z_i) &= a_i, \end{aligned} \quad (2.9)$$

where a_i corresponds to the initial scale factor.

To integrate this equation, we use the Tsitouras algorithm [69], a Runge-Kutta method known for its efficiency and accuracy.

The solver is implemented in the `DifferentialEquations.jl`⁹ package, part of the `Julia ScientificMachineLearning` (SciML) suite, which provides a high-performance environment for solving differential equations [70].

The solver is made fully differentiable, supporting both forward and backward automatic differentiation (AD) systems. This ensures that gradients can be computed accurately and efficiently during optimization processes that require differentiation of the solver output, such as when integrating `Effort.jl` with gradient-based pipelines.

The growth rate $f(z)$ is related to the evolution of the growth factor through

$$f(z) \equiv \frac{d \ln D(z)}{d \ln a}, \quad (2.10)$$

⁸The adaptive Gaussian quadrature scheme is used within our suite of unit tests to verify that the faster Gauss-Lobatto method is accurate enough for typical cosmological applications.

⁹<https://github.com/SciML/DifferentialEquations.jl>.

where a is the scale factor. The solver also retrieves the first derivative of the growth factor, $D'(a)$, which can be used directly to compute $f(z)$. These background quantities are dynamically computed for each cosmological model considered in `Effort.jl`, providing a flexible and precise foundation for modeling galaxy clustering. A comparison between the growth rate computed by `CLASS` and `Effort.jl` shows residual smaller than 0.1%.

2.4.2 Alcock-Paczynski (AP) effect

The Alcock-Paczynski (AP) effect is a geometric distortion that arises when converting redshift space into physical space, since one must assume a reference cosmological model [71]. This effect causes anisotropies in the observed galaxy clustering, affecting the inferred distances along and across the line-of-sight. In this work, we follow the notation used in [72, 73].

To account for the AP effect in `Effort.jl`, we calculate the distortion factors as follows:¹⁰

$$q_{\parallel} = \frac{D_A(z)H(z=0)}{D_A^{\text{ref}}(z)H^{\text{ref}}(z=0)}, \quad q_{\perp} = \frac{H^{\text{ref}}(z)/H^{\text{ref}}(z=0)}{H(z)/H(z=0)} \quad (2.11)$$

Here, q_{\parallel} and q_{\perp} represent the distortions along and transverse to the line-of-sight, respectively. These scaling factors adjust for the differences between the reference and true cosmology, where $D_A(z)$ is the angular diameter distance, and $H(z)$ is the Hubble factor.

The transformation of the power spectrum multipoles due to the AP effect is given by:

$$P_{\ell}(k^{\text{ref}}) = \frac{2\ell+1}{2q_{\parallel}q_{\perp}^2} \int_{-1}^1 d\mu^{\text{ref}} P\left(k\left(k^{\text{ref}}, \mu^{\text{ref}}\right), \mu\left(\mu^{\text{ref}}\right)\right) \mathcal{L}_{\ell}\left(\mu^{\text{ref}}\right). \quad (2.12)$$

This equation expresses the power spectrum multipoles $P_{\ell}(k)$, where μ^{ref} is the cosine of the angle between the wavevector k and the line-of-sight in the reference cosmology. The function $\mathcal{L}_{\ell}(\mu^{\text{ref}})$ is the Legendre polynomial of order ℓ , and the transformation adjusts the power spectrum to account for differences between the true and reference cosmology.

The relations between the wavevector k and angle μ in the true and reference cosmologies are given by [74–76]:

$$k = \frac{k^{\text{ref}}}{q_{\perp}} \left[1 + \left(\mu^{\text{ref}}\right)^2 \left(\frac{1}{F^2} - 1\right) \right]^{1/2}, \quad \mu = \frac{\mu^{\text{ref}}}{F} \left[1 + \left(\mu^{\text{ref}}\right)^2 \left(\frac{1}{F^2} - 1\right) \right]^{-1/2} \quad (2.13)$$

Here, $F = q_{\parallel}/q_{\perp}$ is the anisotropy factor, capturing the relative stretching between the line-of-sight and transverse directions due to cosmological distortions. These transformations ensure that the clustering signal is correctly interpreted in terms of the true cosmological model.

We implemented two methods to integrate the AP effect. We use an adaptive Gaussian quadrature scheme for high precision, although this approach is more computationally expensive. Alternatively, we use a Gauss-Lobatto quadrature with 5 points, which achieves near floating-point precision with reduced computational cost.

The main challenge arises when incorporating AD. To compute the AP effect, we need to interpolate the emulator’s output onto a new k -grid, whose specific values depend on cosmological parameters. For this, we employ a quadratic spline interpolation, as in `pybird`.

¹⁰These are the appropriate definitions when using wavenumber units of h/Mpc , which we assume throughout.

However, calculating the Jacobians of the splines with respect to their input leads to a sparse Jacobian matrix, which the AD system does not naturally optimize for.

Although the AP calculation itself takes approximately $30\,\mu\text{s}$, backward propagation of gradients through the AP calculation initially took around 100 ms, significantly degrading `Effort.jl`'s performance.

To resolve this, we reimplemented the spline interpolation and wrote custom backward differentiation rules that take advantage of the sparsity pattern. This optimization reduced the time for calculating the Jacobians to approximately $200\,\mu\text{s}$, representing a three-order-of-magnitude improvement.

We validated the correctness of these custom differentiation rules using reverse and forward mode AD systems¹¹ and finite differences derivatives as well. Differences were always smaller than 0.01%, both when giving input that were spanning the cosmological parameter space considered and both when providing randomly generated arrays.

2.4.3 Window mask convolution

The finite size and irregular geometry of surveys introduce an observational window function that modulates the observed clustering signal. Accounting for this requires a multiplication of the theory correlation function with a window mask in real space, or a convolution of the window mask with the theory power spectrum in Fourier space.

In `Effort.jl`, we adopt the approach of [78], implementing the window convolution as an efficient array contraction. The window function, derived from the survey mask, is applied to the theoretical predictions at each step of the MCMC sampling, ensuring that observational effects are consistently incorporated into the analysis.

The convolution is executed using `Tullio.jl` [79], a high-performance framework for array operations. The entire process is completed in a few μs . To further enhance performance, we have written custom differentiation rules, enabling the automatic differentiation (AD) system to efficiently manage the convolution operation. We validated the window-convolution rules using the same procedure applied to the AP-differentiation rules, comparing them against reverse-mode and forward-mode automatic-differentiation systems as well as finite-difference derivative approximations.

By accurately incorporating these observational effects, `Effort.jl` delivers precise predictions for the power spectrum, supporting robust cosmological parameter inference.

3 Cosmological inference framework

3.1 Probabilistic programming with `Turing.jl`

For cosmological parameter inference, we employ a probabilistic programming language (PPL) which is built in `Julia`, namely `Turing.jl` [80]. PPLs allow users to specify probabilistic models that include the necessary priors and likelihoods, streamlining the process of drawing samples from the posterior distribution. This flexibility is particularly useful for complex models, such as those involving cosmological parameters, as it enables integration with advanced samplers and AD frameworks.

¹¹Specifically, we used `Zygote.jl` and `ForwardDiff.jl` [77].

`Turing.jl` provides a straightforward way to define probabilistic models by allowing users to declare their priors and construct the likelihood function. Once the model is defined, `Turing.jl` takes care of deriving the posterior and can utilize various sampling algorithms for inference. A key strength of `Turing.jl` is its compatibility with gradient-based samplers, such as the Hamiltonian Monte Carlo (HMC) and its variants, including the No-U-Turn Sampler (NUTS) and MicroCanonical Langevin Monte Carlo [81].

In this work, we use NUTS and MicroCanonical Langevin Monte Carlo [81] as samplers, all of them accessed through the `Turing.jl` interface. This approach has been successfully employed in various cosmological studies, as demonstrated in [14, 25, 39]. These samplers efficiently explore the posterior distributions of cosmological and nuisance parameters, taking full advantage of the differentiability of `Effort.jl`.

3.2 The Hamiltonian MonteCarlo sampler

The Hamiltonian Monte Carlo (HMC), also known as Hybrid Monte Carlo, is a state-of-the-art method for drawing samples from a probability distribution [31]. It is particularly useful when the distribution is over high-dimensional spaces and/or with complex geometries.

The key idea behind HMC is to introduce momentum variables, p , for each parameter in the model, which is instead represented as a position variable q . A potential energy $V(q)$ is introduced, given as minus the logarithm of the joint-likelihood $\log L$:

$$-\log \mathcal{P}(q) = V(q) \quad H(q, p) = V(q) + U(q, p), \quad (3.1)$$

where the kinetic term is given by:

$$U(q, p) = p^T M^{-1} p. \quad (3.2)$$

These momentum variables, together with the original parameters, form a Hamiltonian system. The system evolves according to Hamilton's equations, which in the vanilla case read:

$$\begin{aligned} \frac{dp}{dt} &= -\frac{\partial V}{\partial q} = -\frac{\partial \log \mathcal{P}}{\partial q} \\ \frac{dq}{dt} &= +\frac{\partial U}{\partial p} = M^{-1} p. \end{aligned} \quad (3.3)$$

In the vanilla HMC, the system evolves deterministically for a fixed amount of time, following a trajectory that is likely to stay in regions of high probability under the target distribution. The trajectory is usually numerically integrated with a symplectic integrator.

The deterministic evolution in HMC allows it to explore the target distribution more efficiently. However, it requires the ability to compute gradients of the log-probability of the target distribution, which can be computationally expensive for complex models.

Vanilla HMC is a powerful sampling method that combines ideas from physics and statistics and can efficiently draw samples from high-dimensional distributions, but requires tuning and may be computationally expensive for complex models. One of the main challenges with the vanilla HMC is the need to carefully choose and tune the step size and the number of steps. If these parameters are not set correctly, the HMC can either miss important regions of the parameter space or waste computational resources by taking too many steps.

The No-U-Turn Sampler (NUTS) is a self-tuning variant of the Hamiltonian Monte Carlo (HMC) method, designed to address some of the shortcomings of the vanilla HMC [30]. The name “No-U-Turn” comes from the sampler’s unique feature of stopping its trajectory once it starts to turn back on itself, hence avoiding unnecessary computations.

NUTS tunes HMC hyper-parameters by adaptively choosing the trajectory length. It starts with a trajectory of length 1, and then doubles the length as long as the trajectory does not make a U-turn. A U-turn is detected when the current position and the proposed new position are moving in opposite directions along the gradient of the log-probability.

This adaptive algorithm ensures that NUTS spends more time exploring new regions of the parameter space and less time retracing its steps. It also removes the need for manual tuning of the trajectory length, making NUTS more user-friendly than vanilla HMC.

However, like HMC, NUTS requires the ability to compute gradients of the log-probability, which can be computationally expensive for complex models. In our case, the computation of the log-likelihood gradient is performed using one of the Julia AD systems.

3.3 MicroCanonical Hamiltonian Monte Carlo sampler

The MicroCanonical Hamiltonian Monte Carlo (MCHMC) sampler [81, 82], is a variation of the traditional HMC method. Unlike standard HMC, which relies on a Metropolis-Hastings correction step to ensure detailed balance, MCHMC operates within the microcanonical ensemble, where the Hamiltonian energy $H(q, p)$ is conserved throughout the entire trajectory. Removing the need for Metropolis corrections, reduces computational overhead and increases sampling efficiency.

MCHMC also employs a variable mass Hamiltonian in eq. (3.2), which adapts to the geometry of the posterior distribution. In regions of higher probability density, the particle moves slower, improving the sampler’s capacity to explore complex and high-curvature posterior distributions.

This method is especially well-suited for cosmological inference tasks, where the parameter space is often large and complex. MCHMC has been demonstrated to outperform traditional HMC methods in terms of sampling efficiency for such high-dimensional problems, providing faster convergence while maintaining accuracy in the estimation of posterior distributions [14, 81, 83].

4 Results

4.1 Accuracy checks and preprocessing impact

Having laid out the structure of `Effort.jl`, we can now discuss a concrete implementation. Given its flexible structure, it is possible to use `Effort.jl` with two different approaches, i.e. including redshift as a free parameter or working at a specific redshift. While the former approach is more flexible, as it enables us to train one generic emulator that can be used in more situations, the latter option naturally leads to a more precise surrogate, as we are removing one of the input parameters. Considering that surveys like DESI and Euclid divide galaxy in only a handful of bins with a specific effective redshift, the second approach is useful in case one needs to increase the accuracy of the surrogate, as was done in [84]. Here

Parameter	Min	Max
z	0.25	2.0
$\ln 10^{10} A_s$	2.5	3.5
n_s	0.8	1.10
H_0 [km/s/Mpc]	50.0	80.0
ω_b	0.02	0.025
ω_c	0.08	0.2
M_ν [eV]	0.0	0.5

Table 1. Parameter bounds for redshift and cosmological parameters.

we focus on an emulator of the former kind, since we want to provide a tool that does not require additional training; in case it was required a higher level of precision, users can employ our flexible emulator for a preliminary analysis and train one at a fixed redshift only in the relevant region of parameters space, in an active learning approach [85]. The training dataset contained 60,000 couples of input parameters and $P_\ell(k)$ s; the linear power spectra were computed using CLASS while the EFT predictions are based on `pybird`. A description of the EFTofLSS as implemented in `pybird` is presented in appendix A, while the code used to compute the elements of the training dataset is reported in appendix B. We input to the NN the redshift and cosmological parameters distributed according to the Latin Hypercube in the parameter range described in table 1. We employ NNs with 5 hidden layers, each of them with 64 neurons. We used a batch size of 128, and trained for 100,000 epochs using an ADAM optimizer with an initial learning rate of 10^{-4} ; when using an 8-core CPU, this process took around one hour of wall-time. We employed a vanilla mean square error loss function for the emulator presented in this section and used in the BOSS analysis, while for the PT-challenge we used a modified loss function which included a $1/k^2$ weight. The dataset has been split in 80% and 20% for, respectively, the training and test sets. To prevent overfitting, we save the weights only when the loss on the testing dataset shows improvements.

Before showing a few applications of trained emulators, we want to gauge the impact of the preprocessing procedure outlined in section 2.3.

The question we aim to answer is: how can we improve the performance of our emulators? The straightforward approach is to use a bigger dataset and/or a bigger NN, according to the approximation theorem [86]. The second approach which we want to investigate here is a rescaling of the output features which depends on the value of the input features; this method has already been applied in [14], where the CMB spectra were rescaled by A_s , leading to a reduction of the residuals by a factor of 3, clearly showing the benefits of this strategy. Here we can make a further step, dividing the output features by $D(z)$ as already outlined in section 2.3.

In order to assess the impact of rescaling strategies on the observables considered here in a quantitative manner, we study the distribution of the percentage residuals in the following scenarios:

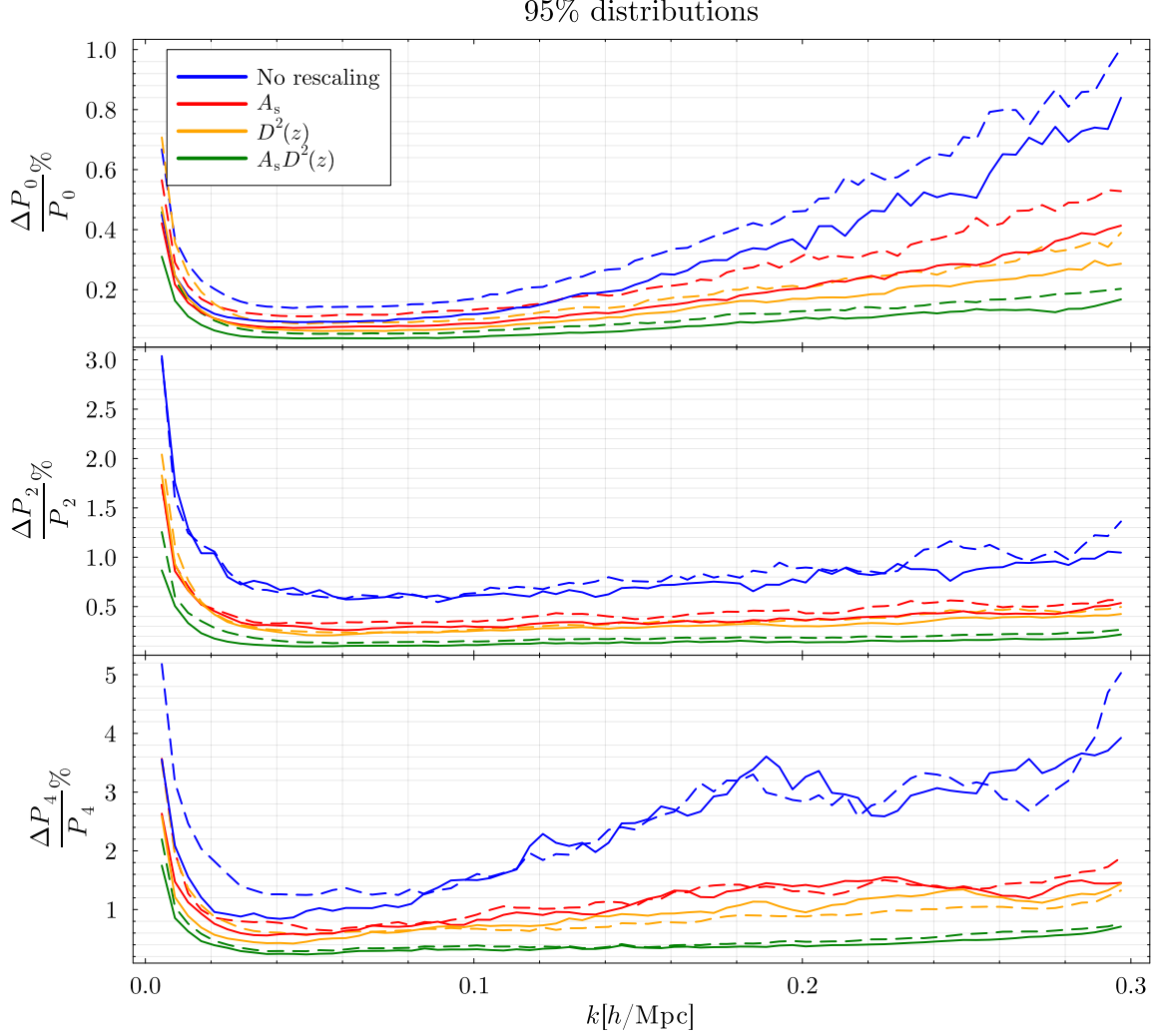


Figure 1. 95% distribution of the percentage errors, averaged over 10 realizations. We show the impact of the different rescaling schemes employed. The dot-dashed and solid lines refers respectively to a training dataset with 20,000 and 60,000 elements. As can be seen, a bigger dataset helps in enhancing the performance, but the rescaling is even more impactful in reducing the residuals. For this test, we fix the EFT parameters, using a combination obtained from the measurements employed in [43].

- No rescaling is employed
- We rescale by A_s
- We rescale by $D^2(z)$
- We rescale by $A_s D^2(z)$

Furthermore, we consider two training datasets, with 20,000 and 60,000 samples. The results of this comparison are shown in figure 1; in order to suppress the variance inherent to such a process, we repeat the procedure 10 times with different random seeds and average the results. As can be seen from the comparison of the dashed and solid lines, which represents

respectively the 20,000 and the 60,000 samples training dataset, getting a bigger dataset helps in improving the accuracy of the emulator. However, we want to emphasize that the rescalings have a much bigger impact. This shows how it is possible to get a high-precision emulator even when using a small NN: in other studies, like [11, 21, 22], the NNs employed had a significantly higher number of neurons.

Finally, it is worth mentioning that our most effective physics-based preprocessing procedure, while lowering the emulation error, relies on the solution of an ODE that, although relatively fast (on the order of $150 \mu\text{s}$), has now become the main bottleneck of our pipeline since `Effort.jl` computes each multipole in around $15 \mu\text{s}$. To circumvent this issue, as already noted in this manuscript, it would be possible to train a NN to emulate $D(z)$ as well. Since this approach has already been successfully applied in the literature, we employed a symbolic regression approach (using `SymbolicRegression.jl` [87]) to emulate the ODE solution by directly seeking a closed-form expression for the growth factor. This strategy offers two main advantages: first, once the symbolic expression is found, it can be ported to any programming language without loss of precision; second, it greatly speeds up the preprocessing step. Indeed, evaluating the symbolic expression takes only 200 ns, offering a dramatic speedup compared to the ODE solution (see also [18, 88–91] for other applications of symbolic regression related to LSS observables).

Moreover, while joint analyses can partially amortize the ODE cost by solving it once for multiple redshifts, it remains desirable to reduce this expense further. We show the residuals for the obtained expression in figure 2; the higher error at low z is due to the higher dynamical range of the growth factor when the redshift approaches 0.

Substituting the ODE solver with the symbolic growth factor cause a modest increase in the residuals, that is almost negligible for the hexadecapole emulator, due to their initial higher residuals. Nonetheless, the symbolic approach proves advantageous in terms of portability, consistency, and computational efficiency. However, for users who do not want to sacrifice accuracy, it is still possible to choose to use the ODE solver in `Effort.jl`.¹²

Supported by our findings, we advocate for physics-based preprocessing, as it helps in improving the accuracy of the emulators by leveraging the domain knowledge of the scientists.

4.2 Application to the PT-challenge simulations and BOSS

In this section we show the application of `Effort.jl` to two different sets of measurements, the PT-challenge and BOSS datasets.

The PT-challenge¹³ was designed to demonstrate the reliability and robustness of the EFTofLSS as a powerful and trustworthy tool for analyzing cosmological datasets. To achieve this, a set of high-precision simulations was generated. This approach provided an ideal testing ground for validating the accuracy and effectiveness of theoretical models like the EFTofLSS.

The simulations consisted of ten boxes with independent random realizations, each comprising $3,072^3$ mass elements within a periodic cube of side length $3,840 h^{-1} \text{ Mpc}$,

¹²We note that, with respect to the first version of this draft, the error increase when using the symbolic surrogate for the growth factor is higher. This is the case as the previously trained emulators had higher residuals and hence the relative error induced by the symbolic emulator was lower.

¹³<https://www2.yukawa.kyoto-u.ac.jp/takahiro.nishimichi/data/PTchallenge/>.

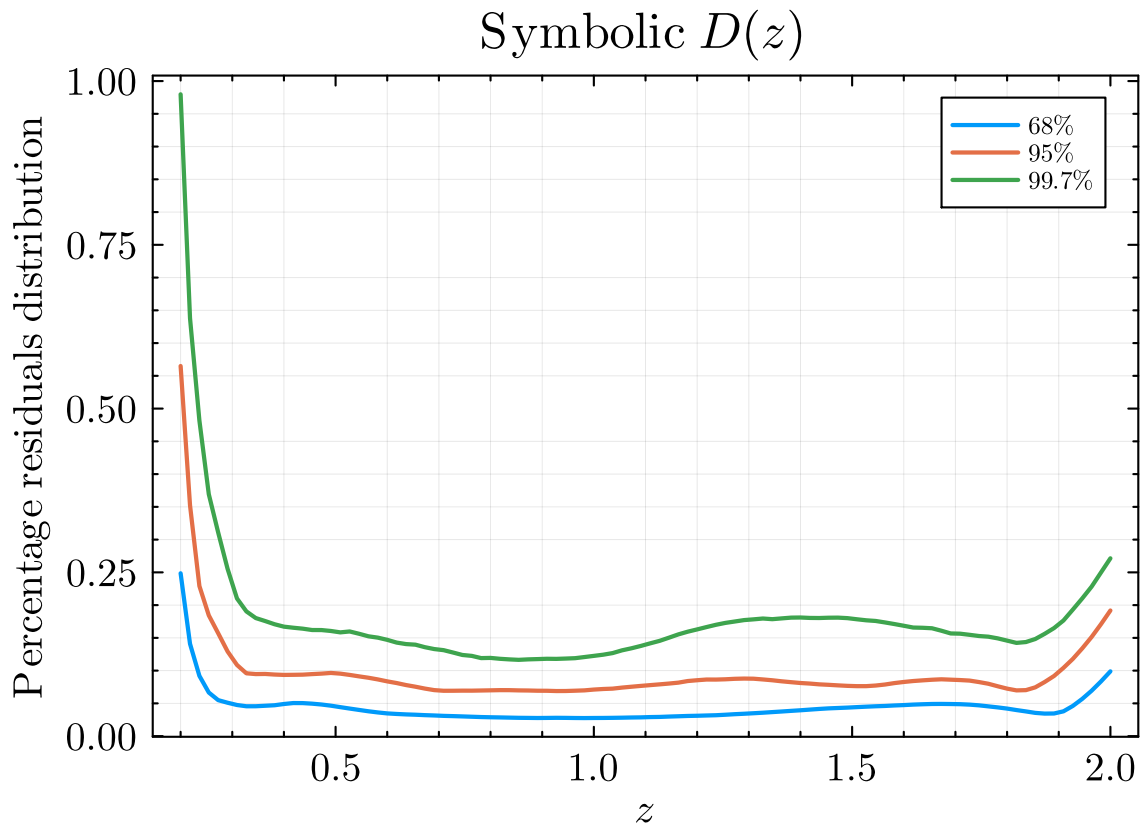


Figure 2. 68, 95, and 99.7 % distribution of the percentage errors, comparing the ODE and the symbolic regression $D(z)$.

resulting in a total simulated volume of $566 (h^{-1} \text{Gpc})^3$. Initial conditions were generated with second-order Lagrangian Perturbation Theory (2LPT) and evolved using GADGET2. Particle snapshots were recorded at six redshifts: $z = 3, 2, 1, 0.61, 0.51$, and 0.38 .

Halos were identified with the ROCKSTAR halo finder, and galaxies were probabilistically assigned based on the virial mass of halos, ensuring that mock galaxies were consistent with observational data.

The resulting mock galaxy catalogs correspond to redshifts $z = 0.38$ (LOWZ), $z = 0.51$ (CMASS1), and $z = 0.61$ (CMASS2), mirroring the observational data sets; specifically, the measurement employed in this work are those at $z = 0.61$. These simulations provide a robust framework for testing cosmological models and statistical methods, owing to their large volume and the precision of the mock galaxy distributions.¹⁴ Here we analyzed these simulations up to $k_{\text{max}} = 0.12 h/\text{Mpc}$.

The Baryon Oscillation Spectroscopic Survey (BOSS) provides publicly available redshift catalogs that enable galaxy clustering measurements [92, 93]. Power spectrum multipoles and window function matrices are the ones used in [94].

The BOSS dataset is divided into subsamples based on the Northern and Southern Galactic Caps (NGC and SGC). We divide them in two redshift bins (CMASS and LOWZ),

¹⁴The Julia version of the PT-challenge likelihood is available [here](#).

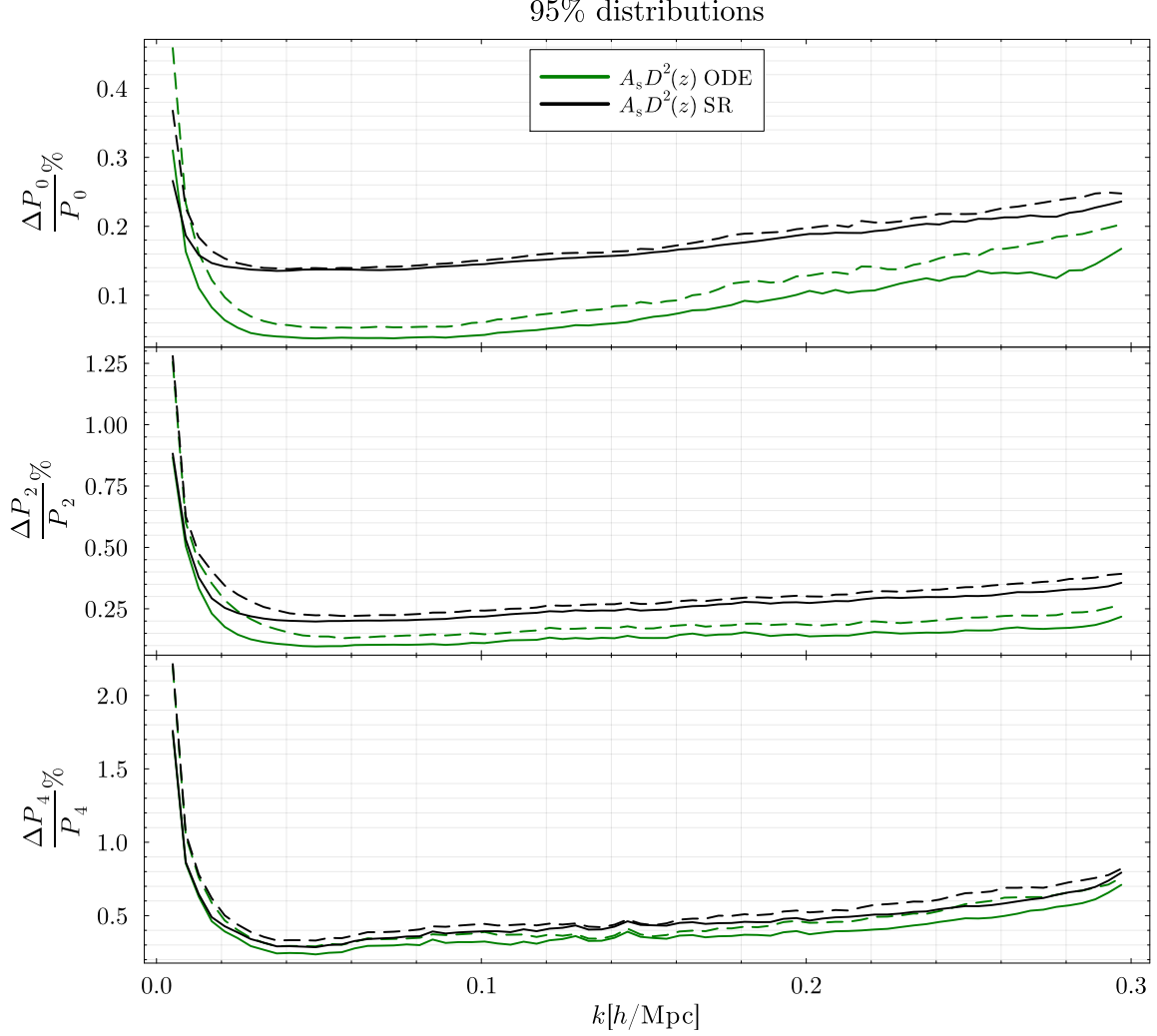


Figure 3. 95% distribution of the percentage errors, comparing the ODE and the symbolic regression rescaling.

corresponding to effective redshifts $z_{\text{eff}} = [0.57, 0.32]$ respectively. This results in a total of four sets of multipoles.¹⁵ We fit the power spectra with $k_{\text{max}} = 0.23 h/\text{Mpc}$ for CMASS and $0.20 h/\text{Mpc}$ for LOWZ.

We validate the accuracy of our emulators on these datasets by comparing the Bayesian posteriors produced by `Effort.jl` and `pybird`. Specifically, we present the posterior distributions for the cosmological and EFT parameters that `pybird` cannot marginalize analytically, although `Effort.jl` does not perform the analytical marginalization. To preserve the secrecy of the true cosmology of the PT-challenge, axis ticks and labels are omitted from the plots related to that analysis.

Regarding the prior, we used the same prior and analysis settings used in [53] from the West Coast Team and we developed a different emulator, tailored to have only 3

¹⁵The `Julia` version of the BOSS likelihood is available [here](#).

input cosmological parameters.¹⁶ The results demonstrate remarkable accuracy, with an excellent agreement between the two methods, consistent with Montecarlo noise. In terms of performance, the `pybird` chains were generated using the `MontePython` sampler [95] and required several hours to obtain a very high degree of convergence on multiple CPUs of a computing cluster. Specifically, we used `MontePython` adaptive Metropolis-Hastings sampler, with the covariance matrix and jumping-factor updates; convergence was monitored using the Gelman-Rubin \hat{R} statistics [96], ensuring $\hat{R} - 1 < 0.01$ for all parameters. `Effort.jl` achieved convergence with significantly higher efficiency. We ran four chains for both the NUTS and MCHMC samplers, using 500 and 10,000 burn-in steps and 2,000 and 200,000 accepted steps, respectively. These chains were initialized using `Pathfinder.jl`, a variational inference method that quickly generates samples from the typical set [48, 97]. The entire analysis with `Effort.jl` required approximately 10 minutes on a laptop and converged to the same posterior distribution as `pybird`. The Effective Sample Size per second (ESS/s) was 1.2 for NUTS and 5.1 for MCHMC. We stress that, given the large volume of the PT-challenge, this is a very stringent test of the accuracy our emulator.

A direct comparison of sampling efficiency with `pybird` was not made, as `pybird` samples from a smaller-dimensional parameter space. However, even when analyzing a higher-dimensional space, `Effort.jl` achieves a sampling efficiency several orders of magnitude greater than standard analysis pipelines.

For the BOSS analysis, the agreement between `Effort.jl` and `pybird` is similarly outstanding, as shown in figure 5. We employed the same priors as in [94], also including the correlated prior among EFT parameters. The `pybird` chains, using the `MontePython` sampler, required a few days on a computing cluster to obtain chains with a very high degree of convergence. Using `Effort.jl`, we ran eight chains for both NUTS and MCHMC, with the same burn-in and accepted steps as in the PT challenge. Chains were again initialized from a `Pathfinder.jl` run. The total wall-clock time was slightly over one hour, with ESS/s of 0.4 for NUTS and 2.4 for MCHMC.

Finally, we report the runtime of `CLASS` and `pybird` and compare with that of `Effort.jl`, to benchmark the code timing. For a DESI-like scenario with 6 redshift bins, the standard pipeline takes 1.7 seconds to run whereas `Effort.jl` requires only 650 μ s: an improvement of more than three orders of magnitude. We emphasize that the computational gain when doing inference comes both from the improved speed and better analysis algorithm.

5 Conclusions

In this work, we introduced `Effort.jl`,¹⁷ a novel emulator for the EFTofLSS with a strong emphasis on computational performance. The design of `Effort.jl` prioritizes efficiency without compromising accuracy, leveraging state-of-the-art numerical methods and machine learning techniques to provide robust predictions for cosmological analyses.

¹⁶We will not release the trained emulators for this part of the work to ensure the input cosmology remains undisclosed to the community.

¹⁷A repository, with the files to reproduce the plots of the paper, can be found [here](#), while the training and validation dataset have been uploaded on [Zenodo](#).

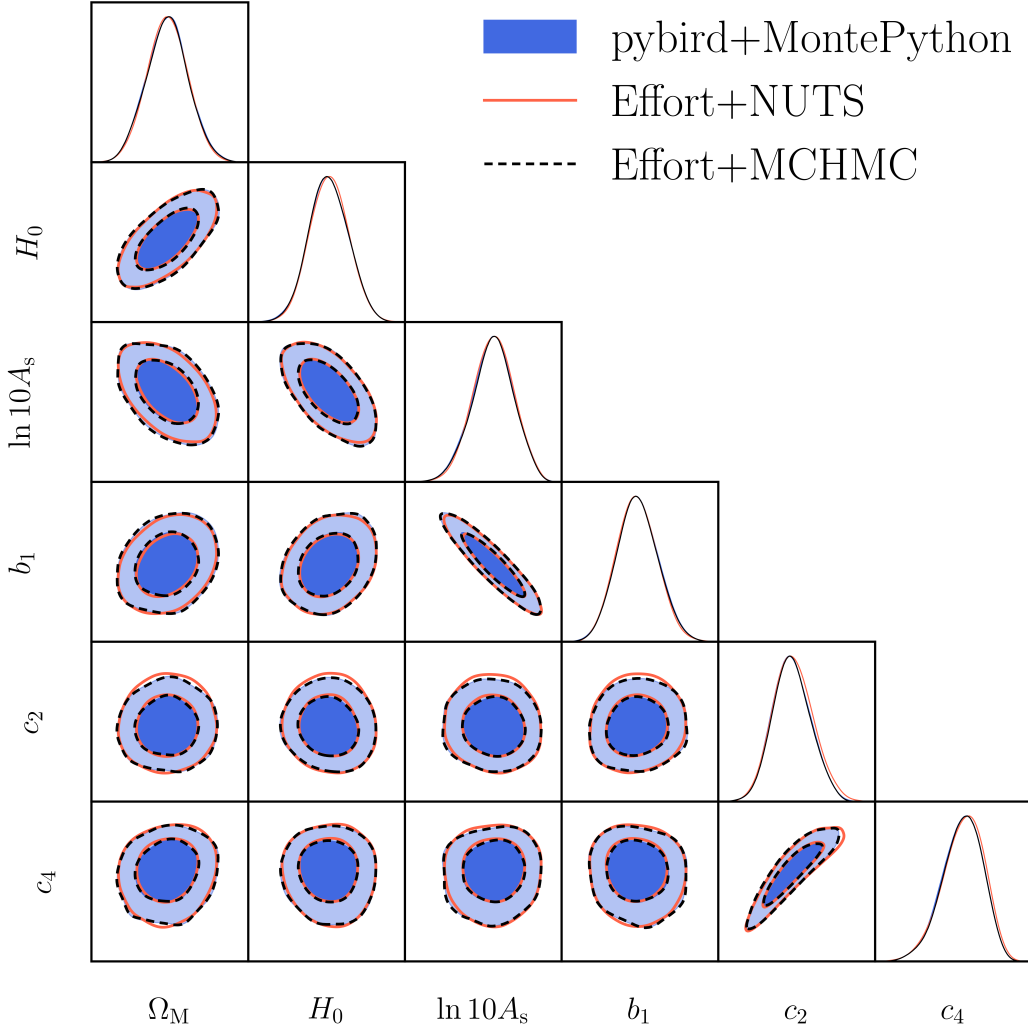


Figure 4. Triangle plot showing the standard `pybird` chains alongside those obtained using `Effort.jl` in combination with `Turing.jl`, for the PT-challenge analysis, obtained with the NUTS and MCHMC samplers. To align with the PT-challenge guidelines, axis ticks and labels have been removed to avoid disclosing the true cosmology.

Great care has been devoted to implementing observational effects, such as the Alcock-Paczynski effect and window mask convolution, ensuring high computational efficiency. This rigorous implementation enables `Effort.jl` to handle the complexity of LSS data while maintaining precise modelling of observational effects. We checked the accuracy of these calculations both against standard Boltzmann solvers and `pybird`, finding an excellent agreement.

We explored various preprocessing strategies, showcasing their significant impact on the emulator accuracy. Notably, we combined numerical method, such as solving ODE and symbolic regression, with NN emulators of cosmological observables. This hybrid approach demonstrates the potential for improving emulator precision and efficiency by leveraging domain knowledge readily available to cosmologists.

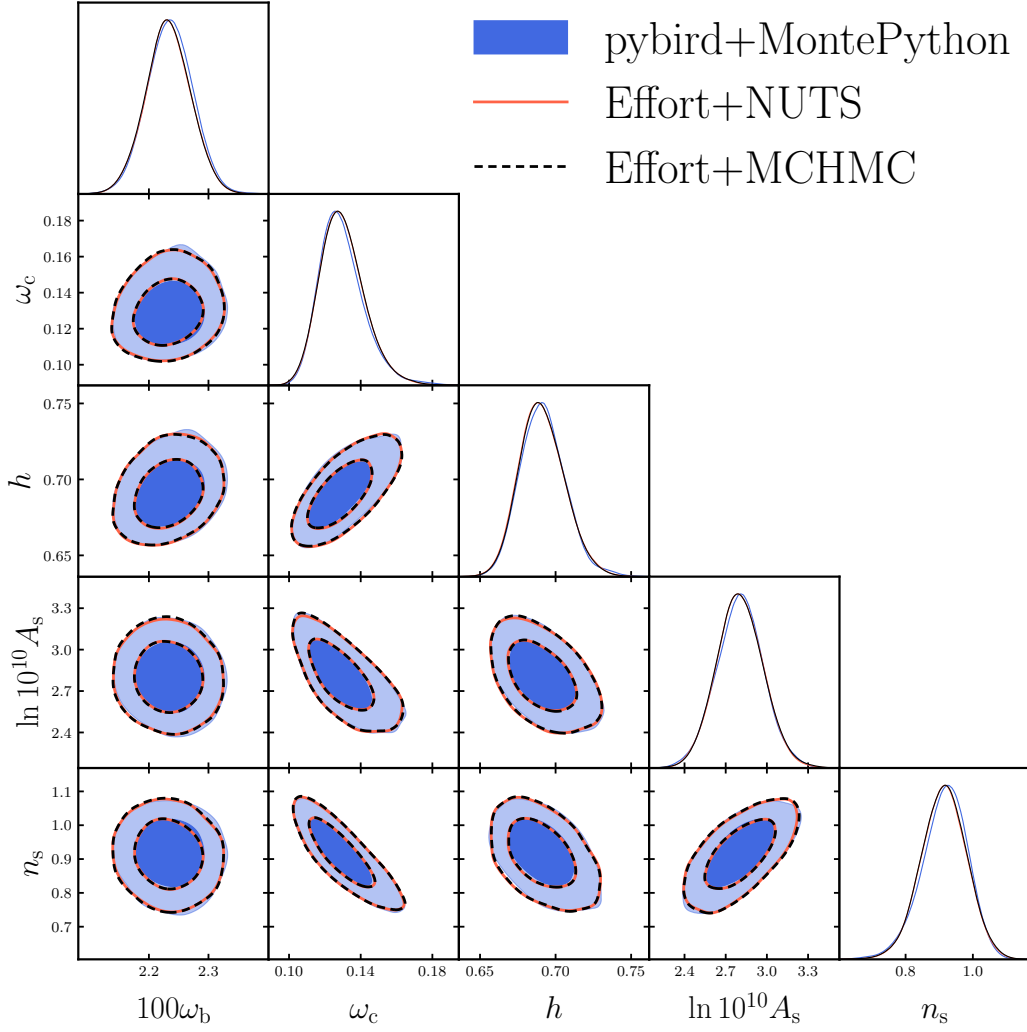


Figure 5. Triangle plot showing the standard `pybird` chains alongside those obtained using `Effort.jl`, for the BOSS analysis, focusing on cosmological parameters. As in figure 4, we compare the standard `pybird` chains with the ones obtained with `Effort.jl` coupled with NUTS and MCHMC.

The coupling of `Effort.jl` with the `Turing.jl` framework highlights its versatility, allowing it to efficiently integrate with gradient-based samplers such as NUTS and MCHMC. Our results demonstrate the ability of `Effort.jl` to sample Bayesian posteriors very efficiently, reducing computational costs significantly while achieving excellent agreement with standard pipelines. In particular, this will become of paramount importance when considering scenarios, like those found in [98–103], where analytical marginalization is not a viable option and we need to explore the full parameter space.

Looking forward, we plan to apply `Effort.jl` to the forthcoming datasets from the DESI survey [3], harnessing its computational efficiency to analyze next-generation cosmological data with precision. In addition, we foresee significant potential in combining `Effort.jl` with tools such as `Capse.jl` [14], which emulates CMB observables, and `Blast.jl` [104], which computes the photometric 3×2 pt. This integration will facilitate efficient joint analyses of diverse cosmological datasets and probes.

The combination of `Effort.jl` and `Blast.jl` is particularly compelling, as it enables the joint analysis of the Euclid main probes without relying on the Limber approximation, which can introduce inaccuracies in cosmological inference [105, 106]. Furthermore, as demonstrated in [36], the use of HMC samplers is expected to play a pivotal role in the analysis of 3×2 pt datasets from Stage IV surveys like Euclid. This approach will become even more critical when combining Euclid’s photometric and spectroscopic datasets for joint analyses.

The capabilities of `Effort.jl` provide a solid foundation for further enhancements and expansions. Additionally, we plan to extend the use of symbolic regression to $w_0 w_a$ cosmologies, providing a computationally inexpensive replacement for the ODE computations, further optimizing the preprocessing pipeline.

The modular structure of `Effort.jl` is sufficiently general to support compatibility with other EFT-based codes, such as `velocileptors` [107], `CLASS-PT` [108], `FOLPS` [109], and `CLASS-OneLoop` [110]. This flexibility opens the door for training `Effort.jl` to emulate these codes as well, broadening its application and usability.

Regarding the limitations of our work, we identify two aspects that are worth discussing. First, as a surrogate model, its applicability is inherently confined to the parameter space explored during emulator training. This restriction becomes especially critical for extended cosmological models, where projection effects may drive inferences toward the edges of the trained domain [94, 111], potentially necessitating a full retraining of the emulators.

Second, `Effort.jl`’s workflow depends not only on neural-network emulators but also on high-performance routines for computing background quantities and observational effects. Adapting `Effort.jl` to a new model therefore requires either a careful, bespoke implementation of its governing equations or the development of an additional surrogate model to handle the relevant background computations. While for some cases, like a scale independent nDGP analysis this can be easily done [112], it would be more complex for models with a richer phenomenology, like when considering particle-physics inspired models [113, 114]. Finally, as we have shown for the AP effect and window convolution, achieving both accuracy and efficient gradient evaluation demands meticulous optimization; any further extensions will require the same level of care.

Finally, we are actively working on a `jax`-based version of `Effort.jl`. Since the cosmological community is more proficient with `python`, we thus see advantageous to provide a version of our software compatible with such a language. While a fully equivalent `jax` implementation of `Capse.jl` is already complete,¹⁸ we are now focussing on translating `Effort.jl` into `jax`.

Acknowledgments

MB is supported in part by funding from the Government of Canada’s New Frontiers in Research Fund (NFRF). MB also acknowledges the support of the Canadian Space Agency and the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number RGPIN-2019-03908]. This research was enabled in part by support provided by Compute Ontario (computeontario.ca) and the Digital Research Alliance of Canada (alliancecan.ca). MB acknowledges financial support from INAF MiniGrant 2022.

¹⁸<https://github.com/CosmologicalEmulators/jaxcapse>.

MB is expecially grateful to Anton Baleato-Lizancos, Guadalupe Canas-Herrera, Pedro Carrilho, Emanuele Castorina, Arnaud de Mattia, Minas Karamanis, Chiara Moretti, Andrea Pezzotta, Uros Seljak, and Martin White for useful discussions. GDA is especially grateful to Pierre Zhang for useful discussions. We thank Takahiro Nishimichi for the PT-challenge simulations. Part of this work has been carried on the High Performance Computing facility of the University of Parma, whose support team we thank. MB thanks University of California Berkeley for hospitality during his visit, during which this project was started. CC is supported by Italian Research Center on High Performance Computing Big Data and Quantum Computing (ICSC), project funded by European Union — NextGenerationEU — and National Recovery and Resilience Plan (NRRP) — Mission 4 Component 2 within the activities of Spoke 3 (Astrophysics and Cosmos Observations).

A The galaxy power spectrum in the EFTofLSS

The Effective Field Theory of Large-Scale Structure (EFTofLSS) provides a systematic approach to modeling the redshift-space galaxy power spectrum by thoroughly accounting for the effects of small-scale physics on large-scale clustering. This framework extends standard perturbation theory by introducing counterterms that describe how small-scale physics, including galaxy formation, influences the observed large-scale distribution. We give a concise overview below and refer readers to [115, 116] for more details.

The one-loop EFTofLSS expression for the redshift-space galaxy power spectrum is:

$$\begin{aligned}
P_g(k, \mu) = & Z_1(\mu)^2 P_{11}(k) + 2 \int \frac{d^3 q}{(2\pi)^3} Z_2(\mathbf{q}, \mathbf{k} - \mathbf{q}, \mu)^2 P_{11}(|\mathbf{k} - \mathbf{q}|) P_{11}(q) \\
& + 6 Z_1(\mu) P_{11}(k) \int \frac{d^3 q}{(2\pi)^3} Z_3(\mathbf{q}, -\mathbf{q}, \mathbf{k}, \mu) P_{11}(q) \\
& + 2 Z_1(\mu) P_{11}(k) \left(c_{\text{ct}} \frac{k^2}{k_{\text{M}}^2} + c_{r,1} \mu^2 \frac{k^2}{k_{\text{R}}^2} + c_{r,2} \mu^4 \frac{k^2}{k_{\text{R}}^2} \right) \\
& + \frac{1}{\bar{n}_g} \left(c_{\epsilon,0} + c_{\epsilon,1} \frac{k^2}{k_{\text{M}}^2} + c_{\epsilon,2} f \mu^2 \frac{k^2}{k_{\text{M}}^2} \right), \tag{A.1}
\end{aligned}$$

where we follow the notation of [117]. This includes linear contributions, one-loop Standard Perturbation Theory (SPT) terms, counterterms, and stochastic terms. Here, μ is the cosine of the angle between the line of sight (LoS) and the wavenumber \mathbf{k} ; $P_{11}(k)$ is the linear matter power spectrum; and f is the growth factor. The scale k_{M}^{-1} characterizes the size of collapsed objects, while k_{R}^{-1} governs counterterms needed to handle products of the velocity field at the same point [117, 118], and \bar{n}_g denotes the mean galaxy number density.

The functions Z_n are the redshift-space galaxy density kernels of order n :

$$\begin{aligned}
Z_1(\mathbf{q}_1) &= K_1(\mathbf{q}_1) + f \mu_1^2 G_1(\mathbf{q}_1) = b_1 + f \mu_1^2, \\
Z_2(\mathbf{q}_1, \mathbf{q}_2, \mu) &= K_2(\mathbf{q}_1, \mathbf{q}_2) + f \mu_{12}^2 G_2(\mathbf{q}_1, \mathbf{q}_2) + \frac{1}{2} f \mu q \left(\frac{\mu_2}{q_2} G_1(\mathbf{q}_2) Z_1(\mathbf{q}_1) + \text{perm.} \right), \\
Z_3(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mu) &= K_3(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3) + f \mu_{123}^2 G_3(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3) \\
&+ \frac{1}{3} f \mu q \left(\frac{\mu_3}{q_3} G_1(\mathbf{q}_3) Z_2(\mathbf{q}_1, \mathbf{q}_2, \mu_{123}) + \frac{\mu_{23}}{q_{23}} G_2(\mathbf{q}_2, \mathbf{q}_3) Z_1(\mathbf{q}_1) + \text{cyc.} \right), \tag{A.2}
\end{aligned}$$

and K_n are the corresponding n th-order galaxy density kernels in real space:

$$\begin{aligned}
K_1 &= b_1, \\
K_2(\mathbf{q}_1, \mathbf{q}_2) &= b_1 \frac{\mathbf{q}_1 \cdot \mathbf{q}_2}{q_1^2} + b_2 \left(F_2(\mathbf{q}_1, \mathbf{q}_2) - \frac{\mathbf{q}_1 \cdot \mathbf{q}_2}{q_1^2} \right) + b_4 + \text{perm.}, \\
K_3(k, q) &= \frac{b_1}{504k^3q^3} \left(-38k^5q + 48k^3q^3 - 18kq^5 + 9(k^2 - q^2)^3 \log \left[\frac{k-q}{k+q} \right] \right) \\
&\quad + \frac{b_3}{756k^3q^5} \left(2kq(k^2 + q^2)(3k^4 - 14k^2q^2 + 3q^4) + 3(k^2 - q^2)^4 \log \left[\frac{k-q}{k+q} \right] \right). \tag{A.3}
\end{aligned}$$

We omit the full expressions for the second-order kernel F_2 and velocity kernels G_n (from SPT) for brevity. Altogether, the model uses four bias parameters b_1, b_2, b_3, b_4 , three counterterm parameters $c_{\text{ct}}, c_{r,1}, c_{r,2}$, and three stochastic parameters $c_{\epsilon,0}, c_{\epsilon,1}, c_{\epsilon,2}$, for a total of ten parameters:

$$\{b_1, b_2, b_3, b_4, c_{\text{ct}}, c_{r,1}, c_{r,2}, c_{\epsilon,0}, c_{\epsilon,1}, c_{\epsilon,2}\}. \tag{A.4}$$

Finally, we employ IR-resummation [56, 119, 120] to account for the large effects of long-wavelength displacements.

B CLASS and pybird settings

In this appendix we report the `python` code used to generate the training dataset, in order to ensure full reproducibility and showing the settings for `CLASS` and `pybird`.

```

def pybird(CosmoDict):
    z = CosmoDict["z"]
    cosmo_params = {
        "output": "mPk", #Request the matter power spectrum (mPk)
        "P_k_max_h/Mpc": 20.0, #Maximum k value (in units of h/Mpc)
        "z_pk": "0.0,3", #Redshifts at which evaluate the power spectra
        "h": CosmoDict["H0"] / 100, #Hubble parameter
        "omega_b": CosmoDict["ombh2"], #Baryon density parameter
        "omega_cdm": CosmoDict["omch2"], #Cold dark matter density parameter
        "ln10^{10}A_s": CosmoDict["ln10As"], #Amplitude of the primordial power spectrum
        "n_s": CosmoDict["ns"], #Scalar spectral index
        "tau_reio": 0.0568, #Optical depth at reionization
        "N_ur": 2.0308,
        "N_ncdm": 1,
        "m_ncdm": CosmoDict["Mnu"],
        "use_ppf": "yes",
        "w0_fld": -1.,
        "wa_fld": 0.,
        "fluid_equation_of_state": "CLP",
        "cs2_fld": 1.,
        "Omega_Lambda": 0.,
        "Omega_scf": 0.
    }

    # Initialize Class and compute linear power spectrum
    M = Class()

```

```

M.set(cosmo_params)
M.compute()

# Generate logarithmic k values and compute linear power spectrum
kk = 10 ** np.linspace(-5, 0, 200)
pk_lin = [M.pk_cb(k * M.h(), z) * M.h()**3 for k in kk]

# Compute growth factors
D1 = M.scale_independent_growth_factor(z)
f1 = M.scale_independent_growth_factor_f(z)

# Initialize Correlator
N = Correlator()
dk = 0.004
kd = np.arange(0.005, 0.3, dk)

# Set parameters for the correlator
N.set({
    "output": "bPk",
    "multipole": 3,
    "kmax": 0.3,
    "xdata": kd,
    "km": 0.7,
    "kr": 0.35,
    "nd": 3e-4,
    "eft_basis": "eftoflss",
    "with_stoch": False,
    "with_bias": False,
    "with_resum": True})

# Compute the correlator with input power spectrum and growth factors
N.compute({
    "kk": kk,
    "pk_lin": pk_lin,
    "f": f1
})
P1l1 = N.bird.P1l1
Ploop1 = N.bird.Ploop1
Pct1 = N.bird.Pct1
return kk, pk_lin, kd, P1l1, Ploop1, Pct1

```

C Fixed redshift emulator

In this appendix we explore the accuracy gain when working with a fixed redshift emulator. As noted in section 4.1, this class of emulators, albeit less flexible than the one which takes redshift as an input, has the advantage of a reduced number of input parameters and hence dynamical range of the output features. For such a reason we expect the accuracy of such an emulator, as employed in [84], to be higher than its more flexible counterpart. As can be seen in figure 6, the residuals for the monopole and quadrupole reduce by a factor of 2 whereas the hexadecapole one improve by a factor of 3.

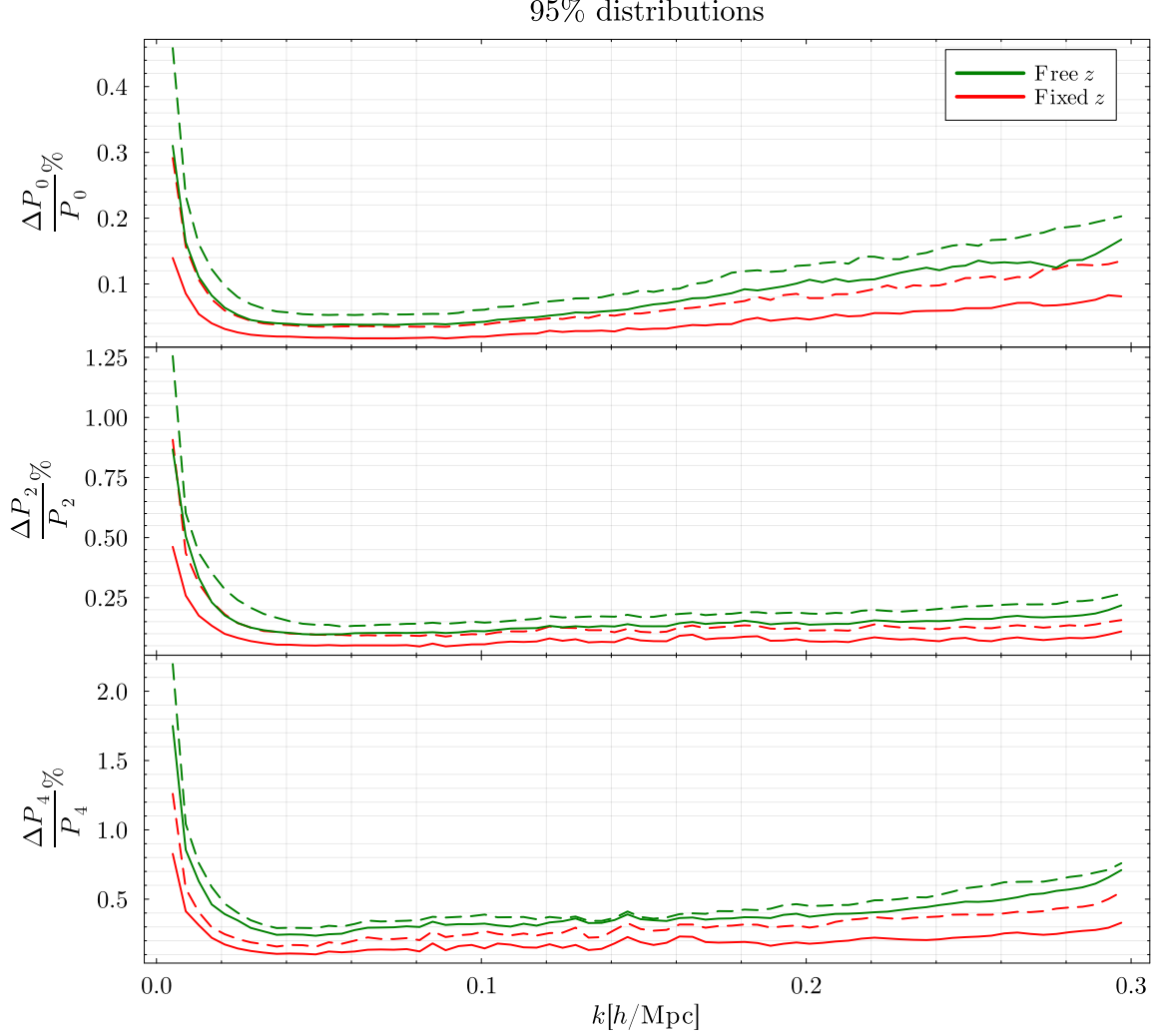


Figure 6. 95% distribution of the percentage errors for the free (green) and fixed (red) z emulators, with the $A_s D(z)$ rescaling.

References

- [1] DESI collaboration, *The DESI experiment part I: science, targeting, and survey design*, [arXiv:1611.00036](#) [INSPIRE].
- [2] DESI collaboration, *Overview of the instrumentation for the dark energy spectroscopic instrument*, *Astron. J.* **164** (2022) 207 [[arXiv:2205.10939](#)] [INSPIRE].
- [3] DESI collaboration, *DESI 2024 VII: cosmological constraints from the full-shape modeling of clustering measurements*, *JCAP* **07** (2025) 028 [[arXiv:2411.12022](#)] [INSPIRE].
- [4] EUCLID collaboration, *Euclid. I. Overview of the Euclid mission*, *Astron. Astrophys.* **697** (2025) A1 [[arXiv:2405.13491](#)] [INSPIRE].
- [5] R. Jimenez, L. Verde, H. Peiris and A. Kosowsky, *Fast cosmological parameter estimation from microwave background temperature and polarization power spectra*, *Phys. Rev. D* **70** (2004) 023005 [[astro-ph/0404237](#)] [INSPIRE].

- [6] W.A. Fendt and B.D. Wandelt, *Pico: parameters for the impatient cosmologist*, *Astrophys. J.* **654** (2006) 2 [[astro-ph/0606709](#)] [[INSPIRE](#)].
- [7] T. Auld, M. Bridges, M.P. Hobson and S.F. Gull, *Fast cosmological parameter estimation using neural networks*, *Mon. Not. Roy. Astron. Soc.* **376** (2007) L11 [[astro-ph/0608174](#)] [[INSPIRE](#)].
- [8] A. Lewis, A. Challinor and A. Lasenby, *Efficient computation of CMB anisotropies in closed FRW models*, *Astrophys. J.* **538** (2000) 473 [[astro-ph/9911177](#)] [[INSPIRE](#)].
- [9] D. Blas, J. Lesgourgues and T. Tram, *The Cosmic Linear Anisotropy Solving System (CLASS) II: approximation schemes*, *JCAP* **07** (2011) 034 [[arXiv:1104.2933](#)] [[INSPIRE](#)].
- [10] J. Albers et al., *CosmicNet. Part I. Physics-driven implementation of neural networks within Einstein-Boltzmann solvers*, *JCAP* **09** (2019) 028 [[arXiv:1907.05764](#)] [[INSPIRE](#)].
- [11] A. Spurio Mancini et al., *CosmoPower: emulating cosmological power spectra for accelerated Bayesian inference from next-generation surveys*, *Mon. Not. Roy. Astron. Soc.* **511** (2022) 1771 [[arXiv:2106.03846](#)] [[INSPIRE](#)].
- [12] A. Nygaard, E.B. Holm, S. Hannestad and T. Tram, *CONNECT: a neural network based framework for emulating cosmological observables and cosmological parameter inference*, *JCAP* **05** (2023) 025 [[arXiv:2205.15726](#)] [[INSPIRE](#)].
- [13] S. Günther et al., *CosmicNet II: emulating extended cosmologies with efficient and accurate neural networks*, *JCAP* **11** (2022) 035 [[arXiv:2207.05707](#)] [[INSPIRE](#)].
- [14] M. Bonici, F. Bianchini and J. Ruiz-Zapatero, *Capse.jl: efficient and auto-differentiable CMB power spectra emulation*, *Open J. Astrophys.* **7** (2024) [[arXiv:2307.14339](#)] [[INSPIRE](#)].
- [15] A. Mootoovaloo, A.H. Jaffe, A.F. Heavens and F. Leclercq, *Kernel-based emulator for the 3D matter power spectrum from CLASS*, *Astron. Comput.* **38** (2022) 100508 [[arXiv:2105.02256](#)] [[INSPIRE](#)].
- [16] J. Donald-McCann, F. Beutler, K. Koyama and M. Karamanis, *matryoshka: halo model emulator for the galaxy power spectrum*, *Mon. Not. Roy. Astron. Soc.* **511** (2022) 3768 [[arXiv:2109.15236](#)] [[INSPIRE](#)].
- [17] G. Aricò, R.E. Angulo and M. Zennaro, *Accelerating large-scale-structure data analyses by emulating Boltzmann solvers and Lagrangian perturbation theory*, *Open Res. Europe* **1** (2022) 152 [[arXiv:2104.14568](#)] [[INSPIRE](#)].
- [18] D.J. Bartlett et al., *A precise symbolic emulator of the linear matter power spectrum*, *Astron. Astrophys.* **686** (2024) A209 [[arXiv:2311.15865](#)] [[INSPIRE](#)].
- [19] T. Bakx, N.E. Chisari and Z. Vlah, *COBRA: optimal factorization of cosmological observables*, *Phys. Rev. Lett.* **134** (2025) 191002 [[arXiv:2407.04660](#)] [[INSPIRE](#)].
- [20] A. Eggemeier et al., *COMET: clustering observables modelled by emulated perturbation theory*, *Mon. Not. Roy. Astron. Soc.* **519** (2022) 2962 [[arXiv:2208.01070](#)] [[INSPIRE](#)].
- [21] J. Donald-McCann, K. Koyama and F. Beutler, *matryoshka II: accelerating effective field theory analyses of the galaxy power spectrum*, *Mon. Not. Roy. Astron. Soc.* **518** (2022) 3106 [[arXiv:2202.07557](#)] [[INSPIRE](#)].
- [22] S. Trusov, P. Zarrouk and S. Cole, *Neural network-based model of galaxy power spectrum: fast full-shape galaxy power spectrum analysis*, *Mon. Not. Roy. Astron. Soc.* **538** (2025) 1789 [[arXiv:2403.20093](#)] [[INSPIRE](#)].

- [23] A. Manrique-Yus and E. Sellentin, *Euclid-era cosmology for everyone: neural net assisted MCMC sampling for the joint 3×2 likelihood*, *Mon. Not. Roy. Astron. Soc.* **491** (2020) 2655 [[arXiv:1907.05881](#)] [[INSPIRE](#)].
- [24] A. Mootooyaloo, A.F. Heavens, A.H. Jaffe and F. Leclercq, *Parameter inference for weak lensing using Gaussian processes and MOPED*, *Mon. Not. Roy. Astron. Soc.* **497** (2020) 2213 [[arXiv:2005.06551](#)] [[INSPIRE](#)].
- [25] M. Bonici, L. Biggio, C. Carbone and L. Guzzo, *Fast emulation of two-point angular statistics for photometric galaxy surveys*, *Mon. Not. Roy. Astron. Soc.* **531** (2024) 4203 [[arXiv:2206.14208](#)] [[INSPIRE](#)].
- [26] C.-H. To et al., *LINNA: Likelihood Inference Neural Network Accelerator*, *JCAP* **01** (2023) 016 [[arXiv:2203.05583](#)] [[INSPIRE](#)].
- [27] K. Zhong et al., *Attention-based neural network emulators for multiprobe data vectors. I. Forecasting the growth-geometry split*, *Phys. Rev. D* **111** (2025) 123519 [[arXiv:2402.17716](#)] [[INSPIRE](#)].
- [28] E. Saraivanov et al., *Attention-based neural network emulators for multiprobe data vectors. II. Assessing tension metrics*, *Phys. Rev. D* **111** (2025) 123520 [[arXiv:2403.12337](#)] [[INSPIRE](#)].
- [29] LSST DARK ENERGY SCIENCE collaboration, *Machine learning LSST 3x2pt analyses — forecasting the impact of systematics on cosmological constraints using neural networks*, [arXiv:2403.11797](#) [[INSPIRE](#)].
- [30] M.D. Hoffman and A. Gelman, *The no-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo*, [arXiv:1111.4246](#) [[INSPIRE](#)].
- [31] M. Betancourt, *A conceptual introduction to Hamiltonian Monte Carlo*, [arXiv:1701.02434](#) [[INSPIRE](#)].
- [32] M. Millea, E. Anderes and B.D. Wandelt, *Sampling-based inference of the primordial CMB and gravitational lensing*, *Phys. Rev. D* **102** (2020) 123542 [[arXiv:2002.00965](#)] [[INSPIRE](#)].
- [33] Y. Li et al., *pmwd: a differentiable cosmological particle-mesh N-body library*, [arXiv:2211.09958](#) [[INSPIRE](#)].
- [34] Y. Li et al., *Differentiable cosmological simulation with the adjoint method*, *Astrophys. J. Suppl.* **270** (2024) 36 [[arXiv:2211.09815](#)] [[INSPIRE](#)].
- [35] J.-E. Campagne et al., *JAX-COSMO: an end-to-end differentiable and GPU accelerated cosmology library*, *Open J. Astrophys.* **6** (2023) 1 [[arXiv:2302.05163](#)] [[INSPIRE](#)].
- [36] D. Piras and A. Spurio Mancini, *CosmoPower-JAX: high-dimensional Bayesian inference with differentiable cosmological emulators*, *Open J. Astrophys.* **6** (2023) [[arXiv:2305.06347](#)] [[INSPIRE](#)].
- [37] O. Hahn, F. List and N. Porqueres, *DISCO-DJ I: a differentiable Einstein-Boltzmann solver for cosmology*, *JCAP* **06** (2024) 063 [[arXiv:2311.03291](#)] [[INSPIRE](#)].
- [38] M.S. Cagliari, E. Castorina, M. Bonici and D. Bianchi, *Optimal constraints on primordial non-Gaussianity with the eBOSS DR16 quasars in Fourier space*, *JCAP* **08** (2024) 036 [[arXiv:2309.15814](#)] [[INSPIRE](#)].
- [39] J. Ruiz-Zapatero et al., *LimberJack.jl: auto-differentiable methods for angular power spectra analyses*, *Open J. Astrophys.* **7** (2024) [[arXiv:2310.08306](#)] [[INSPIRE](#)].

- [40] A. Mootoovaloo, J. Ruiz-Zapatero, C. García-García and D. Alonso, *Assessment of gradient-based samplers in standard cosmological likelihoods*, *Mon. Not. Roy. Astron. Soc.* **534** (2024) 1668 [[arXiv:2406.04725](#)] [[INSPIRE](#)].
- [41] C. Giovanetti et al., *LINX: a fast, differentiable, and extensible big bang nucleosynthesis package*, [arXiv:2408.14538](#) [[INSPIRE](#)].
- [42] SPT-3G collaboration, *Cosmology from CMB lensing and delensed EE power spectra using 2019–2020 SPT-3G polarization data*, *Phys. Rev. D* **111** (2025) 083534 [[arXiv:2411.06000](#)] [[INSPIRE](#)].
- [43] H. Zhang et al., *HOD-informed prior for EFT-based full-shape analyses of LSS*, *JCAP* **04** (2025) 041 [[arXiv:2409.12937](#)] [[INSPIRE](#)].
- [44] A. Reeves, P. Zhang and H. Zheng, *PyBird-JAX: accelerated inference in large-scale structure with model-independent emulation of one-loop galaxy power spectra*, [arXiv:2507.20990](#) [[INSPIRE](#)].
- [45] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, second edition, Society for Industrial and Applied Mathematics, U.S.A. (2008).
- [46] M. Blondel and V. Roulet, *The elements of differentiable programming*, [arXiv:2403.14606](#).
- [47] S. Scardapane, *Alice’s adventures in a differentiable Wonderland — volume I, a tour of the land*, [arXiv:2404.17625](#).
- [48] G. Arya, M. Schauer, F. Schäfer and C. Rackauckas, *Automatic differentiation of programs with discrete randomness*, [arXiv:2210.08572](#).
- [49] B. Horowitz et al., *Differentiable stochastic halo occupation distribution*, *Mon. Not. Roy. Astron. Soc.* **529** (2024) 2473 [[arXiv:2211.03852](#)] [[INSPIRE](#)].
- [50] A.S. Jurling and J.R. Fienup, *Applications of algorithmic differentiation to phase retrieval algorithms*, *J. Opt. Soc. Amer. A* **31** (2014) 1348.
- [51] J. DeRose, S.-F. Chen, M. White and N. Kokron, *Neural network acceleration of large-scale structure theory calculations*, *JCAP* **04** (2022) 056 [[arXiv:2112.05889](#)] [[INSPIRE](#)].
- [52] J. Bezanson, A. Edelman, S. Karpinski and V.B. Shah, *Julia: a fresh approach to numerical computing*, *SIAM Rev.* **59** (2017) 65 [[arXiv:1411.1607](#)] [[INSPIRE](#)].
- [53] T. Nishimichi et al., *Blinded challenge for precision cosmology with large-scale structure: results from effective field theory for the redshift-space galaxy power spectrum*, *Phys. Rev. D* **102** (2020) 123541 [[arXiv:2003.08277](#)] [[INSPIRE](#)].
- [54] BOSS collaboration, *The clustering of galaxies in the SDSS-III Baryon Oscillation Spectroscopic Survey: RSD measurement from the LOS-dependent power spectrum of DR12 BOSS galaxies*, *Mon. Not. Roy. Astron. Soc.* **460** (2016) 4188 [[arXiv:1509.06386](#)] [[INSPIRE](#)].
- [55] J. Bezanson, A. Edelman, S. Karpinski and V.B. Shah, *Julia: a fresh approach to numerical computing*, *SIAM Rev.* **59** (2017) 65 [[arXiv:1411.1607](#)] [[INSPIRE](#)].
- [56] L. Senatore and M. Zaldarriaga, *The IR-resummed effective field theory of large scale structures*, *JCAP* **02** (2015) 013 [[arXiv:1404.5954](#)] [[INSPIRE](#)].
- [57] T. Baldauf, M. Mirbabayi, M. Simonović and M. Zaldarriaga, *Equivalence principle and the baryon acoustic peak*, *Phys. Rev. D* **92** (2015) 043514 [[arXiv:1504.04366](#)] [[INSPIRE](#)].
- [58] A.G. Sanchez, A.N. Ruiz, J.G. Jara and N.D. Padilla, *Evolution mapping: a new approach to describe matter clustering in the non-linear regime*, *Mon. Not. Roy. Astron. Soc.* **514** (2022) 5673 [[arXiv:2108.12710](#)] [[INSPIRE](#)].

- [59] M.A. Rodriguez-Meza et al., *fkPT: constraining scale-dependent modified gravity with the full-shape galaxy power spectrum*, *JCAP* **03** (2024) 049 [[arXiv:2312.10510](#)] [[INSPIRE](#)].
- [60] S. Ioffe and C. Szegedy, *Batch normalization: accelerating deep network training by reducing internal covariate shift*, [arXiv:1502.03167](#) [[INSPIRE](#)].
- [61] M. Zennaro et al., *Initial conditions for accurate N-body simulations of massive neutrino cosmologies*, *Mon. Not. Roy. Astron. Soc.* **466** (2017) 3244 [[arXiv:1605.05283](#)] [[INSPIRE](#)].
- [62] A.E. Bayer, A. Banerjee and Y. Feng, *A fast particle-mesh simulation of non-linear cosmological structure formation with massive neutrinos*, *JCAP* **01** (2021) 016 [[arXiv:2007.13394](#)] [[INSPIRE](#)].
- [63] M. Chevallier and D. Polarski, *Accelerating universes with scaling dark matter*, *Int. J. Mod. Phys. D* **10** (2001) 213 [[gr-qc/0009008](#)] [[INSPIRE](#)].
- [64] E.V. Linder, *Exploring the expansion history of the universe*, *Phys. Rev. Lett.* **90** (2003) 091301 [[astro-ph/0208512](#)] [[INSPIRE](#)].
- [65] K. Akita and M. Yamaguchi, *A precision calculation of relic neutrino decoupling*, *JCAP* **08** (2020) 012 [[arXiv:2005.07047](#)] [[INSPIRE](#)].
- [66] J. Froustey, C. Pitrou and M.C. Volpe, *Neutrino decoupling including flavour oscillations and primordial nucleosynthesis*, *JCAP* **12** (2020) 015 [[arXiv:2008.01074](#)] [[INSPIRE](#)].
- [67] J.J. Bennett et al., *Towards a precision calculation of N_{eff} in the Standard Model. Part II. Neutrino decoupling in the presence of flavour oscillations and finite-temperature QED*, *JCAP* **04** (2021) 073 [[arXiv:2012.02726](#)] [[INSPIRE](#)].
- [68] LSST DARK ENERGY SCIENCE collaboration, *Core cosmology library: precision cosmological predictions for LSST*, *Astrophys. J. Suppl.* **242** (2019) 2 [[arXiv:1812.05995](#)] [[INSPIRE](#)].
- [69] C. Tsitouras, *Runge-Kutta pairs of order 5(4) satisfying only the first column simplifying assumption*, *Comput. Math. Appl.* **62** (2011) 770.
- [70] C. Rackauckas and Q. Nie, *DifferentialEquations.jl — a performant and feature-rich ecosystem for solving differential equations in Julia*, *J. Open Res. Software* **5** (2017) 15.
- [71] C. Alcock and B. Paczynski, *An evolution free test for non-zero cosmological constant*, *Nature* **281** (1979) 358 [[INSPIRE](#)].
- [72] G. D’Amico, L. Senatore and P. Zhang, *Limits on w CDM from the EFTofLSS with the PyBird code*, *JCAP* **01** (2021) 006 [[arXiv:2003.07956](#)] [[INSPIRE](#)].
- [73] M.M. Ivanov, M. Simonović and M. Zaldarriaga, *Cosmological parameters from the BOSS galaxy power spectrum*, *JCAP* **05** (2020) 042 [[arXiv:1909.05277](#)] [[INSPIRE](#)].
- [74] W.E. Ballinger, J.A. Peacock and A.F. Heavens, *Measuring the cosmological constant with redshift surveys*, *Mon. Not. Roy. Astron. Soc.* **282** (1996) 877 [[astro-ph/9605017](#)] [[INSPIRE](#)].
- [75] H.-J. Seo and D.J. Eisenstein, *Probing dark energy with baryonic acoustic oscillations from future large galaxy redshift surveys*, *Astrophys. J.* **598** (2003) 720 [[astro-ph/0307460](#)] [[INSPIRE](#)].
- [76] BOSS collaboration, *The clustering of galaxies in the completed SDSS-III Baryon Oscillation Spectroscopic Survey: anisotropic galaxy clustering in Fourier-space*, *Mon. Not. Roy. Astron. Soc.* **466** (2017) 2242 [[arXiv:1607.03150](#)] [[INSPIRE](#)].
- [77] J. Revels, M. Lubin and T. Papamarkou, *Forward-mode automatic differentiation in Julia*, [arXiv:1607.07892](#).

- [78] F. Beutler, E. Castorina and P. Zhang, *Interpreting measurements of the anisotropic galaxy power spectrum*, *JCAP* **03** (2019) 040 [[arXiv:1810.05051](#)] [[INSPIRE](#)].
- [79] M. Abbott et al., *mcabbott/tullio.jl: v0.3.7*, [Zenodo](#), October 2023.
- [80] H. Ge, K. Xu and Z. Ghahramani, *Turing: a language for flexible probabilistic inference*, in *International conference on Artificial Intelligence and Statistics, AISTATS 2018, 9–11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, <http://proceedings.mlr.press/v84/ge18b.html> (2018), p. 1682.
- [81] J. Robnik and U. Seljak, *Fluctuation without dissipation: microcanonical Langevin Monte Carlo*, [arXiv:2303.18221](#) [[INSPIRE](#)].
- [82] J. Robnik, G.B. De Luca, E. Silverstein and U. Seljak, *Microcanonical Hamiltonian Monte Carlo*, [arXiv:2212.08549](#) [[INSPIRE](#)].
- [83] A.E. Bayer, U. Seljak and C. Modi, *Field-level inference with microcanonical Langevin Monte Carlo*, in the proceedings of the 40th international conference on machine learning, (2023) [[arXiv:2307.09504](#)] [[INSPIRE](#)].
- [84] A. Baleato Lizancos et al., *Selecting samples of galaxies with fewer Fingers-of-God*, *JCAP* **07** (2025) 014 [[arXiv:2501.10587](#)] [[INSPIRE](#)].
- [85] S. Günther et al., *OLÉ — Online Learning Emulation in cosmology*, [arXiv:2503.13183](#) [[INSPIRE](#)].
- [86] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, *Math. Control Signals Syst.* **2** (1989) 303 [[INSPIRE](#)].
- [87] M. Cranmer, *Interpretable machine learning for science with PySR and SymbolicRegression.jl*, [arXiv:2305.01582](#).
- [88] D.J. Bartlett et al., *SYREN-HALOFIT: a fast, interpretable, high-precision formula for the Λ CDM nonlinear matter power spectrum*, *Astron. Astrophys.* **686** (2024) A150 [[arXiv:2402.17492](#)] [[INSPIRE](#)].
- [89] A. Aizpuru, R. Arjona and S. Nesseris, *Machine learning improved fits of the sound horizon at the baryon drag epoch*, *Phys. Rev. D* **104** (2021) 043521 [[arXiv:2106.00428](#)] [[INSPIRE](#)].
- [90] J.B. Orjuela-Quintana, S. Nesseris and D. Sapone, *Machine learning unveils the linear matter power spectrum of modified gravity*, *Phys. Rev. D* **109** (2024) 063511 [[arXiv:2307.03643](#)] [[INSPIRE](#)].
- [91] C. Sui et al., *SYREN-NEW: precise formulae for the linear and nonlinear matter power spectra with massive neutrinos and dynamical dark energy*, *Astron. Astrophys.* **698** (2025) A1 [[arXiv:2410.14623](#)] [[INSPIRE](#)].
- [92] BOSS collaboration, *The Baryon Oscillation Spectroscopic Survey of SDSS-III*, *Astron. J.* **145** (2013) 10 [[arXiv:1208.0022](#)] [[INSPIRE](#)].
- [93] BOSS collaboration, *The clustering of galaxies in the completed SDSS-III Baryon Oscillation Spectroscopic Survey: cosmological analysis of the DR12 galaxy sample*, *Mon. Not. Roy. Astron. Soc.* **470** (2017) 2617 [[arXiv:1607.03155](#)] [[INSPIRE](#)].
- [94] G. D’Amico et al., *The BOSS bispectrum analysis at one loop from the effective field theory of large-scale structure*, *JCAP* **05** (2024) 059 [[arXiv:2206.08327](#)] [[INSPIRE](#)].
- [95] T. Brinckmann and J. Lesgourgues, *MontePython 3: boosted MCMC sampler and other features*, *Phys. Dark Univ.* **24** (2019) 100260 [[arXiv:1804.07261](#)] [[INSPIRE](#)].

- [96] A. Gelman and D.B. Rubin, *Inference from iterative simulation using multiple sequences*, *Statist. Sci.* **7** (1992) 457 [[INSPIRE](#)].
- [97] S. Axen, D. Karrasch, J. Burton, M. Hauru and P. Yong, *mlcolab/pathfinder.jl: v0.9.8*, [Zenodo](#), December 2024.
- [98] M.M. Ivanov et al., *Full-shape analysis with simulation-based priors: constraints on single field inflation from BOSS*, *Phys. Rev. D* **110** (2024) 063538 [[arXiv:2402.13310](#)] [[INSPIRE](#)].
- [99] S. Paradiso et al., *Reducing nuisance prior sensitivity via non-linear reparameterization, with application to EFT analyses of large-scale structure*, *JCAP* **07** (2025) 005 [[arXiv:2412.03503](#)] [[INSPIRE](#)].
- [100] K. Akitsu, *Mapping the galaxy-halo connection to the galaxy bias: implication to the HOD-informed prior*, [arXiv:2410.08998](#) [[INSPIRE](#)].
- [101] M. Shiferaw, N. Kokron and R.H. Wechsler, *How do uncertainties in galaxy formation physics impact field-level galaxy bias?*, *Astrophys. J.* **989** (2025) 218 [[arXiv:2412.06886](#)] [[INSPIRE](#)].
- [102] DESI collaboration, *Enhancing DESI DR1 full-shape analyses using HOD-informed priors*, [arXiv:2504.10407](#) [[INSPIRE](#)].
- [103] DESI collaboration, *Frequentist cosmological constraints from full-shape clustering measurements in DESI DR1*, [arXiv:2508.11811](#) [[INSPIRE](#)].
- [104] S. Chiarenza, M. Bonici, W.J. Percival and M. White, *BLAST: Beyond Limber Angular power Spectra Toolkit. A fast and efficient algorithm for 3x2 pt analysis*, [arXiv:2410.03632](#) [[INSPIRE](#)].
- [105] D.N. Limber, *The analysis of counts of the extragalactic nebulae in terms of a fluctuating density field*, *The Astrophysical Journal* **117** (1953) 134.
- [106] X. Fang, E. Krause, T. Eifler and N. MacCrann, *Beyond Limber: efficient computation of angular power spectra for galaxy clustering and weak lensing*, *JCAP* **05** (2020) 010 [[arXiv:1911.11947](#)] [[INSPIRE](#)].
- [107] S.-F. Chen, Z. Vlah, E. Castorina and M. White, *Redshift-space distortions in Lagrangian perturbation theory*, *JCAP* **03** (2021) 100 [[arXiv:2012.04636](#)] [[INSPIRE](#)].
- [108] A. Chudaykin, M.M. Ivanov, O.H.E. Philcox and M. Simonović, *Nonlinear perturbation theory extension of the Boltzmann code CLASS*, *Phys. Rev. D* **102** (2020) 063533 [[arXiv:2004.10607](#)] [[INSPIRE](#)].
- [109] H.E. Noriega, A. Aviles, S. Fromenteau and M. Vargas-Magaña, *Fast computation of non-linear power spectrum in cosmologies with massive neutrinos*, *JCAP* **11** (2022) 038 [[arXiv:2208.02791](#)] [[INSPIRE](#)].
- [110] D. Linde et al., *CLASS-OneLoop: accurate and unbiased inference from spectroscopic galaxy surveys*, *JCAP* **07** (2024) 068 [[arXiv:2402.09778](#)] [[INSPIRE](#)].
- [111] P. Carrilho, C. Moretti and A. Pourtsidou, *Cosmology with the EFTofLSS and BOSS: dark energy constraints and a note on priors*, *JCAP* **01** (2023) 028 [[arXiv:2207.14784](#)] [[INSPIRE](#)].
- [112] L. Piga et al., *Constraints on modified gravity from the BOSS galaxy survey*, *JCAP* **04** (2023) 038 [[arXiv:2211.12523](#)] [[INSPIRE](#)].
- [113] M. Archidiacono, E. Castorina, D. Redigolo and E. Salvioni, *Unveiling dark fifth forces with linear cosmology*, *JCAP* **10** (2022) 074 [[arXiv:2204.08484](#)] [[INSPIRE](#)].
- [114] S. Bottaro et al., *From 100 kpc to 10 Gpc: dark matter self-interactions before and after DESI observations*, *Phys. Rev. D* **112** (2025) 023525 [[arXiv:2407.18252](#)] [[INSPIRE](#)].

- [115] A. Perko, L. Senatore, E. Jennings and R.H. Wechsler, *Biased tracers in redshift space in the EFT of large-scale structure*, [arXiv:1610.09321](#) [INSPIRE].
- [116] G. D’Amico et al., *The cosmological analysis of the SDSS/BOSS data from the effective field theory of large-scale structure*, *JCAP* **05** (2020) 005 [[arXiv:1909.05271](#)] [INSPIRE].
- [117] G. D’Amico, L. Senatore, P. Zhang and T. Nishimichi, *Taming redshift-space distortion effects in the EFTofLSS and its application to data*, *JCAP* **01** (2024) 037 [[arXiv:2110.00016](#)] [INSPIRE].
- [118] M.M. Ivanov et al., *Cosmological constraints without nonlinear redshift-space distortions*, *Phys. Rev. D* **105** (2022) 043531 [[arXiv:2110.00006](#)] [INSPIRE].
- [119] L. Senatore and M. Zaldarriaga, *Redshift space distortions in the effective field theory of large scale structures*, [arXiv:1409.1225](#) [INSPIRE].
- [120] M. Lewandowski and L. Senatore, *An analytic implementation of the IR-resummation for the BAO peak*, *JCAP* **03** (2020) 018 [[arXiv:1810.11855](#)] [INSPIRE].