

Hough, R.T., Rugg, R.G., Sahlu, S. & Abebe, A. (2026). Kosmulator: a Python framework for cosmological inference with MCMC. In Abebe, A. & Paliathanasis, A. (eds), *Proceedings of the South African Gravity Society Conference 2025 (SAGS2025)*. Potchefstroom: Axiom Academic Publishers, pp. 7 - 19.

Kosmulator: a Python framework for cosmological inference with MCMC

Renier T. Hough, Robert G. Rugg, Shambel Sahlu and Amare Abebe

Abstract

We present `Kosmulator`, a modular and vectorised Python framework designed to accelerate the statistical testing of cosmological models. As the theoretical landscape expands beyond standard ΛCDM, implementing new expansion histories into traditional Einstein–Boltzmann solvers becomes a significant computational bottleneck. `Kosmulator` addresses this by leveraging array-native execution and efficient ensemble slice sampling (via `Zeus`) to perform rapid Bayesian inference. We validate the framework against the industry-standard `Cobaya` code using a combination of Type Ia Supernovae, Cosmic Chronometers, and Baryon Acoustic Oscillation (BAO) data. Our results demonstrate that `Kosmulator` reproduces `Cobaya`'s posterior constraints to within $\leq 0.3\sigma$ statistical agreement on H_0 and Ω_m and $< 0.6\%$ precision on χ^2, while achieving a $\sim 4.5\times$ reduction in wall-clock time on a single CPU core compared to a standard MPI-parallelised baseline. Furthermore, we showcase the framework's utility by constraining the implicit power-law $f(Q)$ "f_1CDM" model and demonstrating its automated model selection capabilities (AIC/BIC). `Kosmulator` is introduced as a "scientific sieve" for rapid hypothesis testing, allowing researchers to efficiently filter theoretical candidates before deploying high-precision resources.

Keywords: Cosmological inference; MCMC; Bayesian statistics; Python; $f(Q)$ gravity

Kosmulator: A Python framework for cosmological inference with MCMC

Renier T. Hough[1, 2], Robert G. Rugg[1], Shambel Sahlu[1, 2], and Amare Abebe[1, 2]

[1]Centre for Space Research, North-West University, Potchefstroom, South Africa, 2520
[2]National Institute for Theoretical and Computational Sciences (NITheCS), Potchefstroom, South Africa, 2520
E-mail: renierht@gmail.com

Abstract

We present `Kosmulator`, a modular and vectorised Python framework designed to accelerate the statistical testing of cosmological models. As the theoretical landscape expands beyond standard ΛCDM, implementing new expansion histories into traditional Einstein–Boltzmann solvers becomes a significant computational bottleneck. `Kosmulator` addresses this by leveraging array-native execution and efficient ensemble slice sampling (via `Zeus`) to perform rapid Bayesian inference. We validate the framework against the industry-standard `Cobaya` code using a combination of Type Ia Supernovae, Cosmic Chronometers, and Baryon Acoustic Oscillation (BAO) data. Our results demonstrate that `Kosmulator` reproduces `Cobaya`'s posterior constraints to within $\leq 0.3\sigma$ statistical agreement on H_0 and Ω_m and $< 0.6\%$ precision on χ^2, while achieving a $\sim 4.5\times$ reduction in wall-clock time on a single CPU core compared to a standard MPI-parallelised baseline. Furthermore, we showcase the framework's utility by constraining the implicit power-law $f(Q)$ "f_1CDM" model and demonstrating its automated model selection capabilities (AIC/BIC). `Kosmulator` is introduced as a "scientific sieve" for rapid hypothesis testing, allowing researchers to efficiently filter theoretical candidates before deploying high-precision resources.

1 Introduction

The standard ΛCDM model of cosmology has been remarkably successful in explaining a wide range of observations, from the Cosmic Microwave Background (CMB) to the large-scale structure of the Universe. However, the emergence of statistical tensions, most notably the Hubble tension (H_0) and the growth of structure tension (S_8), has spurred a resurgence of interest in alternative cosmological models and modified theories of gravity [1–8].

Theoretical physicists are currently exploring a vast landscape of potential solutions, ranging from early dark energy [9] and interacting dark sectors [10, 11] to modifications of General Relativity such as $f(R)$ [12], $f(Q)$ [13] and $f(T)$ [14] gravity. However, a significant computational bottleneck exists: testing these theoretical models against observational data typically requires their implementation into full Einstein–Boltzmann solvers such as `CLASS`[1] [15] or `CAMB`[2] [16], to be utilised by frameworks like `MontePython`[3] [17] and `Cobaya`[4] [18], which are the industry standards for precision inference. These frameworks possess steep learning curves and adapting them for non-standard models often requires modifications in C and Fortran, which can be time-consuming for preliminary model screening.

This technical barrier often slows the "hypothesis-to-test" cycle. Researchers may spend months implementing a model only to find it is ruled out by fundamental background observables. Therefore, there is a need for a lightweight, modular, and rapid inference framework that operates primarily at the background and phenomenological level.

In this paper, we introduce the core philosophy and modular architecture of `Kosmulator`[5], while providing a proof-of-concept demonstration of its utility for rapid model prototyping. Serving as a "scientific sieve," the

[1]`CLASS` available at `https://github.com/lesgourg/class_public.git`
[2]`CAMB` available at `https://camb.readthedocs.io/en/latest/`
[3]`MontePython` available at `https://github.com/brinckmann/montepython_public`
[4]`Cobaya` available at `https://cobaya.readthedocs.io/en/latest/index.html`
[5]`Kosmulator` available at `https://github.com/renierht/Kosmulator`

framework is designed to allow researchers to quickly test non-standard Friedmann equations and growth parameterisations using vectorised likelihoods before committing to more computationally intensive pipelines. We outline the fundamental usage of the package and present a baseline consistency check against the established `Cobaya` framework [18] to demonstrate the reliability of our likelihood implementation for standard cosmologies. While a rigorous performance benchmark suite and high-dimensional parameter analysis are reserved for a forthcoming publication, this work establishes the framework's baseline viability for the community.

2 Kosmulator

In this section, we describe the technical architecture of the `Kosmulator` MCMC package. Unlike standard Einstein–Boltzmann pipelines, which prioritize full perturbation-level accuracy at the cost of implementation complexity, `Kosmulator` is designed as a streamlined, pure-Python framework. Its primary objective is the rapid numerical evaluation of expansion histories and growth rates, even for models where the Friedmann equations are not analytically solvable.

The package is built on a modular architecture, enabling the direct comparison of theoretical predictions against late-time observational probes with minimal implementation effort. Building upon the foundational logic described in [19], this current version introduces a highly flexible sampling layer and streamlined dataset integration to facilitate broader community use.

`Kosmulator` utilises both the `EMCEE`[6] [20] and `Zeus`[7] [21] samplers as interchangeable sampling engines. By providing access to both the Affine Invariant Ensemble Sampler (`EMCEE`) and the Ensemble Slice Sampler (`Zeus`), users can select the algorithm best suited to the specific topology of their model's parameter space. The latter, in particular, often provides superior performance for the highly non-linear or degenerate posteriors frequently encountered in modified gravity. For detailed discussions on the underlying methodologies, we refer the reader to [20, 21].

2.1 Features and current capabilities

`Kosmulator` contains the standard features found in competing software, such as:

- **Multiple MCMC Backends** — Support for both `Zeus` and `EMCEE`, allowing for automatic or user-controlled sampler selection.
- **Chain Reuse and Resume Capability** — Reload, resume, or extend existing MCMC chains for reproducibility and efficient experimentation.
- **Publication-Ready Statistical Outputs and Visualisation** — Automatic generation of corner plots, best-fit comparisons, autocorrelation diagnostics, and LaTeX-ready tables. Kosmulator's plotting pipeline utilises the `GetDist` package for post-processing and visualisation.
- **Automatic Parameter Injection and Dataset Awareness** — Parameters (including nuisance parameters) are automatically added, fixed, or conditioned based on the selected observational datasets.
- **Support for a Wide Range of Observational Data**:
  - Type Ia Supernovae (JLA, Pantheon, Pantheon+, Union3, DES-Y5) [22–28]
  - Baryon Acoustic Oscillations (BAO) [29]
  - DESI (Data Release 1 and 2) [8, 30]
  - Cosmic Chronometers (CC) / Observational Hubble Distance (OHD) [31–36]
  - Cosmic Microwave Background (CMB-plik; Planck 2018 low-ℓ, high-ℓ TT, combined TTTEEE, and CMB lensing when used jointly with other datasets via `CLIK`[8]/`CLASS`) [1, 15]
  - Big Bang Nucleosynthesis (D/H, `PRyMordial`-based BBN, and AlterBBN[9]-based likelihoods) [8, 37]

While these functionalities are shared with other inference frameworks, such as `Cobaya`, which offers a broad suite of engines including nested samplers, and `MontePython`, which is tightly coupled with the `CLASS` code, `Kosmulator` distinguishes itself by matching these capabilities through a more streamlined, automated parameter-handling architecture.

[6] `EMCEE` available at `https://emcee.readthedocs.io/en/stable/`.
[7] `Zeus` available at `https://zeus-mcmc.readthedocs.io/en/latest/index.html`.
[8] `CLIK` available at `http://pla.esac.esa.int/pla/`.
[9] `AlterBBN` available at `https://github.com/espensem/AlterBBN.git`.

2.1.1 Non-analytically solvable Friedmann equations

Due to `Kosmulator`'s fully modular model architecture, any cosmological model that can be expressed as a standard `Python` function can be numerically solved and fitted to observational data. This design is particularly useful for cosmological models with implicit Friedmann equations. While inference frameworks such as `Cobaya` and `MontePython` can in principle be extended to support such models through modifications of external Einstein–Boltzmann solvers, doing so typically requires non-trivial development effort. `Kosmulator`, by contrast, was explicitly developed with numerical background-level model flexibility in mind.

In `Kosmulator`, adding a new cosmological model is as simple as defining a function of the form:

```
def LCDM_MODEL(z: Number, p: Dict[str, float]) -> Number:
    """Flat ΛCDM: E(z) = sqrt(Ωm(1+z)^3 + 1-Ωm)."""
    z = np.asarray(z, dtype=float)
    Om = float(p["Omega_m"])
    E2 = Om * (1.0 + z)**3 + (1.0 - Om)
    return np.where(E2 > 0, np.sqrt(E2), np.nan)
```

and registering it within the `User_defined_modules.py` script. Although this example corresponds to the standard ΛCDM model (already included in `Kosmulator`), the same structure applies to any background expansion history $E(z)$.

For models lacking closed-form expansion histories, users can simply include an appropriate numerical solver within the model function. For example, in [38] — the first scientific application of the open-sourced code — we considered a Power-law $f(Q)$ model [39] defined implicitly by :

$$E(z)^2 = \Omega_m(1+z)^3 + (1-\Omega_m)E^{2n}, \tag{1}$$

which admits no closed-form solution for $n \neq 0$. Its implementation in `Kosmulator` is straightforward:

```
from scipy.optimize import fsolve
def f1CDM_MODEL(z: Number, p: Dict[str, float]) -> Number:
    """f1CDM implicit model: E^2 = Om(1+z)^3 + (1-Om) E^(2n)
    Solved per redshift using a scalar root finder."""
    Om = float(p["Omega_m"])
    n  = float(p["n"])
    z_arr = _asarray(z)
    out = np.empty_like(z_arr, dtype=float)

    for i, zi in enumerate(z_arr):
        def eq(E):
            return E**2 - (Om*(1.0+zi)**3 + (1.0-Om)*E**(2.0*n))

        E0 = np.sqrt(Om*(1.0+zi)**3 + (1.0-Om))  # ΛCDM seed
        try:
            sol = fsolve(eq, x0=E0, xtol=1e-8, maxfev=100)[0]
            out[i] = sol if np.isfinite(sol) and sol > 0.0 else np.nan
        except Exception:
            out[i] = np.nan

    return _scalar_or_array(out)
```

As illustrated, the overall structure of the example function remains unchanged; the user simply introduces a suitable numerical root-finding procedure. It should be noted that due to limitations of the iterative solvers, care must be taken to select a solver appropriate to the model in order to avoid numerical instabilities, and to include exception handling for robustness.

This approach applies to all background and late-time observational probes supported by `Kosmulator`. The primary exception is CMB analysis, which requires perturbation-level equations to be implemented within `CLASS`, via a Taylor expansion [40], to compute theoretical power spectra; for such cases, we implemented an interface within `Kosmulator` to communicate directly with `CLASS` and `CLIK` for the specified model (briefly discussed in Section 2.1.5).

2.1.2 Vectorised models and observation pipelines

A key performance-oriented feature of `Kosmulator` is support for vectorised model and observation evaluations. In this context, vectorisation refers to evaluating both background cosmological quantities (most notably the ex-

pansion rate $E(z)$) and the corresponding observation-model predictions (e.g. $H(z)$, distance moduli, and growth observables) for an entire array of redshifts or data points in a single function call, using array-based arithmetic rather than explicit Python loops.

To illustrate the idea intuitively, consider a simple function such as $f(x) = 3x-2$. When sketching this function by hand, one does not explicitly compute $f(x)$ for each individual value of x; instead, the functional form is applied simultaneously across the domain, i.e., a straight line going through $f(0) = -2$ with a gradient of $+3$. In contrast, many numerical implementations evaluate functions point-by-point, looping over each input value and storing the result in an array. While straightforward, such scalar evaluation introduces significant overhead which becomes increasingly computationally expensive when repeated many times during an MCMC sampling process.

Vectorised implementations instead apply the same mathematical operations simultaneously to all input values, delegating the computation to optimised numerical backends such as `NumPy`. This substantially reduces Python-level overhead and improves cache efficiency. In `Kosmulator`, vectorisation extends beyond the background expansion history itself. When a model is flagged as vectorised, the full observation pipeline, including distance calculations, Hubble-rate evaluations, and phenomenological growth predictions, is evaluated in an array-native manner within the `Zeus` ensemble likelihood function. As a result, a single model evaluation produces the complete vector of theoretical predictions required for a given dataset, rather than looping over individual data points.

For analytically solvable models, vectorisation is typically straightforward. For non-analytically solvable or implicitly defined models, vectorisation can still be achieved by implementing array-based numerical solvers. As an example, the implicit $f(Q)$-inspired model defined by Eq. (1) can be implemented in a fully vectorised form by replacing a scalar root-finding routine with an array-based Newton–Raphson scheme. An example implementation is shown below:

```
def f1CDM_MODEL_vectorised(z: Number, p: Dict[str, float]) -> Number:
    """Vectorised f1CDM solver using Newton-Raphson."""
    z = np.asarray(z)
    Om, n = float(p["Omega_m"]), float(p["n"])

    # Seed with LCDM solution (vectorised)
    E = np.sqrt(Om * (1 + z)**3 + (1 - Om))

    # Newton-Raphson iteration on full array
    for _ in range(60):
        f = E**2 - (Om * (1 + z)**3 + (1 - Om) * E**(2.0*n))
        df = 2.0 * E - (1.0 - Om) * (2.0*n) * E**(2.0*n - 1.0)

        step = f / df
        E -= step

        if np.all(np.abs(step) < 1e-8): break

    return E
```

As shown, the overall structure of the model function remains unchanged; the primary difference lies in replacing a scalar solver with an array-based iterative scheme and introducing appropriate safeguards for numerical stability. This allows even implicitly defined Friedmann equations to benefit from vectorised execution.

To ensure predictable and reproducible performance, vectorised model evaluations in `Kosmulator` are executed in a strictly single-process, single-thread configuration. Internal BLAS, OpenMP, and related numerical backends are constrained to a single thread, avoiding hidden parallelism, thread oversubscription, and non-deterministic scaling. The resulting performance gains arise primarily from reduced Python overhead rather than implicit multi-threading.

Certain datasets, most notably CMB and BBN likelihoods, necessarily rely on external solvers such as `CLASS`, `CLIK`, or `AlterBBN`, and therefore remain non-vectorised in the current implementation.

The computational benefit of vectorisation is most pronounced for moderate walker counts and late-time datasets. For very large walker numbers or environments with abundant computational resources, parallel execution may become more efficient.

2.1.3 Built-in information criteria and model comparison

`Kosmulator` includes built-in support for standard information-criterion-based model comparison. In particular, users may designate a *reference model* against which alternative or modified cosmological models are evaluated.

In practice, this reference model is often chosen to be ΛCDM, reflecting its status as the current concordance cosmology, although other baseline models, such as the w_0w_aCDM model employed in recent DESI analyses [8], may also be used.

Given a specified baseline, `Kosmulator` automatically computes a set of widely adopted diagnostics for each candidate model, including the best-fit likelihood, reduced χ^2, the Akaike Information Criterion (AIC) [41], and the Bayesian Information Criterion (BIC) [42]. Differences in these metrics (ΔAIC and ΔBIC) are evaluated relative to the reference model to provide a clear indication of relative performance while penalising unnecessary model complexity.

`Kosmulator` reports these statistics in a consistent, publication-ready format and uses them to highlight potential issues such as overfitting or lack of statistical support for additional model parameters. The intent is not to replace a full Bayesian model-comparison analysis, but rather to provide immediate, interpretable guidance during exploratory model development.

At present, the statistical analysis is limited to these standard information-criterion-based comparisons [41, 42]. Future development plans include the possible incorporation of additional diagnostics such as the Deviance Information Criterion (DIC) [43], the Watanabe–Akaike Information Criterion (WAIC) [44], Leave-One-Out Cross-Validation (LOO-CV) [45], Bayesian evidence and Bayes factors [46], small-sample-corrected variants such as AICc [47], the Hannan–Quinn Information Criterion (HQIC) [48], and focused information criteria (FIC) [49], subject to feasibility and computational cost.

2.1.4 Structure growth

Kosmulator provides optional likelihood support for phenomenological structure-growth observables, specifically the growth rate $f(z)$ and redshift-space distortion observable $f_{\sigma_8}(z)$ [50, 51]. In this approach, the growth of linear matter perturbations is modelled using the growth-index parameterisation

$$f(z) \equiv \frac{d\ln D}{d\ln a} = \Omega_m^{\gamma}(z), \tag{2}$$

where $D(a)$ is the linear growth factor, Ω_m is derived from the background expansion history $E(z)$, and γ is treated as either a free phenomenological parameter or fixed to a GR-motivated value.

Predictions for $f_{\sigma_8}(z)$ are obtained by combining this parameterisation with the evolution of the growth factor,

$$f_{\sigma_8}(z) = \sigma_8\Omega_m(z)^{\gamma}\exp\left[-\int_0^z \frac{\Omega_m(z')^{\gamma}}{1+z'}dz'\right], \tag{3}$$

allowing direct comparison with observational growth measurements without requiring perturbation-level calculations. In [38], the structure growth observations were added to `Kosmulator`.This framework intentionally decouples clustering inference from Boltzmann solvers, relying only on the background expansion history $E(z)$. As a result, it enables exploration of how matter clusters in deviations from General Relativity. However, a critical distinction remains: this prescription assumes scale-independent growth, effectively isolating the influence of the background geometry on structure formation. It is, therefore, a "geometric consistency check" rather than a full perturbation analysis. It cannot, for instance, capture the scale-dependent suppression or enhancement of power often found in screened modified gravity theories (e.g., Chameleon mechanisms in $f(R)$ [52]) without further modification. Consequently, this module should be viewed as a high-speed diagnostic tool—a means to flag potential growth tensions before committing to computationally expensive, scale-dependent perturbation solvers.

2.1.5 CMB Integration

While `Kosmulator` focuses on modular Pythonic implementations for background cosmology, it supports early-universe probes via a wrapper for the `CLASS` Boltzmann solver [15]. The package automatically manages the interface with the CMB likelihood, handling parameter mapping and nuisance parameters for Planck 2018 likelihoods (TT, TE, EE, and lensing) transparently. This allows researchers to combine custom background models with standard CMB constraints without modifying any underlying C-code. A more detailed exploration of this integration will be presented in a forthcoming research article.

3 Validation and Performance Showcase

To verify the accuracy of our likelihood implementations, the runs in Table 1 compare `Kosmulator` against a `Cobaya` ΛCDM background only baseline[10]. We contrast the industry-standard Metropolis-Hastings sampler (via `Cobaya`) with the `Zeus` ensemble slice sampler (via `Kosmulator`). `Zeus` is chosen specifically to demonstrate the efficacy of the vectorised likelihood: because the vectorised approach is built to run on a single CPU core to maximise array efficiency, it allows us to quantify the speed recovered by avoiding standard Python loops and MPI overhead[11].

The observational suite, consisting of *Pantheon+SH0ES*, *DESI DR2* BAO, and *CC*, was selected to provide a robust anchor for the expansion history. *Pantheon+SH0ES* provides a calibrated distance ladder that is necessary for constraining H_0 and the absolute magnitude $M_{\rm abs}$. The *DESI DR2* BAO measurements provide tight constraints on the transverse and radial distance scales [8]. Finally, *CC* data were included because they provide direct, model-independent measurements of $H(z)$. This helps break degeneracies among parameters such as $M_{\rm abs}$, H_0, and r_d, including the well-known H_0–r_d degeneracy that can arise when fitting the ΛCDM model directly to BAO-related datasets [53, 54].

Crucially, the runtimes reported correspond to the wall-clock time required to reach statistical convergence, rather than a fixed number of steps. For `Cobaya`, we utilised the standard Gelman-Rubin ($R-1$) stopping criterion on a background-only cosmology. For `Kosmulator`, we configured the samplers with a relaxed iteration cap, but enabled dynamic termination based on the integrated autocorrelation time (τ). Lastly, we added a run using the non-analytical f_1CDM model to show that the internal numerical solvers can handle models without closed-form expansion histories without a significant performance penalty.

It is worth noting that a direct comparison of iterations per second (it/s) between Metropolis-Hastings and Ensemble Slice Sampling is conservative. Slice sampling typically requires significantly more likelihood evaluations per "step" to guarantee detailed balance; however, it often produces samples with a lower autocorrelation time (higher effective sample size) than Metropolis chains. Thus, the raw wall-clock speed-up shown in Table 1 may understate the practical sampling advantage, since the relevant comparison depends not only on iterations per second but also on autocorrelation time and effective sample size.

3.1 Discussion of Results

The numerical validation is visualised in Figure 1. The left panel displays the constraints from the joint *Pantheon+SH0ES* and *CC* likelihood combination. The contours from `Kosmulator` (Blue) and `Cobaya` (Red) are virtually indistinguishable, validating the precision of the background evolution solvers. As shown in Table 1, the best-fit χ^2 values agree to within $\leq 0.6\%$, and the derived constraints on H_0 and Ω_m are consistent to within 0.3σ.

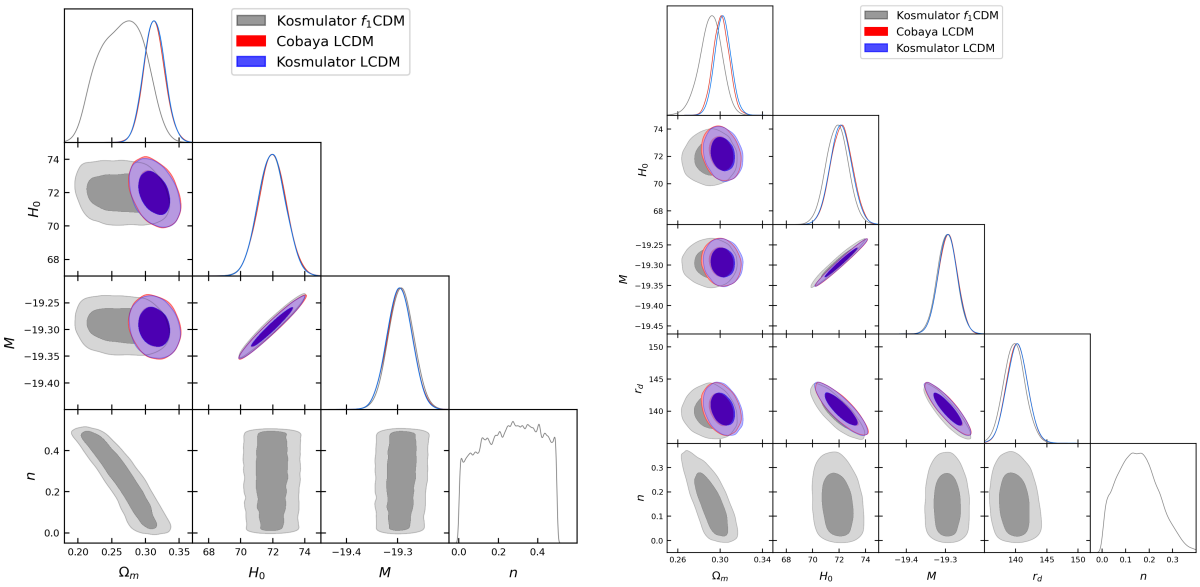
The performance differential is substantial. For the ΛCDM background run, `Kosmulator` achieved convergence in just **2 minutes and 9 seconds using a single CPU core**. In contrast, the `Cobaya` baseline required **9 minutes and 44 seconds distributed across 12 cores**. This represents a wall-clock speed-up of $\sim 4.5\times$ while using only one twelfth of the core count. This efficiency gain is critical for future surveys, where the volume of data will require millions of likelihood evaluations that would otherwise be computationally prohibitive.

The right panel of Figure 1 and the second section of Table 1 introduce the *DESI DR2* BAO data. Notably, even with the addition of the r_d parameter (which `Kosmulator` samples directly as a free parameter (for this particular case), whereas standard `Cobaya` runs often treat it as a derived parameter in this configuration), the agreement remains robust with a similar time speed up. The f_1CDM model runs further demonstrate that expanding the parameter space to include the additional power-law parameter n incurs only a modest runtime penalty for this model and dataset combination, validating the solver's stability in a non-standard cosmological scenario. [12].

[10]Note: `Cobaya` was not used for the non-analytical f_1CDM model, since this run is included only as an illustrative example of how `Kosmulator` can fit models with implicit Friedmann equations.

[11]While `EMCEE` is supported, the current vectorised likelihood implementation is optimised for the `Zeus` sampler interface. `EMCEE` can still fit non-analytical models, but typically relies on MPI parallelisation across multiple cores rather than single-core vectorisation.

[12]It should be noted that these comparison runs utilise baseline configurations. A more comprehensive performance analysis covering high-dimensional parameter spaces will be presented in the forthcoming article.



(a) Pantheon$^+$ + SH0ES + CC

(b) Pantheon$^+$ + SH0ES + DESI DR2 + CC

Figure 1: Corner plots comparing the posterior distributions obtained from `Kosmulator` (Blue/Black) and `Cobaya` (Red). **Left:** Results for the Pantheon$^+$ + SH0ES + CC combination. **Right:** Results including DESI DR2 BAO data, showing the tightening of constraints and the robust agreement between pipelines.

3.2 Automated Model Selection Features

A key design philosophy of `Kosmulator` is to streamline the interpretation of results. Beyond standard parameter constraints, the pipeline natively computes model selection statistics (specifically the AIC and BIC criteria) and their relative differences (Δ) between competing models.

The software outputs an automated natural-language interpretation of these statistics using standard rule-of-thumb thresholds for information-criterion differences (which can be found in the Statistical Analysis Tables folder). This helps users quickly assess whether a new model is statistically supported. Table 2 presents the direct output of this module for the f_1CDM vs ΛCDM comparison. While the f_1CDM model achieves a slightly lower χ^2, the `Kosmulator` report correctly identifies that the additional complexity is penalised by the BIC, providing "moderate evidence against" the extended model for the combined datasets. This automation removes a common manual step in the theoretical workflow, allowing researchers to assess model viability instantly.

4 Discussion: The "Fail-Fast" Philosophy

`Kosmulator` is intended to complement, rather than replace, high-precision frameworks like `Cobaya`. Its primary value lies in the exploratory phase of model building—a "Fail-Fast" prototyping stage.

By using a pure-Python environment with vectorised likelihoods, the "hypothesis-to-test" cycle is reduced from weeks of C-level modifications to minutes of Python scripting. If a proposed modified gravity model cannot achieve a statistically competitive fit at the background or phenomenological growth level using `Kosmulator's` fast check, it is unlikely to justify the significant development effort required for a full perturbation-level analysis needed in `Cobaya/CLASS`, unless they possess a compelling theoretical motivation, a distinctive perturbative signature, or a clear mechanism not captured by the fast-check approximation.

4.1 Utility and Practicality

- **High-Throughput Screening:** The ability to run full MCMC chains for background-only models in minutes on standard hardware allows for the rapid elimination of non-viable theories.
- **Modularity:** Because the codebase avoids complex dependency chains and recompilation, it is well-suited for collaborative projects where different team members may need to inject custom phenomenological pa-

Table 1: Baseline consistency and performance showcase. The f_1CDM model introduces n as an additional degree of freedom. Comparison highlights the efficiency of the single-core `Kosmulator` pipeline against the MPI-parallelised `Cobaya` baseline.

Model	**Pipeline**	**Sampler**	**Cores**	**Time**	**Iter.**	**It/s**	χ^2	
	Constraints	H_0	Ω_m	r_d	M_{abs}	$\Omega_b h^2$	$\Omega_{\rm CDM} h^2$	n
				Dataset: Pantheon$^+$ + SH0ES + CC (Background Only)				
ΛCDM	**Cobaya**	Metropolis	12	9m 44s	105,717	181.02	1486.84	
	Results	$71.986^{+0.861}_{-0.861}$	$^\dagger 0.313^{+0.016}_{-0.016}$	—	$-19.296^{+0.025}_{-0.025}$	$0.0199^{+0.0058}_{-0.0058}$	$0.142^{+0.010}_{-0.010}$	—
ΛCDM	**Kosmulator**	Zeus	1	2m 09s	57,600	443.88	1478.46	
	Results	$71.955^{+0.856}_{-0.860}$	$0.313^{+0.016}_{-0.016}$	—	$-19.296^{+0.024}_{-0.024}$	—	—	—
f_1CDM	**Kosmulator**	Zeus	1	3m 39s	64,800	295.89	1477.46	
	Results	$71.955^{+0.868}_{-0.858}$	$0.266^{+0.033}_{-0.036}$	—	$-19.293^{+0.025}_{-0.025}$	—	—	$0.267^{+0.154}_{-0.168}$
				Dataset: Pantheon$^+$ + SH0ES + CC + BAO				
ΛCDM	**Cobaya r_d-calibrated**	Metropolis	12	10m 12s	56,980	93.10	1498.69	
	Results	$72.194^{+0.819}_{-0.819}$	$^\dagger 0.301^{+0.008}_{-0.008}$	$140.254^{+1.669}_{-1.669}$	$-19.293^{+0.024}_{-0.024}$	$0.0265^{+0.0012}_{-0.0012}$	$0.131^{+0.004}_{-0.004}$	—
ΛCDM	**Kosmulator r_d-free**	Zeus	1	2m 10s	61,200	470.77	1489.41	
	Results	$72.144^{+0.821}_{-0.803}$	$0.303^{+0.008}_{-0.008}$	$140.274^{+1.676}_{-1.686}$	$-19.294^{+0.024}_{-0.024}$	—	—	—
f_1CDM	**Kosmulator r_d-free**	Zeus	1	6m 24s	86,400	225.00	1487.42	
	Results	$71.875^{+0.828}_{-0.841}$	$0.292^{+0.010}_{-0.011}$	$139.810^{+1.704}_{-1.715}$	$-19.295^{+0.024}_{-0.024}$	—	—	$0.150^{+0.092}_{-0.086}$

† *Derived from primary sampling of $\Omega_b h^2 + \Omega_{\rm CDM} h^2$.*

Table 2: Automated model selection statistics generated by `Kosmulator`. (ΛCDM) was chosen as the reference model, against which the software interprets the significance of the Δ values for the alternative (f_1CDM) model. Note the contrast: ΔAIC suggests the models are statistically indistinguishable, while the stricter penalty for the extra parameter in ΔBIC reveals a preference for the simpler ΛCDM model.

Model	$\chi^2_{\rm min}$	**AIC**	**ΔAIC**	**BIC**	**ΔBIC**	**Automated Interpretation**
Dataset: Pantheon$^+$ + SH0ES + CC						
ΛCDM (Ref)	1478.46	1484.46	0.0	1500.77	0.0	—
f_1CDM	1477.46	1485.46	+1.00	1507.20	+6.43	AIC: indistinguishable; BIC: moderate against
Dataset: Pantheon$^+$ + SH0ES + CC + DESI						
ΛCDM (Ref)	1489.39	1497.39	0.0	1519.16	0.0	—
f_1CDM	1487.42	1497.42	+ 0.03	1524.63	+5.47	AIC: indistinguishable; BIC: moderate against

rameters or experimental datasets quickly.

- **Limitations as a Feature:** By explicitly focusing on late-time and background observables, `Kosmulator` maintains a small footprint. It serves as the "scientific sieve" that identifies high-potential candidates for subsequent, rigorous investigation using full Einstein–Boltzmann solvers.

5 Conclusion and Outlook

In this work, we have introduced `Kosmulator`, a modular and vectorised Python framework designed to accelerate the "hypothesis-to-test" cycle in theoretical cosmology. As the landscape of modified gravity and dark energy models expands, the computational cost of implementing every new theory into full Einstein–Boltzmann solvers has become a significant bottleneck. `Kosmulator` addresses this by providing a lightweight, pure-Python environment that prioritises implementation speed, modular model definition, and array-native execution.

Our validation results against the industry-standard `Cobaya` framework demonstrate that this approach can reproduce background-level constraints with good numerical consistency. By leveraging single-core vectorisation and dynamic MCMC termination, `Kosmulator` reproduced ΛCDM constraints to within $< 0.3\sigma$ statistical agreement on H_0 and Ω_m, with $\leq 0.6\%$ agreement in the best-fit χ^2 values, while using only a fraction of the computational resources required by the MPI-parallelised baseline. Furthermore, the successful implementation of the implicit f_1CDM model illustrates the framework's capability to handle non-analytical Friedmann equations,

while the integrated automated model selection module based on AIC/BIC allows the statistical viability of such extensions to be assessed immediately alongside the parameter constraints.

The purpose of `Kosmulator` is not to replace high-precision perturbation codes, but to complement them. Its intended role is that of a rapid scientific sieve: models that fail to provide a competitive fit at the background or phenomenological-growth level can be identified quickly, before substantial effort is invested in full perturbation-level implementations. In this sense, the framework helps focus computational and development resources on the most promising theoretical candidates.

Looking ahead, the long-term vision for `Kosmulator` is to bridge the gap between rapid exploratory prototyping and full-scale precision cosmology. A key future direction is the possible implementation of a natively vectorised Einstein–Boltzmann solver within the framework. By extending the same vectorisation principles currently used for background evolution to the linear perturbation equations, such a development would aim to combine the physical accuracy of established precision codes with the speed advantages of array-native execution.

If feasible, this would allow `Kosmulator` to treat high-dimensional and early-universe parameter spaces with an efficiency closer to that currently achieved for late-time and background observables. Such a capability would move the framework beyond its present role as a rapid screening tool and toward a single high-performance pipeline supporting the full research cycle, from initial hypothesis testing to final precision constraints. However, this represents a significant computational and theoretical undertaking, and its feasibility remains an open question. It should therefore be regarded as a long-term development goal rather than a capability of the current version.

The code is open-source and available to the community at `https://github.com/renierht/Kosmulator`, offering a flexible testbed for the next generation of cosmological theories.

Acknowledgements

The authors acknowledge the limited use of AI-assisted writing tools, including `ChatGPT` and `Gemini`, during manuscript preparation for grammar correction, language editing, and clarity improvements. All text produced or revised with assistance from these tools was checked by the authors for scientific accuracy, technical correctness, and consistency with the results presented in this work. The authors reviewed, edited, and approved the final manuscript and take full responsibility for its content, interpretation, and wording.

References

[1] Planck Collaboration, N. Aghanim, Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. J. Banday, et al. Planck 2018 results. VI. Cosmological parameters. *Astron. Astrophys.*, 641:A6, September 2020.

[2] K. C. Wong, S. H. Suyu, G. C.-F. Chen, C. E. Rusu, M. Millon, D. Sluse, V. Bonvin, C. D. Fassnacht, et al. H0LiCOW - XIII. A 2.4 per cent measurement of H_0 from lensed quasars: 5.3σ tension between early- and late-Universe probes. *Mon. Not. R. Astron. Soc.*, 498(1):1420–1439, October 2020.

[3] M. Asgari, C.-A. Lin, B. Joachimi, B. Giblin, C. Heymans, H. Hildebrandt, A. Kannawadi, B. Stölzner, et al. KiDS-1000 cosmology: Cosmic shear constraints and comparison between two point statistics. *Astron. Astrophys.*, 645:A104, January 2021.

[4] E. Valentino, O. Mena, S. Pan, L. Visinelli, W. Yang, A. Melchiorri, D. F. Mota, A. G. Riess, et al. In the realm of the Hubble tension-a review of solutions. *Class. Quantum Grav.*, 38(15):153001, July 2021.

[5] T. M. C. Abbott, M. Aguena, A. Alarcon, S. Allam, O. Alves, A. Amon, F. Andrade-Oliveira, J. Annis, et al. Dark Energy Survey Year 3 results: Cosmological constraints from galaxy clustering and weak lensing. *Phys. Rev. D*, 105(2):023520, January 2022.

[6] E. Abdalla, G. F. Abellán, A. Aboubrahim, A. Agnello, Ö. Akarsu, Y. Akrami, G. Alestas, D. Aloni, et al. Cosmology intertwined: A review of the particle physics, astrophysics, and cosmology associated with the cosmological tensions and anomalies. *J. High Energy Astrophys.*, 34:49–211, June 2022.

[7] A. G. Riess, W. Yuan, L. M. Macri, D. Scolnic, D. Brout, S. Casertano, D. O. Jones, Y. Murakami, et al. A Comprehensive Measurement of the Local Value of the Hubble Constant with $1\,\mathrm{km\,s^{-1}\,Mpc^{-1}}$ Uncertainty from the Hubble Space Telescope and the SH0ES Team. *Astrophys. J. Lett.*, 934(1):L7, July 2022.

[8] M. Abdul Karim, J. Aguilar, S. Ahlen, S. Alam, L. Allen, C. A. Prieto, O. Alves, A. Anand, et al. DESI DR2 results. II. Measurements of baryon acoustic oscillations and cosmological constraints. *Phys. Rev. D*, 112(8):083515, October 2025.

[9] V. Poulin, T. L. Smith, T. Karwal, and M. Kamionkowski. Early Dark Energy can Resolve the Hubble Tension. *Phys. Rev. Lett.*, 122(22):221301, June 2019.

[10] M. van der Westhuizen, A. Abebe, and E. Di Valentino. I. Linear interacting dark energy: Analytical solutions and theoretical pathologies. *Phys. Dark Universe*, 50:102119, December 2025.

[11] M. van der Westhuizen, A. Abebe, and E. Di Valentino. II. Non-linear interacting dark energy: Analytical solutions and theoretical pathologies. *Phys. Dark Universe*, 50:102120, December 2025.

[12] W. Hu and I. Sawicki. Models of $f(R)$ cosmic acceleration that evade solar system tests. *Phys. Rev. D*, 76(6):064004, September 2007.

[13] J. Beltran Jimenez, L. Heisenberg, and T. Koivisto. Coincident General Relativity. *Phys. Rev. D*, 98(4):044048, August 2018.

[14] G. R. Bengochea and R. Ferraro. Dark torsion as the cosmic speed-up. *Phys. Rev. D*, 79(12):124019, June 2009.

[15] J. Lesgourgues. The Cosmic Linear Anisotropy Solving System (CLASS) I: Overview. *arXiv e-prints*, page arXiv:1104.2932, April 2011.

[16] A. Lewis and A. Challinor. CAMB: Code for Anisotropies in the Microwave Background. Astrophysics Source Code Library, record ascl:1102.026, February 2011.

[17] T. Brinckmann and J. Lesgourgues. MontePython 3: Boosted MCMC sampler and other features. *Phys. Dark Universe*, 24:100260, March 2019.

[18] J. Torrado and A. Lewis. Cobaya: code for Bayesian analysis of hierarchical physical models. *J. Cosmol. Astropart. Phys.*, 2021(5):057, May 2021.

[19] R. T. Hough, A. Abebe, and S. E. S. Ferreira. Viability tests of $f(R)$-gravity models with Supernovae Type 1A data. *Eur. Phys. J. C*, 80(8):787, August 2020.

[20] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman. emcee: The MCMC Hammer. *Publ. Astron. Soc. Pac.*, 125(925):306, March 2013.

[21] M. Karamanis, F. Beutler, and J. A. Peacock. zeus: a PYTHON implementation of ensemble slice sampling for efficient Bayesian parameter inference. *Mon. Not. R. Astron. Soc.*, 508(3):3589–3603, December 2021.

[22] M. Hicken, P. Challis, S. Jha, R. P. Kirshner, T. Matheson, M. Modjaz, A. Rest, W. M. Wood-Vasey, et al. CfA3: 185 Type Ia Supernova Light Curves from the CfA. *Astrophys. J.*, 700(1):331–357, July 2009.

[23] J. D. Neill, M. Sullivan, D. A. Howell, A. Conley, M. Seibert, D. C. Martin, T. A. Barlow, K. Foster, et al. The Local Hosts of Type Ia Supernovae. *Astrophys. J.*, 707(2):1449–1465, December 2009.

[24] A. Conley, J. Guy, M. Sullivan, N. Regnault, P. Astier, C. Balland, S. Basa, R. G. Carlberg, et al. Supernova Constraints and Systematic Uncertainties from the First Three Years of the Supernova Legacy Survey. *Astrophys. J. Supp.*, 192(1):1, January 2011.

[25] D. M. Scolnic, D. O. Jones, A. Rest, Y. C. Pan, R. Chornock, R. J. Foley, M. E. Huber, R. Kessler, et al. The Complete Light-curve Sample of Spectroscopically Confirmed SNe Ia from Pan-STARRS1 and Cosmological Constraints from the Combined Pantheon Sample. *Astrophys. J.*, 859(2):101, June 2018.

[26] D. Brout, D. Scolnic, B. Popovic, A. G. Riess, A. Carr, J. Zuntz, R. Kessler, T. M. Davis, et al. The Pantheon+ Analysis: Cosmological Constraints. *Astrophys. J.*, 938(2):110, October 2022.

[27] D. Rubin, G. Aldering, M. Betoule, A. Fruchter, X. Huang, A. G. Kim, C. Lidman, E. Linder, et al. Union through UNITY: Cosmology with 2000 SNe Using a Unified Bayesian Framework. *Astrophys. J.*, 986(2):231, June 2025.

[28] DES Collaboration, T. M. C. Abbott, M. Acevedo, M. Aguena, A. Alarcon, S. Allam, O. Alves, A. Amon, et al. The Dark Energy Survey: Cosmology Results with ~ 1500 New High-redshift Type Ia Supernovae Using the Full 5 yr Data Set. *Astrophys. J. Lett.*, 973(1):L14, September 2024.

[29] A. G. Adame, J. Aguilar, S. Ahlen, S. Alam, D. M. Alexander, M. Alvarez, O. Alves, A. Anand, et al. DESI 2024 VI: cosmological constraints from the measurements of baryon acoustic oscillations. *J. Cosmol. Astropart. Phys.*, 2025(2):021, February 2025.

[30] K. Lodha, A. Shafieloo, R. Calderon, E. Linder, W. Sohn, J. L. Cervantes-Cota, A. de Mattia, J. García-Bellido, et al. DESI 2024: Constraints on physics-focused aspects of dark energy using DESI DR1 BAO data. *Phys. Rev. D*, 111(2):023532, January 2025.

[31] M. Moresco, R. Jimenez, L. Verde, A. Cimatti, and L. Pozzetti. Setting the Stage for Cosmic Chronometers. II. Impact of Stellar Population Synthesis Models Systematics and Full Covariance Matrix. *Astrophys. J.*, 898(1):82, July 2020.

[32] J.-Z. Qi, P. Meng, J.-F. Zhang, and X. Zhang. Model-independent measurement of cosmic curvature with the latest $H(z)$ and SNe Ia data: A comprehensive investigation. *Phys. Rev. D*, 108(6):063522, September 2023.

[33] S. I. Loubser, A. B. Alabi, M. Hilton, Y.-Z. Ma, X. Tang, N. Hatamkhani, C. Cress, R. E. Skelton, et al. An independent estimate of $H(z)$ at $z = 0.5$ from the stellar ages of brightest cluster galaxies. *Mon. Not. R. Astron. Soc.*, 540(4):3135–3149, July 2025.

[34] S. I. Loubser. Measuring the expansion history of the Universe with DESI cosmic chronometers. *Mon. Not. R. Astron. Soc.*, 544(4):3064–3075, December 2025.

[35] Z.-f. Wang, L. Lei, and Y.-z. Fan. New $H(z)$ Measurement at Redshift = 0.12 with DESI Data Release 1. *Astrophys. J.*, 1003(1):81, May 2026.

[36] G. S. Sharov and V. O. Vasiliev. How predictions of cosmological models depend on Hubble parameter data sets. *Math. Model. Geom.*, 6(1):1–20, April 2018.

[37] R. J. Cooke, M. Pettini, R. A. Jorgenson, M. T. Murphy, and C. C. Steidel. Precision Measures of the Primordial Abundance of Deuterium. *Astrophys. J.*, 781(1):31, January 2014.

[38] S. Sahlu, R. T. Hough, A. Abebe, and Á. de la Cruz-Dombriz. Constraining viscous fluid models in $f(Q)$ gravity with data. *Eur. Phys. J. C*, 85(7):746, July 2024.

[39] S. Sahlu, Á. de la Cruz-Dombriz, and A. Abebe. Structure growth in $f(Q)$ cosmology. *Mon. Not. R. Astron. Soc.*, 539(2):690–703, May 2025.

[40] S. Nesseris, S. Basilakos, E. N. Saridakis, and L. Perivolaropoulos. Viable $f(T)$ models are practically indistinguishable from ΛCDM. *Phys. Rev. D*, 88(10):103010, November 2013.

[41] H. Akaike. A New Look at the Statistical Model Identification. *IEEE Trans. Autom. Control*, 19:716–723, January 1974.

[42] G. Schwarz. Estimating the Dimension of a Model. *Ann. Stat.*, 6(2):461–464, July 1978.

[43] D. J. Spiegelhalter, N. G. Best, B. P. Carlin, and A. Van Der Linde. Bayesian measures of model complexity and fit. *J. R. Stat. Soc. Ser. B*, 64(4):583–639, 2002.

[44] S. Watanabe. Asymptotic Equivalence of Bayes Cross Validation and Widely Applicable Information Criterion in Singular Learning Theory. *J. Mach. Learn. Res.*, 11(116):3571–3594, 2010.

[45] A. Vehtari, A. Gelman, and J. Gabry. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Stat. Comput.*, 27(5):1413–1432, September 2017.

[46] R. E. Kass and A. E. Raftery. Bayes factors. *J. Am. Stat. Assoc.*, 90(430):773–795, 1995.

[47] C. M. Hurvich and C.-l. Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 06 1989.

[48] E. J. Hannan and B. G. Quinn. The determination of the order of an autoregression. *J. R. Stat. Soc. Ser. B*, 41(2):190–195, 1979.

[49] G. Claeskens and N. L. Hjort. The focused information criterion. *J. Am. Stat. Assoc.*, 98(464):900–916, 2003.

[50] L. Kazantzidis and L. Perivolaropoulos. Evolution of the $f\sigma_8$ tension with the Planck15/ΛCDM determination and implications for modified gravity theories. *Phys. Rev. D*, 97(10):103503, May 2018.

[51] L. Perenon, J. Bel, R. Maartens, and Á. de la Cruz-Dombriz. Optimising growth of structure constraints on modified gravity. *J. Cosmol. Astropart. Phys.*, 2019(6):020, June 2019.

[52] J. Khoury and A. Weltman. Chameleon cosmology. *Phys. Rev. D*, 69(4):044026, February 2004.

[53] J. L. Bernal, L. Verde, and A. G. Riess. The trouble with H_0. *J. Cosmol. Astropart. Phys.*, 2016(10):019, October 2016.

[54] L. Knox and M. Millea. Hubble constant hunter's guide. *Phys. Rev. D*, 101(4):043533, February 2020.