Estimating Cosmological Parameters from Galaxy Power Spectrum

Simulation-Based Inference

— Project Report —

Sabarish Suriya Swaminathan Ravichandran, 242653

Firosh Mohan Chalissery, 269605

Christy Johns, 261910

August 17, 2025

TU Dortmund University

Contents

1 Introduction

Motivation. The large-scale structure of the Universe encodes fundamental information about its composition, expansion history, and the physics of the early cosmos. A central statistical descriptor of this structure is the matter power spectrum, $P(k)$, which quantifies the variance of matter density fluctuations as a function of spatial frequency k (wavenumber). The shape and amplitude of $P(k)$ are highly sensitive to key cosmological parameters, including the Hubble constant (H_0), the matter density parameter (Ω_m), and the scalar spectral index (n_s) **planck2018**. Accurate inference of these parameters is essential for constraining the standard ΛCDM cosmological model and testing extensions or alternative cosmologies.

Problem. We aim to estimate the parameter vector

$$\boldsymbol{\theta} = \{H_0,\, \Omega_m,\, n_s\}$$

from (noisy) observations of the matter power spectrum $P(k)$ at redshift $z = 0$. For realistic galaxy surveys, the exact likelihood $p(\text{data} \mid \theta)$ is typically intractable due to nonlinear structure formation and complex observational effects. Simulation-based inference (SBI) provides a principled alternative: instead of deriving an explicit likelihood, one learns the posterior distribution $p(\theta \mid \text{data})$ directly from forward simulations.

Modeling idea. To generate training data, we use the **CAMB**, which maps parameters $\theta \mapsto P(k)$. For numerical stability, we transform to $\log_{10} P(k)$ and inject Gaussian noise in log-space to mimic observational uncertainties. We then train a neural density estimator within the `BayesFlow` framework to approximate the posterior distribution $p(\theta \mid \log_{10} P(k))$. This approach leverages modern neural density estimation to capture nonlinear dependencies among cosmological parameters.

2 Data

We generate spectra using `CAMB` for $k \in [10^{-3.3}, 10^{0.7}]$ Mpc^{-1} on a logarithmic grid (1000 points). For training efficiency, we evaluate each spectrum on n log-spaced points; in our experiment we use $n = 256$ by default and also tested one experiment at $n = 512$. The final observation model is

$$\log_{10} P_{\text{obs}}(k) \;=\; \log_{10} P_{\text{true}}(k) \;+\; \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2), \tag{1}$$

with $\sigma = 0.2$ (additive Gaussian noise in log-space).

After training, the amortized posterior can be conditioned on a held-out synthetic ”real-like” spectrum.

3 Statistical Model

Our project relies on a simulation-based Bayesian framework for cosmological parameter inference. The statistical model links cosmological parameters

$$\theta = \{H_0, \Omega_m, n_s\}$$

to noisy observations of the matter power spectrum $P(k)$.

As illustrated in Fig. 1, the log-space noise model yields realistic fluctuations around the smooth theoretical spectrum. The distributional characteristics of the noisy spectrum are further summarized in Fig. 2, which shows a histogram and kernel density estimate of $\log_{10} P_{\mathrm{obs}}(k)$ across wavenumbers.

3.1 Prior Distribution

We assign independent uniform priors reflecting physically plausible ranges of cosmological parameters:

$$H_0 \sim \mathcal{U}(50, 90), \quad \Omega_m \sim \mathcal{U}(0.2, 0.4), \quad n_s \sim \mathcal{U}(0.9, 1.1).$$

The baryon density is fixed at $\Omega_b = 0.05$, and the cold dark matter density is defined consistently as

$$\Omega_c = \Omega_m - \Omega_b.$$

3.2 Data-Generating Process

Given a parameter draw θ, the theoretical matter power spectrum is generated via `CAMB`. Specifically, CAMB is configured with $(H_0, \Omega_b, \Omega_c, n_s)$ and used to compute the linear matter power spectrum $P(k)$ at redshift $z = 0$ on a logarithmic k-grid. For numerical stability, the spectrum is transformed into log-space as $\log_{10} P_{\mathrm{true}}(k) = \log_{10} P(k)$. To emulate observational uncertainties, Gaussian noise is then added in log-space, yielding

$$\log_{10} P_{\mathrm{obs}}(k) = \log_{10} P_{\mathrm{true}}(k) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2),\ \sigma = 0.2.$$

This procedure produces paired samples (θ, x), where θ are the cosmological parameters and x is the noisy simulated spectrum. Collectively, these pairs constitute draws from the joint distribution

$$p(x, \theta) = p(x \mid \theta)\, p(\theta).$$

3.3 Bayesian Formulation

The goal is to infer the posterior distribution

$$p(\theta \mid x) \propto p(x \mid \theta)\, p(\theta).$$

However, the likelihood $p(x \mid \theta)$ is intractable due to nonlinear structure formation and observational noise. Instead of evaluating the likelihood explicitly, we employ Simulation-Based Inference (SBI), where the posterior is learned directly from simulated training data. To approximate $p(\theta \mid x)$, we use the `BayesFlow` framework, which links the forward simulator with neural density estimation. Through amortized training on thousands of simulated (x, θ) pairs, BayesFlow learns to approximate the full posterior over θ for any new observed spectrum.

4 Approximator

Adapter (BayesFlow). To use simulator-generated training data with `BayesFlow`, it must follow BayesFlow's standardized dictionary-based structure. The *inference variables* contain the parameters to be estimated — in our project: H_0, Ω_m, and n_s. The *summary* *variables* hold the simulated observables — in our case: $\log P(k)_{\text{noisy}}$. We rely on the built-in `Adapter` class, which flexibly handles tasks such as standardization and formatting.

The implemented adapter is shown in Fig. 3.

4.1 Summary network: RNN

We employed a recurrent neural network (RNN) to process the simulated power spectra. Recurrent architectures are well-suited for sequential data since they explicitly model temporal (or here: spectral) dependencies across the ordered grid of wavenumbers.

We tested gated recurrent units (GRU) and long short-term memory (LSTM) cells in a bidirectional configuration with 128 hidden units, followed by pooling and a dense layer that maps the output to a 32-dimensional latent vector $h(y)$.

The sequence length is fixed at 256 (log-spaced k-values), with a single input channel. For pooling we mainly used average pooling over timesteps, but also explored max and last-timestep pooling. The resulting fixed-length representation ensures that the inference network receives standardized embeddings regardless of sequence structure.

4.2 Inference network

The **inference network** takes this summary and approximates the posterior distribution over the parameters. We tested three architectures:

- an affine coupling flow,
- a spline coupling flow,
- and a flow matching network.

However, we mainly focus on flow matching as we had better results than other two model. This modular setup adapter, summary network, and inference network allows the model to learn informative features from the spectra and then map them to accurate posterior distributions of the cosmological parameters.

5 Training

For training, we generated an **offline dataset** consisting of 6,000 simulations for training, 1,000 for validation, and 1,000 for testing. Using an offline dataset ensured consistency across different runs and allowed us to reuse the same data when comparing different network architectures.

The model was trained for a maximum of 200 epochs with mini-batches of size 128. To avoid overfitting, we employed an *early stopping* strategy with a patience of 50 epochs and restored the best weights once the validation loss stopped improving.

We compared three different inference networks—Affine Coupling, Spline Coupling, and Flow Matching— each combined with the RNN summary network described in Sec. 4.1. All models were trained with the default BayesFlow settings. This allowed for a fair comparison between different flow architectures under identical training conditions. Flow matching provided smoother convergence and stable calibration.

6 Diagnostics

We evaluated the *reliability* and *calibration* of the trained neural posterior estimators on simulated test data using standard BayesFlow diagnostics.

6.1 Loss Trajectory

The loss trajectory (Fig. 4) showed smooth convergence for both training and validation sets. The validation curve closely followed the training curve, with no evidence of overfitting. This indicates stable optimization, good generalization to unseen data, and the effectiveness of moving-average smoothing in highlighting the downward trend.

6.2 Posterior Recovery Plots

The posterior recovery plots (Fig. 5) confirmed strong inference performance. H_0 and Ω_m were estimated very accurately, with high correlations to the ground truth ($r = 0.973$ and $r = 0.912$). Recovery for n_s was weaker ($r = 0.826$) but still followed the expected trend, showing that while n_s is more difficult to identify, the model still captures useful information about it.

6.3 Calibration (ECDF and Rank Histograms)

Calibration diagnostics (Fig. 6) assessed whether credible intervals achieved nominal coverage. The ECDF differences stayed close to zero and within the 95% confidence bands, confirming well-calibrated posteriors. Small systematic effects were observed:

- H_0 tended to be slightly *overconfident*,
- Ω_m mildly *underconfident*,
- n_s was largely well-calibrated, with only minor deviations in the tails.

6.4 z-Score Analysis

The z-score analysis and posterior contraction (Fig. 7) provided further insight into uncertainty quantification.

The z-score contraction plots showed that H_0 was recovered with high confidence and very little bias, Ω_m displayed a moderate spread but remained centered around zero (reasonable calibration), and n_s exhibited wider spread, reflecting greater uncertainty and weaker identifiability, while remaining approximately unbiased overall.

7 Summary and model comparison

Across diagnostics we compared the affine, spline, and flow matching flows combined with the RNN summary network:

- **Convergence:** All models showed smooth loss trajectories with stable convergence, and validation losses closely tracked training losses.
- **Calibration:** ECDF curves and rank histograms indicated overall well-calibrated posteriors, with only minor systematic deviations across parameters.

- **Accuracy & sharpness:** Recovery plots showed strong correlations for H_0 and Ω_m and weaker, but still informative, recovery for n_s. Posterior contraction confirmed sharper posteriors for the more identifiable parameters.

Among the tested architectures, flow matching delivered the most consistent performance across all parameters, while affine flows provided a stable baseline and spline flows offered additional flexibility. This suggests that the added expressivity of flow matching is beneficial for capturing the posterior structure in our cosmological setting.

8 Discussion Conclusion

The diagnostics show that our model recovers H_0 and Ω_m well, while n_s remains harder to constrain. H_0 is very accurate and well calibrated, Ω_m is slightly biased in the center and tails, and n_s shows larger uncertainty. Overall, the trained flows capture the main cosmological parameters reliably, with the strongest performance for H_0. Future work could focus on improving calibration and identifiability for n_s.

Future Directions:

- Expand the training dataset to improve n_s identifiability.
- Experiment with more expressive normalizing flow architectures or hierarchical priors.
- Will use CNN method in place of RNN

Overall, the results provide encouraging evidence that neural posterior estimation can be a competitive tool in cosmological parameter inference, especially as computational simulations and machine learning models continue to advance.

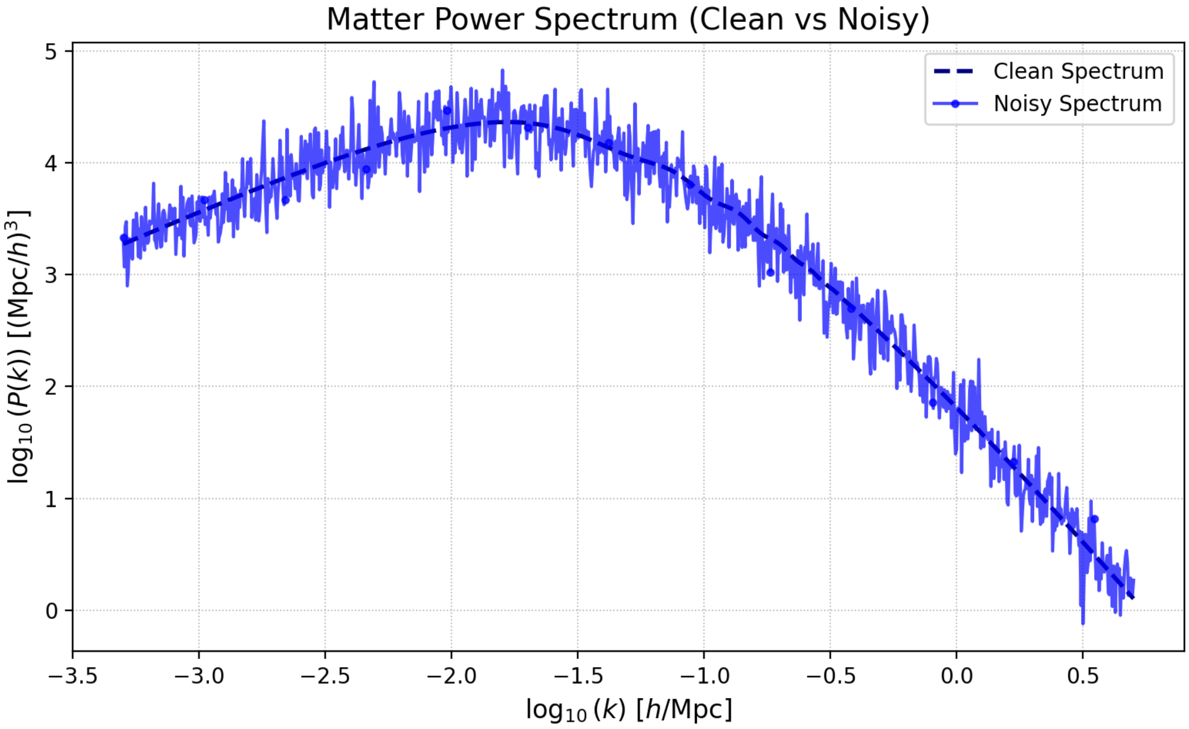


Figure 1: Matter power spectrum (clean vs. noisy). The red curve shows the clean $\log_{10} P(k)$ from CAMB; the blue trace shows the noisy observation with additive Gaussian noise in log-space ($\sigma = 0.2$), as in Eq. (1).

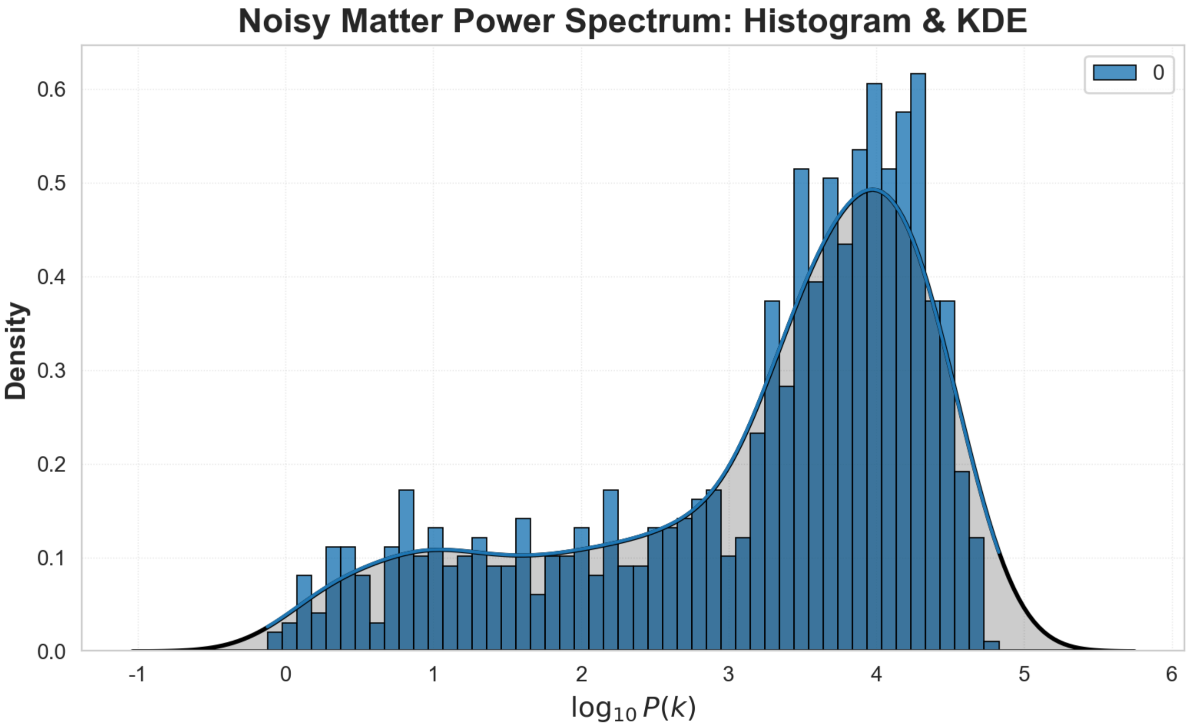
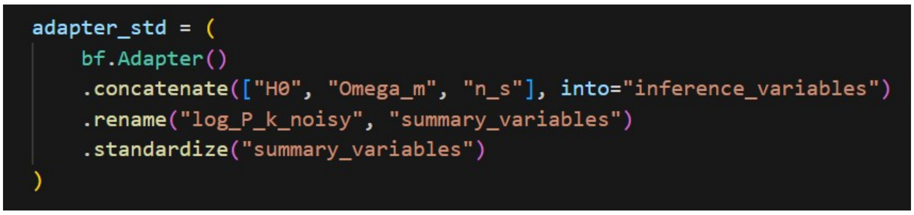


Figure 2: Distribution of the noisy matter power spectrum. Histogram and kernel density estimate (KDE) of $\log_{10} P_{\text{obs}}(k)$ across sampled k values, highlighting the variability introduced by the Gaussian noise model.



```
adapter_std = (
    bf.Adapter()
    .concatenate(["H0", "Omega_m", "n_s"], into="inference_variables")
    .rename("log_P_k_noisy", "summary_variables")
    .standardize("summary_variables")
)
```

Figure 3: Adapter configuration in BayesFlow. The adapter concatenates the inference variables, renames the simulated spectrum, and applies standardization to ensure stable training.

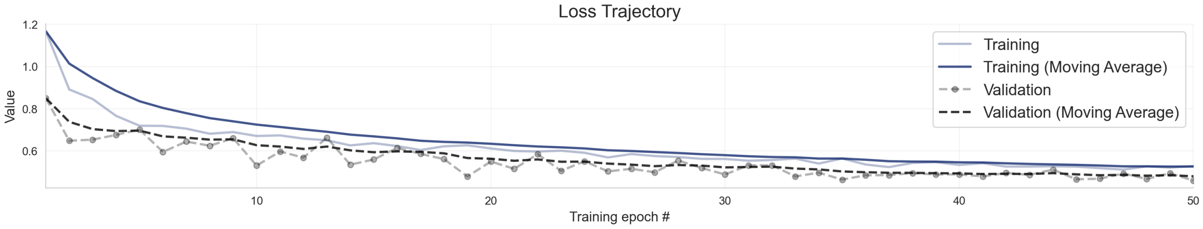


Figure 4: **Loss trajectory (Training vs. Validation).** Produced via `figures_flow_RNN`. Shows steady convergence with moving-average smoothing for both splits.

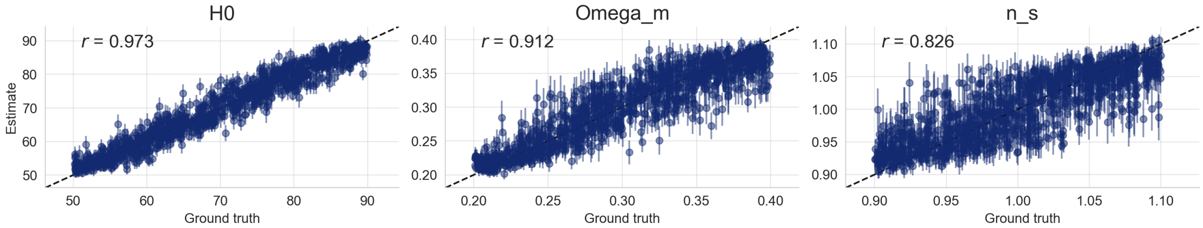


Figure 5: **Posterior recovery plots for H_0, Ω_m, and n_s.** Dashed diagonal is the ideal $y = x$ line; r denotes Pearson correlation.

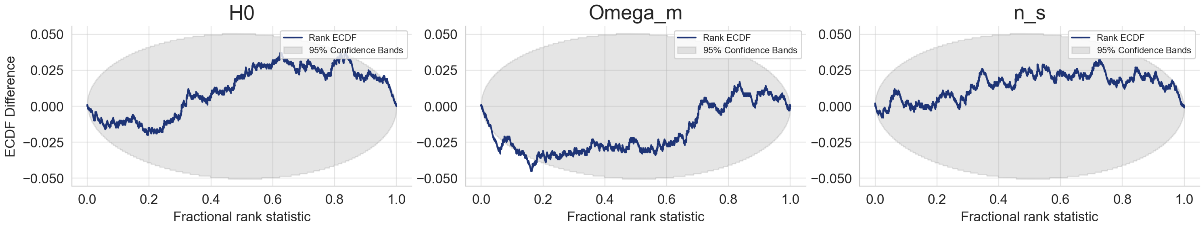


Figure 6: **Calibration ECDF differences (rank ECDF minus uniform).** The shaded band shows the 95% confidence region; curves close to zero indicate good calibration.

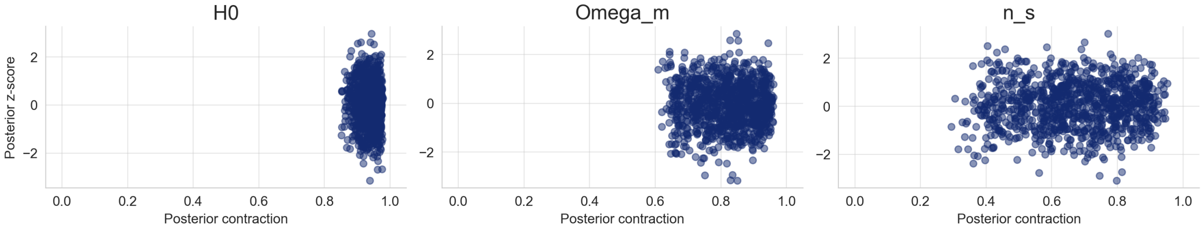


Figure 7: **z-scores vs. posterior contraction.** Points cluster near $z = 0$ with contraction < 1 indicating information gain from data.

References

[1] S. Pettini, *Intro to Cosmology – Lecture 14: Large Scale Structure.* (accessed 2025-08-09). https://people.ast.cam.ac.uk/~pettini/IntroCosmology/Lecture14.pdf.

[2] CAMB Documentation, *Code for Anisotropies in the Microwave Background.* (accessed 2025-08-09). https://camb.readthedocs.io/en/latest/.

[3] Planck Collaboration, *Planck 2018 results. VI. Cosmological parameters.* https://arxiv.org/abs/1807.06209, 2018.

[4] S. Weinberg, *Cosmology.* Oxford University Press, 2008. (overview slides: https://fma.if.usp.br/~mlima/teaching/PGF5292_2021/Weinberg_Cosmology.pdf).

[5] BayesFlow Team, *BayesFlow: Simulation-based Inference Framework.* (accessed 2025-08-09). https://bayesflow.org/main/index.html.

[6] BayesFlow GitHub Repository, *Source Code.* (accessed 2025-08-09). https://github.com/bayesflow-org/bayesflow/tree/main/bayesflow.

[7] BayesFlow GitHub Examples, *Demonstration and Tutorials.* (accessed 2025-08-09). https://github.com/bayesflow-org/bayesflow/tree/main/examples.