ParamInsight: Bayesian Parameter Inference via MCMC

Kevin Mota da Costa

January 12, 2026

Abstract

This document presents the theoretical background and implementation details of *ParamInsight*, a Python framework for Bayesian parameter inference using Markov Chain Monte Carlo (MCMC) methods. The project emphasizes statistical consistency, explicit likelihood modeling, realistic observational uncertainties, and a full MCMC implementation from scratch.

1 Introduction

Parameter inference aims to estimate model parameters given noisy observational data while properly accounting for uncertainty. *ParamInsight* provides a minimal yet rigorous framework that demonstrates:

- Likelihood-based Bayesian inference,
- Explicit construction of Gaussian noise using the Box–Muller transform,
- MCMC sampling via the Metropolis–Hastings algorithm,
- Realistic modeling of observational uncertainties,
- Statistical analysis and visualization of posterior distributions.

All components are implemented explicitly to highlight the underlying statistical and computational principles.

2 Observational Model and Likelihood

Let x_i denote the independent variable, y_i the observed data, and δy_i the uncertainty associated with each observation. For a generic two-parameter model $F(x; a, b)$, the log-likelihood function is defined as

$$\log L(a, b) = -\frac{1}{2} \sum_{i=1}^{N} \left(\frac{F(x_i; a, b) - y_i}{\delta y_i} \right)^2 . \tag{1}$$

This corresponds to a chi-squared likelihood assuming independent Gaussian errors with *point-dependent* variances. The normalization constant of the Gaussian distribution is omitted since it does not affect likelihood ratios used by the MCMC acceptance rule.

3 Observational Uncertainty Model

Unlike fixed-error models, *ParamInsight* generates realistic observational uncertainties for each data point:

$$\delta y_i = \left|\delta_{\text{inst}} + c_{\text{trend}}\, x_i + \epsilon_i\right|, \tag{2}$$

where:

- δ_{inst} is a base instrumental error,
- c_{trend} introduces a systematic trend with x,
- $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is random observational noise.

Observed data are generated as

$$y_i = F(x_i; a_{\text{true}}, b_{\text{true}}) + \mathcal{N}(0, \delta y_i^2). \tag{3}$$

This approach mimics realistic measurement processes encountered in experimental and observational sciences.

4 Random Number Generation

4.1 Uniform Distribution

Uniform random numbers are generated using NumPy's core generator and mapped to arbitrary intervals.

4.2 Gaussian Distribution: Box–Muller Transformation

Gaussian random variables are generated explicitly using the Box–Muller transform:

$$z = \sqrt{-2\ln u_1}\cos(2\pi u_2), \tag{4}$$

$$u_1, u_2 \sim \text{Uniform}(0, 1). \tag{5}$$

The resulting variable z follows a standard normal distribution and is scaled to arbitrary mean μ and standard deviation σ when required. This implementation avoids reliance on black-box Gaussian generators.

5 MCMC Sampling Method

5.1 Metropolis–Hastings Algorithm

The parameter vector $\theta = (a, b)$ is sampled using the Metropolis–Hastings algorithm. Given the current state θ_{n-1}, a proposal θ' is generated and accepted with probability:

$$\alpha = \min\left(1, \exp\left[\log L(\theta') - \log L(\theta_{n-1})\right]\right). \tag{6}$$

Rejected proposals result in the chain retaining its previous state.

5.2 Memory and Inertia Term

To improve exploration efficiency, *ParamInsight* introduces a simple inertia term using the two previous steps:

$$a' = a_{n-1} + \tfrac{1}{2}(a_{n-1} - a_{n-2}) + \eta_a, \tag{7}$$
$$b' = b_{n-1} + \tfrac{1}{2}(b_{n-1} - b_{n-2}) + \eta_b, \tag{8}$$

where $\eta_a, \eta_b \sim \mathcal{N}(0, \sigma^2)$. Although this proposal is not strictly Markovian, detailed balance is preserved through the acceptance rule, ensuring correct posterior sampling.

6 Implemented Models

The framework includes four example models:

- Linear: $F(x) = ax + b$
- Logarithmic: $F(x) = a \log(bx)$
- Quadratic: $F(x) = ax + bx^2$
- Inverse: $F(x) = a/x + b$

Each model is tested with synthetic observational data and independently sampled using the same MCMC engine.

7 Implementation Overview

The project is organized into the following modules:

- `distributions.py`: likelihood computation and random generators,
- `mcmc.py`: 2D Metropolis–Hastings sampler with inertia,
- `examples.py`: data generation, experiment setup, and execution,
- `utils.py`: visualization and result storage,
- `main.py`: entry point for running all experiments.

Each experiment saves observational data, MCMC chains, diagnostic plots, and final statistical summaries.

8 Conclusion

ParamInsight provides a transparent and statistically grounded framework for Bayesian parameter inference. By explicitly implementing likelihoods, noise models, and MCMC sampling from first principles, the project offers both educational value and a solid foundation for extension to more complex inference problems.

9 References

1. Gelman et al., *Bayesian Data Analysis*, CRC Press (2013).
2. Robert & Casella, *Monte Carlo Statistical Methods*, Springer (2004).
3. Box & Muller (1958), *Annals of Mathematical Statistics*.
4. Metropolis et al. (1953), *Journal of Chemical Physics*.
5. Neal (1993), Technical Report CRG-TR-93-1.