ParamInsight: MCMC for Parameter Inference

Kevin Mota da Costa

January 10, 2026

Abstract

This document presents the theory and implementation behind *ParamInsight*, a Python-based framework for performing Bayesian parameter inference using Markov Chain Monte Carlo (MCMC) methods. The tool is designed for educational purposes and as a demonstration of data analysis, statistical inference, and stochastic process implementation from scratch.

1 Introduction

Parameter inference is a fundamental task in statistical modeling. The goal is to determine the most probable values of model parameters given observational data, while accounting for measurement uncertainties. *ParamInsight* provides a minimal yet fully functional framework to demonstrate:

- Bayesian inference using likelihood functions,
- Stochastic simulation via MCMC,
- Generation of Gaussian noise using the Box-Muller transformation,
- Statistical evaluation and visualization of inferred parameters.

2 Observational Model and Likelihood

Let x_i be the independent variable, y_i the observed dependent variable, and δy_i the measurement uncertainty associated with y_i. For a model $F(x; a, b)$ with parameters a and b, the log-likelihood function under the assumption of Gaussian errors is:

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

This expression measures how compatible a given pair of parameters (a, b) is with the observed data. Maximizing $\log L$ corresponds to minimizing the weighted sum of squared residuals (χ^2), a standard method in parameter estimation.

3 Generation of Observational Noise: Box-Muller Transformation

In order to simulate realistic observational datasets, Gaussian noise must be added to the model. *ParamInsight* implements the Box-Muller transformation:

$$z_0 = \sqrt{-2 \ln u_1} \cos(2\pi u_2) \tag{2}$$

$$z_1 = \sqrt{-2 \ln u_1} \sin(2\pi u_2) \tag{3}$$

where $u_1, u_2 \sim \text{Uniform}(0, 1)$. This method transforms uniform random variables into independent standard normal variables. Each observed data point is then generated as:

$$y_i = F(x_i; a_{\text{true}}, b_{\text{true}}) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \delta y_i^2) \tag{4}$$

where ϵ_i represents the stochastic measurement noise.

4 Stochastic Processes and Monte Carlo Sampling

The MCMC algorithm is a stochastic process that constructs a sequence (chain) of parameter values. Each proposed step depends probabilistically on the previous step(s), allowing exploration of the posterior distribution.

4.1 Metropolis-Hastings Algorithm

Given a current state (a_{n-1}, b_{n-1}), a new proposal (a', b') is generated using:

$$a' = a_{n-1} + \text{inertia term} + \eta_a, \quad b' = b_{n-1} + \text{inertia term} + \eta_b \tag{5}$$

where $\eta_a, \eta_b \sim \mathcal{N}(0, \sigma^2)$. The proposal is accepted with probability:

$$\alpha = \min\left(1, \exp(\log L(a', b') - \log L(a_{n-1}, b_{n-1}))\right) \tag{6}$$

Otherwise, the chain retains the previous state. This ensures that, over many iterations, the chain samples from the posterior distribution.

4.2 Memory and Inertia

ParamInsight adds a simple memory term using $n-1$ and $n-2$ steps to give the chain inertia, which can improve convergence by proposing steps that account for recent chain history.

5 Implementation Overview

The tool is implemented in Python using `numpy` and `matplotlib`. Key modules:

- `distributions.py`: likelihood computation, Box-Muller noise generator.
- `mcmc.py`: Metropolis-Hastings MCMC with n-1/n-2 memory.

- `examples.py`: predefined example models and data generation.
- `utils.py`: visualization (trace, histogram, scatter) and results saving.

6 Example: Logarithmic Model

Consider the model $F(x) = a \log(bx)$ with true parameters $a_{\text{true}} = 1.5$, $b_{\text{true}} = 0.5$. Observational data is generated with Gaussian noise as described above.

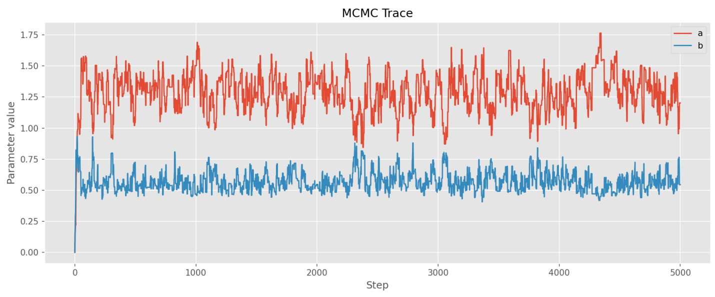

Figure 1: Trace plot for a and b in the logarithmic example.

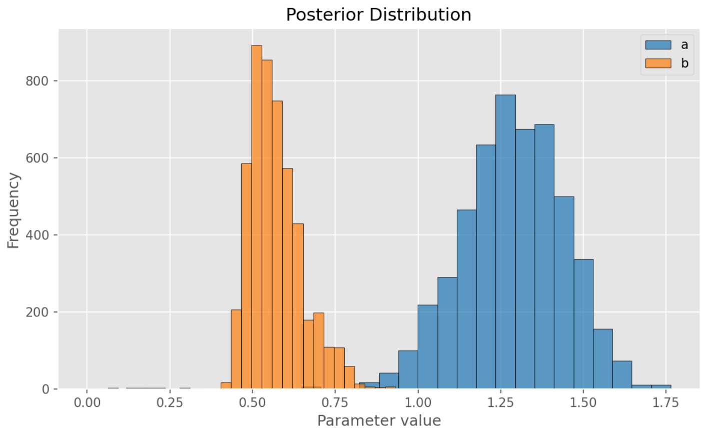

Figure 2: Posterior histogram for a and b.

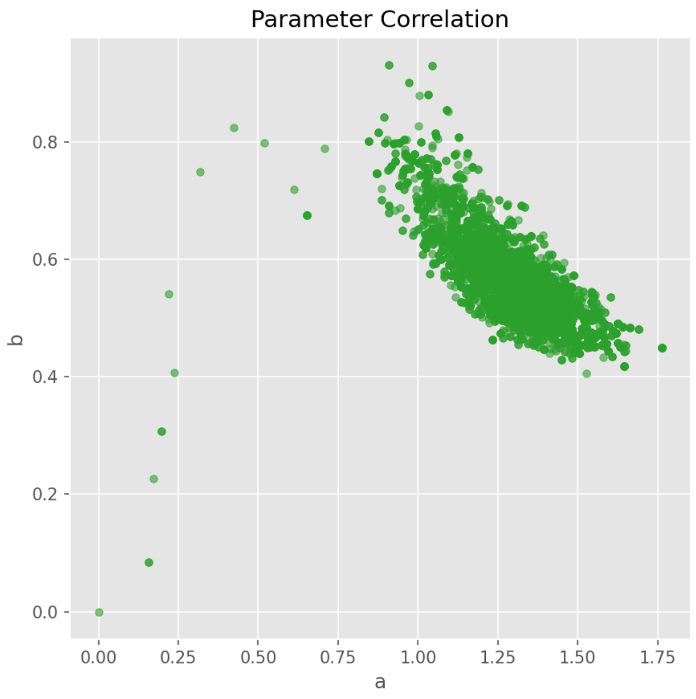

Figure 3: Scatter plot of sampled (a, b) pairs.

7 Conclusion

ParamInsight provides a hands-on demonstration of Bayesian parameter inference using MCMC. It connects stochastic processes, likelihood-based inference, and data simulation in a single framework. Although the example models are simple, the framework is generic and can be applied to other models with two parameters, highlighting skills in statistical analysis, MCMC, and Python implementation from scratch.

8 References

1. Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A., & Rubin, D.B. (2013). *Bayesian Data Analysis*. CRC Press.

2. Robert, C.P., & Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer.

3. Box, G.E.P., & Muller, M.E. (1958). A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*, 29(2), 610–611.

4. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21, 1087–1092.

5. Neal, R.M. (1993). Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical Report CRG-TR-93-1, University of Toronto.