Gravitation Wave Identification

Using Metropolis Hastings Algorithm

Group 1

Anirudh Bhat

Samyak Rai

Shanmukh Machiraju

Index

Problem Statement

Problem Statement

Given a time series strain data with added noise with the structure of a gravitational wave as given below

$$h(t) = \alpha e^{t}[1 - \tanh\{2(t - \beta)\}]\sin(\gamma t)$$

α, β, γ are parameters that signify the physical properties of the given wave. There value ranges are

$$0 < \alpha < 2$$
$$1 < \beta < 10$$
$$1 < \gamma < 20$$

We need to determine the parameter values using a Metropolis Hastings Random walk algorithm in the 3 dimensional space.

Understanding Wave Parameters

Let us visualize how the parameters α, β, and γ influence the waveform

1. α controls the amplitude of the signal
2. β shifts the signal in time
3. γ controls the oscillation frequency

Animation Paramater effects

Methodology

Random Walks

1. **Initialization:** We start with initial parameter values at the midpoints of the given ranges so

$$\alpha = 1, \beta = 5, \gamma = 10$$

2. **Random Walk:**

For each iteration we propose a new set of parameters using

$$\theta_{\text{new}} = \text{normal}\left(\theta_{\text{initial}}, \sigma^2\right) \quad \text{where} \;\; \sigma = [0.01, 0.07, 0.07]$$

The new value is discarded or chosen based on an Acceptance Probability defined as

$$A(\theta_{new}, \theta_{initial}) = \min\left(1, \frac{\text{Posterior}(\theta_{new})}{\text{Posterior}(\theta_{initial})}\right)$$

The Posterior function is defined as the following

```
def likelihood_reduced(y_data: np.ndarray, y_prior: np.ndarray):
    y_err = 0.1 * np.std(y_data)
    Y = np.mean((y_data - y_prior) ** 2) / y_err**2
return -0.5 * Y
```

This is different from the function provided in the problem statement, we will explain why this is better in Section 4

Stochastic Maximum Likelihood Estimation

Hello

Why not Bayesian Inference ?

Hello

Results

Parameter Values

Parameter	α (alpha)	β (beta)	γ (gamma)
Median Value	1.44	3.90	10.00
95% Credibility Interval	0.90 - 1.93	3.61 - 4.19	9.92 - 10.08
Effective Sample Size	62.3	121.7	800.0
MC Standard Error	0.036	0.013	0.001

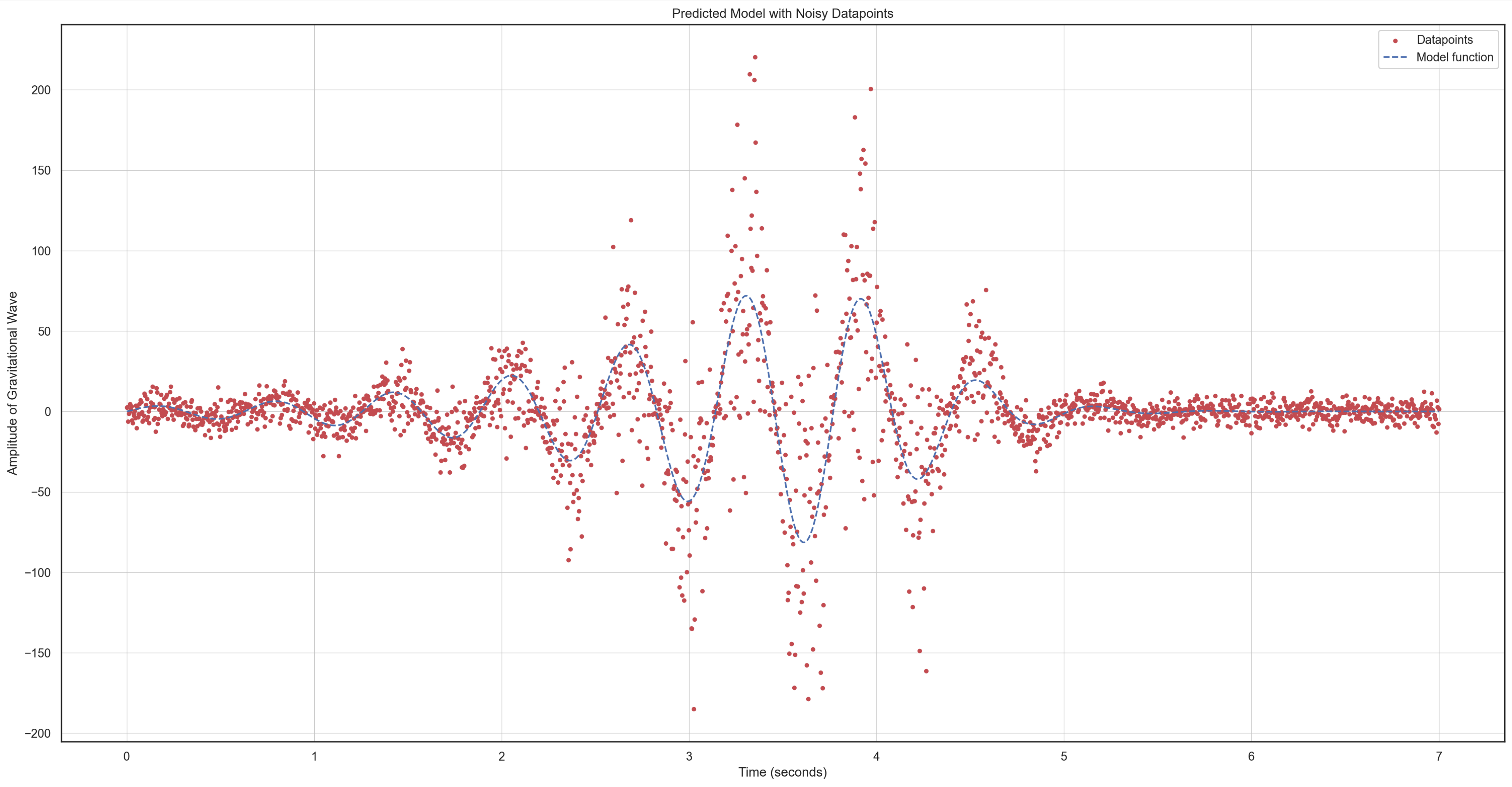
The MCMC Algorithm ran with **Acceptance Ratio** of 0.263.

The Global **Signal to Noise Ratio** was 1.00, with Local SNR of 1.02

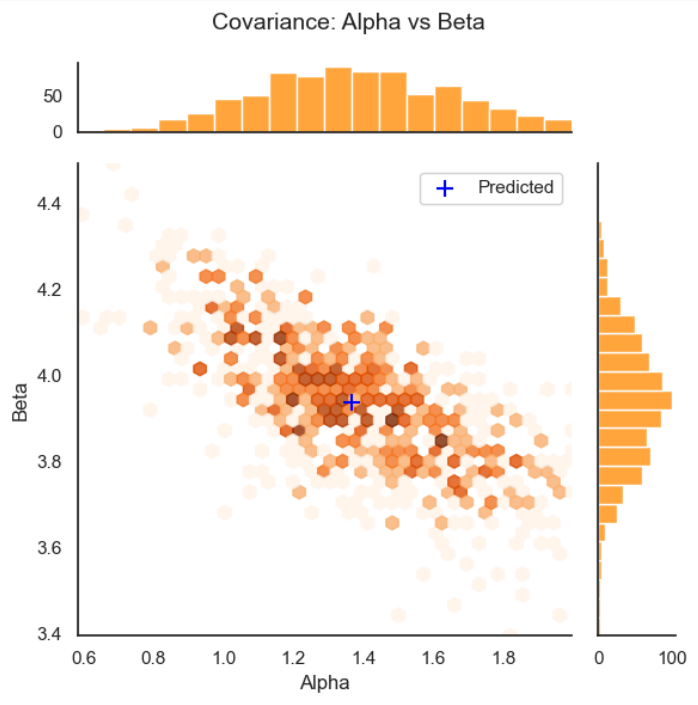
Measurement Metrics for Metropolis Hastings

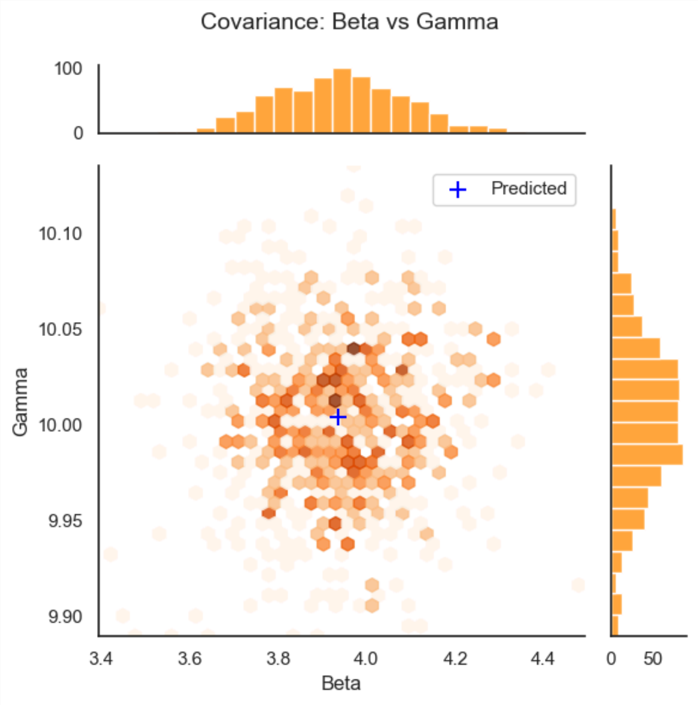
Talk about signal to noise ratio, ESS, AFC, MC Std Err

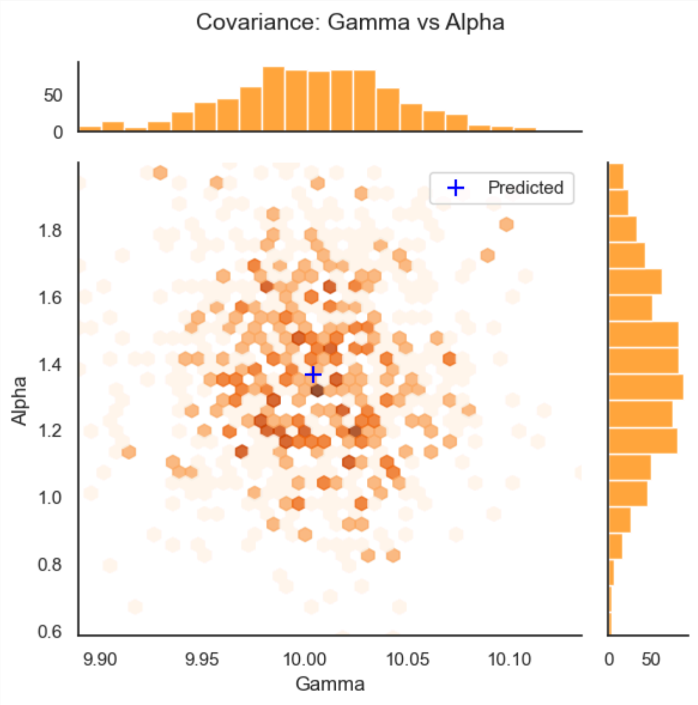
Prediction vs Data



Covariance Scatter Plots





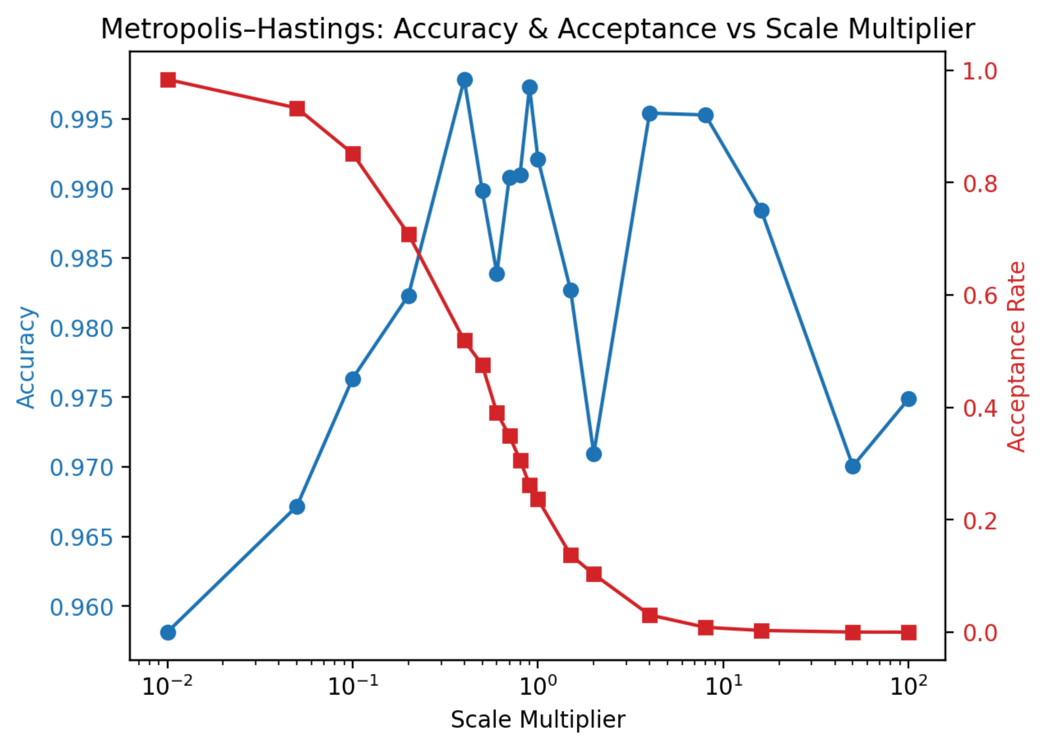


Histograms and Trace Plots

Heloo

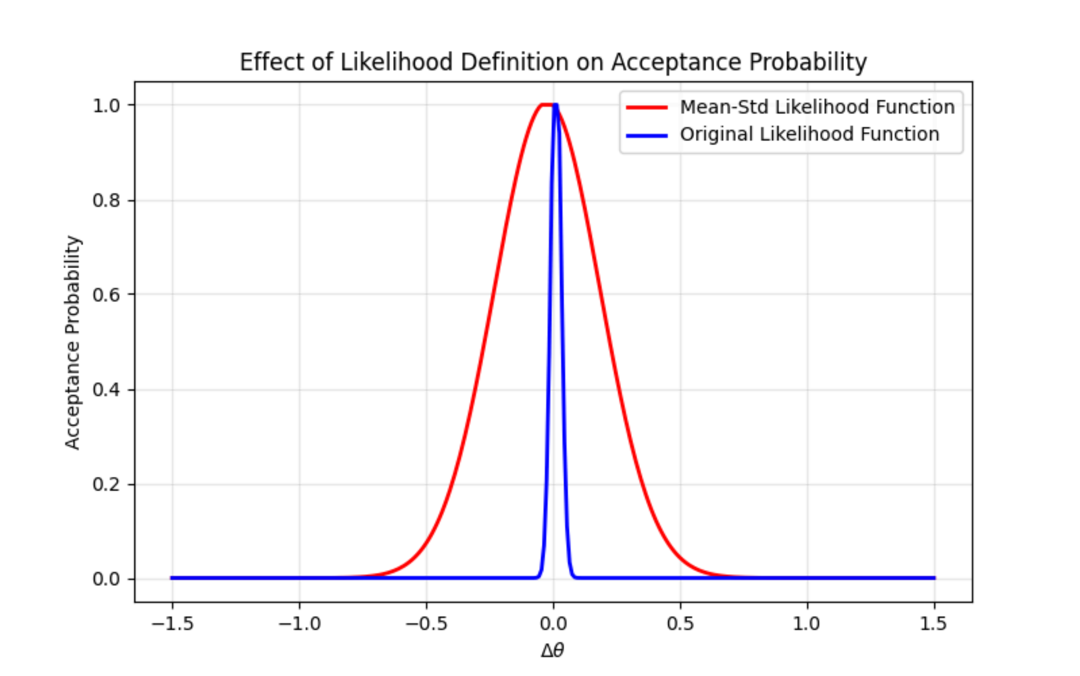
Optimization

Scale Selection



- **Small Scales** (< 10^{-1})
  - Chain barely moves → strong autocorrelation → **low accuracy.**
- **Near Chosen Scales**
  - (Roberts & Rosenthal, 1997) predicts optimal acceptance ≈ **0.234** for high-dimensional targets.
  - Accuracy peaks — this is the optimal region for efficient sampling.
- **Large Scales** (> 5)
  - Acceptance rate falls → chain stagnates → **accuracy degrades.**

The original likelihood makes even tiny parameter changes look catastrophically unlikely, driving acceptance to near zero. The new version fixes this by taking a mean instead of summation, keeping acceptance stable.




```
def likelihood_new(y_data, y_prior):
    y_err = 0.1 * np.std(y_data)
    Y = np.mean((y_data-y_prior)**2)/y_err**2
    return -0.5 * Y

def likelihood_original(y_datay, y_priory):
    y_err = 0.1 * (y_data + y_prior) + 1e-6
    Y = np.sum(((y_data - y_prior)/y_err) ** 2 )
    return -0.5 * Y
```

Data Generation for Better Inference

Hello

Code Structure

Hello

Thank You