The Solar Flare Pulse: Stochastic Signal Recovery

Team Name: LONE STAR
Team Members: Aneesh Shastri

January 8, 2026

1 Description of the Physical Phenomenon

The project analyzes a high-cadence (frequency of data points) time series captured by the Solar Dynamics Observatory (SDO). The data represents a localized magnetic reconnection event, commonly known as a stellar flare.It is a phenomenon where a star emits intense electromagnetic radiation (across the entire EM spectrum), caused by the emission of magnetic energy that happens when field lines snap and reconnect into a better configuration. Physically, this event is characterized by:

1. **Thermal Phase:** An exponential rise in intensity as plasma heats up due to magnetic flux conversion. This is modeled by:-
$$e^t \tag{1}$$
2. **Quenching:** A rapid shutdown of the event as the magnetic loop ruptures, approximated by a hyperbolic tangent decay. This is modeled by:-
$$[1 - \tanh(2(t - \tau))] \tag{2}$$
3. **Oscillation:** magnetohydrodynamic (MHD) waves [which can be approximated to sinusoidal waves] emerge within the plasma loop. This is modeled by:-
$$A \sin(\omega t) \tag{3}$$

The objective is to recover the latent physical parameters (Amplitude, Quench Time, Frequency) from heavily noisy sensor data.

2 Mathematical Model

2.1 Signal Template

The intensity $S(t)$ is modeled by the analytical function:

$$S(t; A, \tau, \omega) = Ae^t \left[1 - \tanh(2(t - \tau))\right] \sin(\omega t) \tag{4}$$

where the parameters are defined as:

- A: Intensity Scale (Amplitude). Constraints: $0 < A < 2$.
- τ: Quench Time (Event Peak). Constraints: $1 < \tau < 10$.
- ω: Angular Frequency. Constraints: $1 < \omega < 20$.

2.2 Error Model & Numerical Stability

The problem statement specifies a relative error model $\sigma_{\text{rel}} = 0.2 \cdot |y_{\text{data}}|$. However, a purely relative error model creates singularities at zero-crossings (where $y \to 0 \implies \sigma \to 0$), causing the Log-Likelihood to diverge. This also results in points closer to 0 having much higher weights compared to points further away, which will cause MH MCMC to fail to correctly estimate the parameters.

To ensure numerical stability as per the problem hint, I implemented a **Mixed Error Model** by introducing a constant noise floor. σ_{floor}:

$$\sigma_i = 0.2 \cdot |y_i| + \sigma_{\text{floor}} \tag{5}$$

The magnitude of σ_{floor} can be interpreted as 'dark noise' or 'background noise' that is always present in our observations regardless of the phenomenon under observation.

2.3 Likelihood Function

Assuming Gaussian noise, the Log-Likelihood function $\mathcal{L}$ is given by:

$$\mathcal{L} = -\sum_{i=1}^{N} \left(\frac{y_{\text{data},i} - S(t_i; \theta)}{\sigma_i} \right)^2 \tag{6}$$

3 Approach and Simulation Methodology

3.1 Noise Floor Optimization

To determine the physically correct noise floor k (where $\sigma_{\text{floor}} = k \cdot \sigma_{\text{data}}$), I solved for the value that aligns the error estimates with the observed data scatter. I defined a target function based on the **Reduced Chi-Squared statistic** (χ^2_ν):

$$f(k) = \chi^2_\nu(k) - 1.0 = 0 \tag{7}$$

Using Differential Evolution (A stochastic global optimization algorithm) with the given parameter search space, along with a root finding algorithm (brentq) on the target function, I determined the optimal noise floor parameter to be $k \approx 0.375$. This ensures that our error bars almost perfectly describe the variance in the data ($\chi^2_\nu \approx 1.00$).

NOTE: The χ^2_ν statistic,

$$\chi^2_\nu = \frac{\chi^2}{\nu} = \frac{1}{N-p} \sum_{i=1}^{N} \left(\frac{y_{\text{data},i} - y_{\text{model},i}}{\sigma_i} \right)^2$$

Is a number that tells us how good our model fits based on the error the data is assumed to have.
Essentially, it takes the model and how it fits the data, and tells you whether you have overestimated the errors or underestimated them. The closer it is to one, the better.
[$\chi^2_\nu < 1$ mean you're overestimating the error while $\chi^2_\nu > 1$ means you're underestimating them]

3.2 Markov Chain Monte Carlo (MCMC)

I performed Bayesian parameter Estimation using the Metropolis-Hastings algorithm. But before that, I had to obtain information about the covariance of the parameter's distribution, to ensure my HM MCMC has an easier time converging. This was done by a Pilot Tuning phase using Robbins-Monro MCMC to find the covariance matrix first.

1. **Initialization:** The chain was initialized at pre-defined middle values [1,5,10] set by me. (although given enough iterations any reasonable guess should result in it converging.)

2. **Robbins Monro MCMC:** I ran Robbins-Monro MCMC for 10,000 iterations, and after removing data from the burn-in phase (identified using the sigma-iteration plot shown later) I used the chain to generate the covariance matrix for my HM MCMC

3. **Metropolis Hastings MCMC:** I ran a standard HM MCMC using the covariance matrix I obtained earlier and ran 400,000 iterations. The trace plots reveal that the algorithm has converged nicely and the posterior distributions are valid.

4 Results and Visual Outputs

4.1 Best Fit Parameters (MAP)

Since uniform priors were used, the Maximum A Posteriori (MAP) estimate corresponds to the Maximum Likelihood estimate. MAP was found using Kernel density estimation, and the uncertainty values are taken assuming the posterior distributions are Guassian.

Parameter	**Symbol**	**MAP Estimate**	**Uncertainty** (1σ)
Amplitude	A	0.2426	± 0.0095
Quench Time	τ	5.066	± 0.0280
Frequency	ω	10.001	± 0.0047

Table 1: Final parameter estimates derived from the MCMC posterior distributions.

4.2 Goodness of Fit

The final model achieves a Reduced Chi-Squared of $\chi^2_\nu = 0.99995$, indicating an ideal statistical fit to the noisy data.

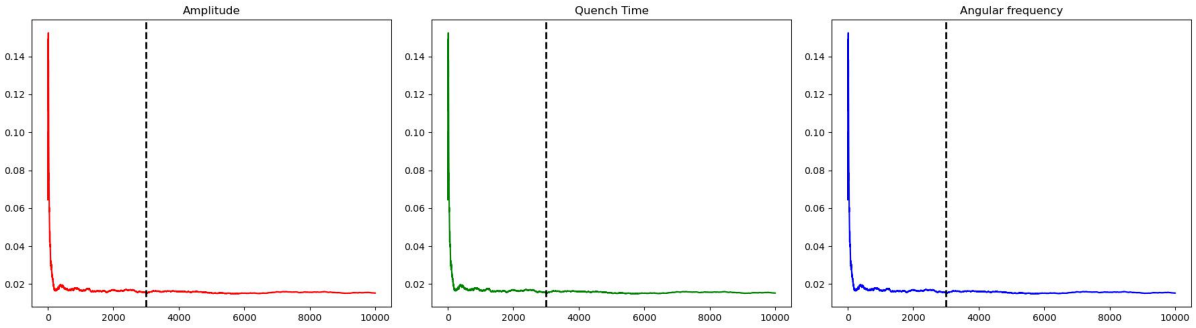

Figure 1: **Sigma-iteration:** This sigma vs iteration graph shows how proposal step size varies with the number of iterations. Only when the variation stops,MCMC is valid. thus the point where the line appears completely flat was chosen as the cutoff for burn-in period

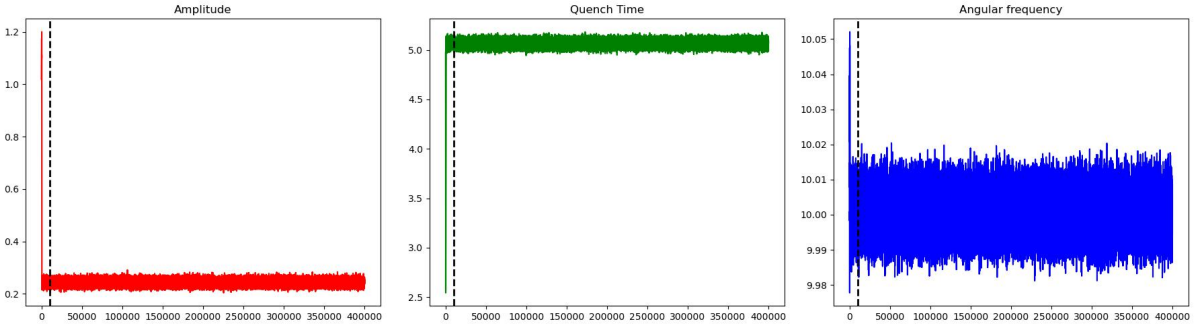

Figure 2: **Trace Plots:** Evolution of the MCMC chain for all three parameters. The "fuzzy caterpillar" shape indicates excellent mixing and convergence.

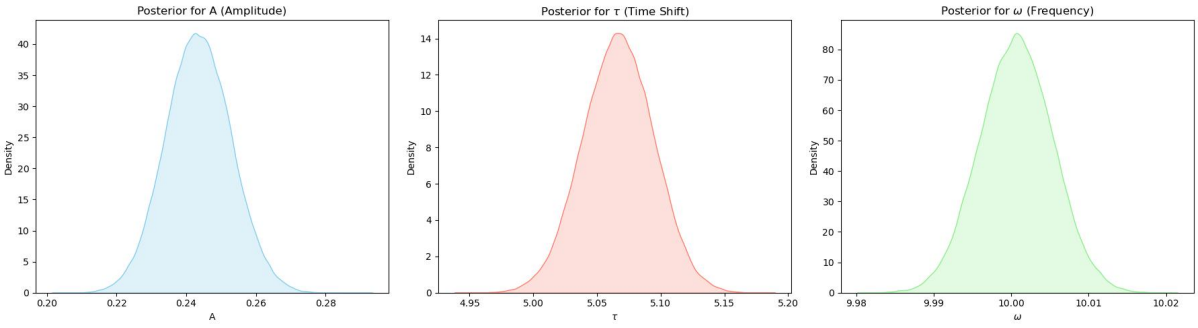

Figure 3: **Posterior Distributions:** KDE(Kernel density estimation) plots of the sampled parameters showing Gaussian-like uncertainty profiles.

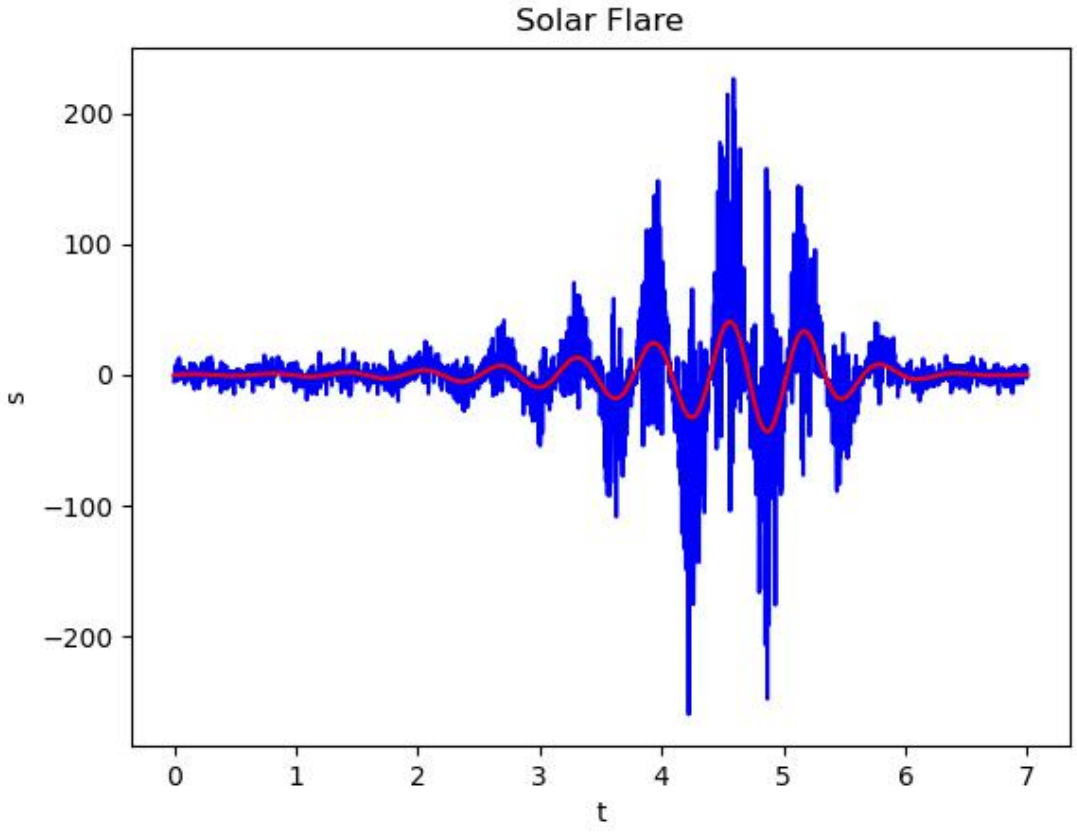

Figure 4: **Model Fit:** The MAP model (red line) overlaid on the raw sensor data (zig zag blue line).

5 Software and Tools

The simulation was performed using Python 3.13.5 . Key libraries included:

- **NumPy:** Vectorized array operations and math functions.
- **Pandas:** Data loading and preprocessing.
- **SciPy (optimize):** `differential_evolution` and `minimize` for finding the global optimum and MAP estimates.
- **SciPy (stats):** `gaussian_kde` for posterior analysis.
- **Matplotlib & Seaborn:** Generation of trace plots and histograms.