Tracking of Erratically Moving Objects Using (Non)-Gaussian Process Models

- Log -

Mingkun Li

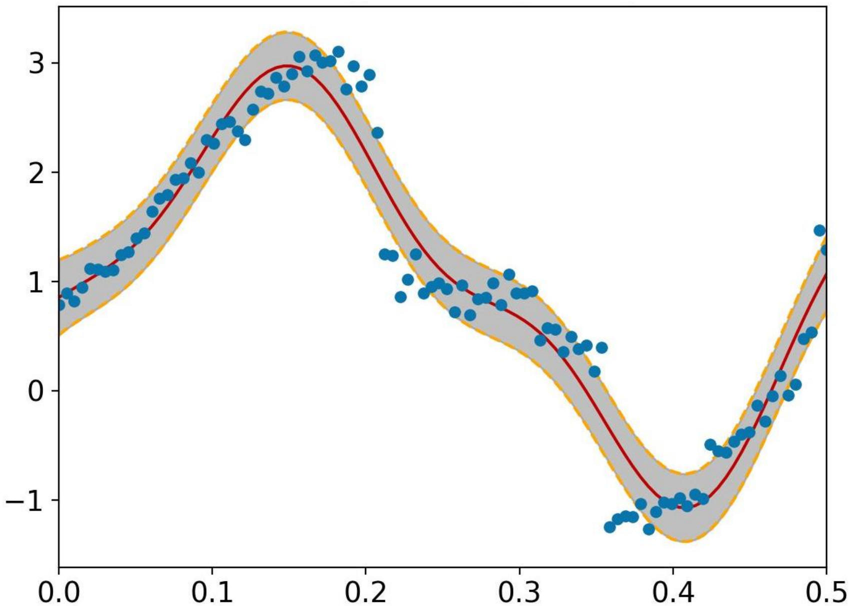
(Supervised by Professor Simon Godsill)

mingkunli01@gmail.com / ml2117@cam.ac.uk

June 20, 2025

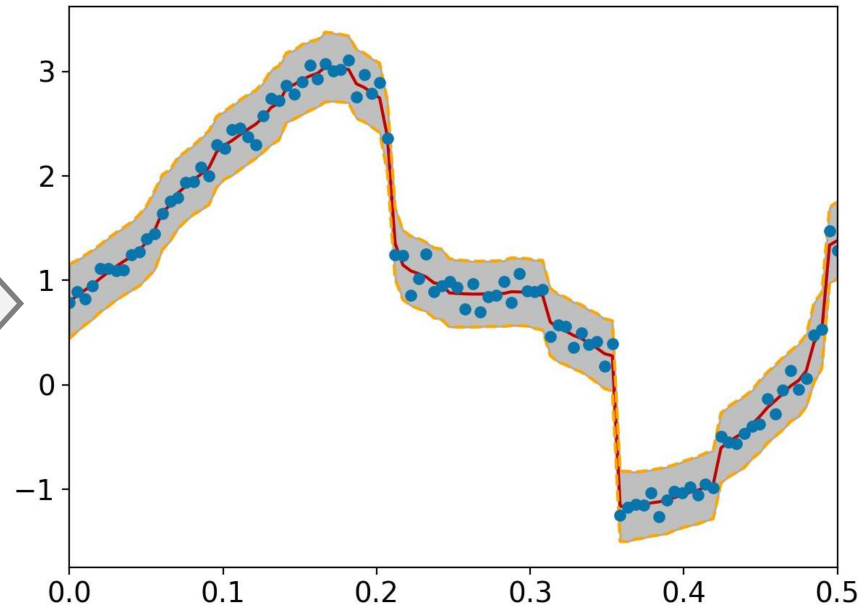
Motivation & Introduction

- sudden *shocks* can not be accurately modelled by Gaussian processes



(Gaussian Process Regression)

?



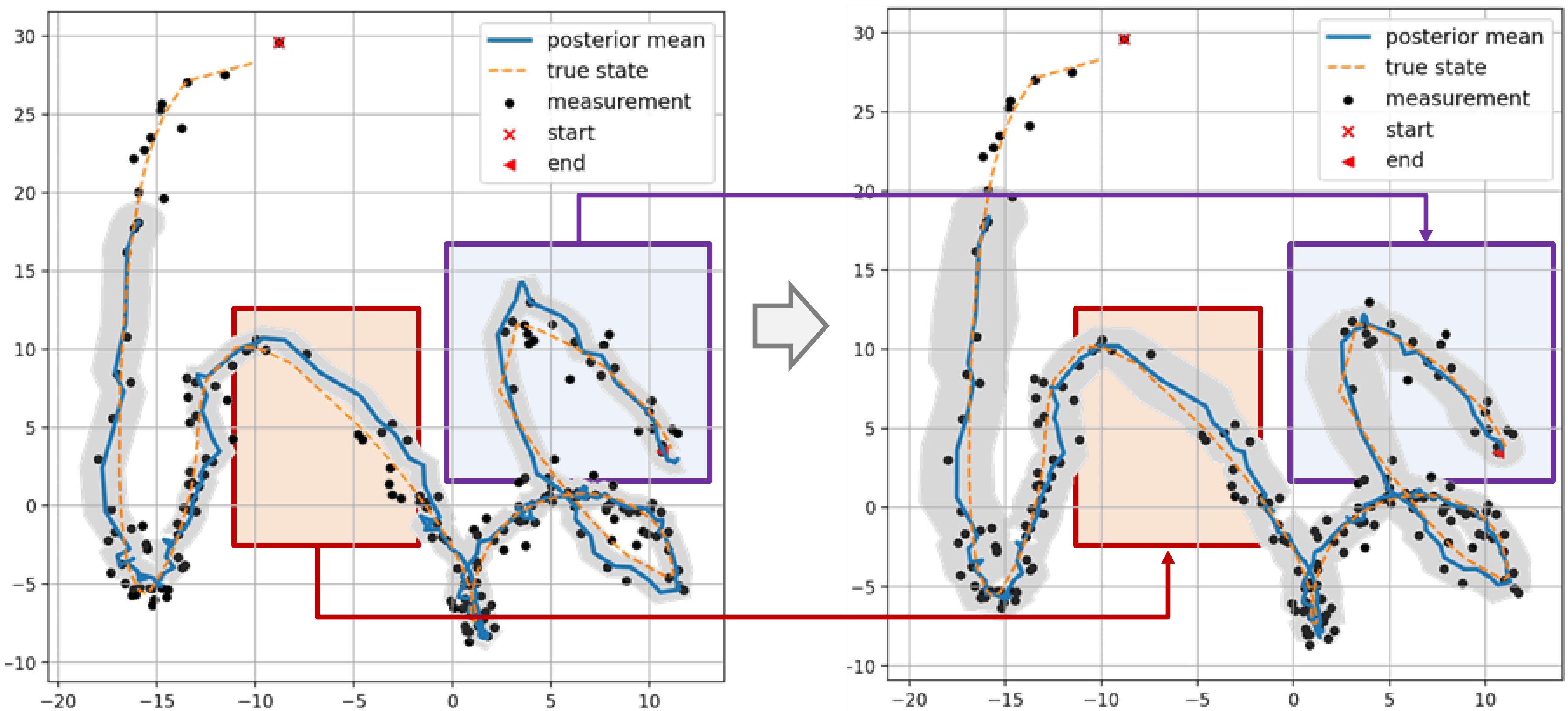
(Non-Gaussian Process Regression)

[1] Yaman Kındap and Simon Godsill. Non-gaussian process regression. *arXiv preprint arXiv:2209.03117*, 2022.

Application & Goal

[2] Yaman Kındap and Simon Godsill. Non-gaussian process dynamical models. *IEEE Open Journal of Signal Processing*, 2025.

- Apply the model in tracking (and prediction) problem in practice



(Gaussian Process Tracking)

(Non-Gaussian Process Tracking)

Tracking Problem

Time-ordered Data → Sequential Data (Tracking)

Non-Gaussian Process Model (NGP)

- Neural Processes models a distribution over functions by using a **latent variable** (often stochastic) and **neural networks** to output predictive distributions. The mean m and the covariance K are not directly modelled or changed as independent stochastic processes.

- **Gaussian Process**:

$$f(t) \sim \boxed{\mathcal{GP}\big(m(t), K(t,t')\big)}$$

- How about we apply a **time-change operation** $T(t)$?

Mixture of conditional Gaussian Processes:

$$f|T(t) \sim \mathcal{GP}\Big(m\big(T(t)\big), K\big(T(t), T(t')\big)\Big) := \boxed{\mathcal{GP}\big(m_T(t), K_T(t,t')\big)}$$

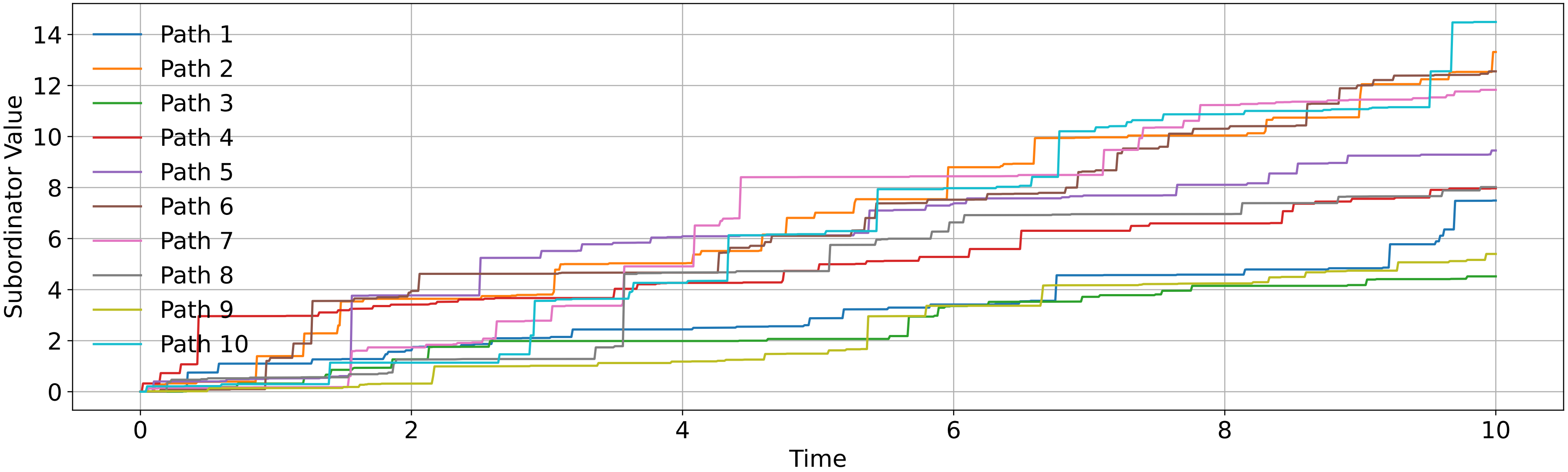
Clearly, the resulting **marginal prior** over the latent process:

$$p(f) = \int p(f,T)\mathrm{d}T = \int p(f|T)p(T)\mathrm{d}T$$

NGP – Time Change Operation (Subordinator)

What properties should the latent input transformation have?

➡ Non-negative, non-decreasing, randomly maps inputs while preserving the order.



We use the **subordinator** as the latent input process, taking values in $[0, \infty)$:

- A *Lévy process,* but contains no *Brownian* or *drift* part.
- Independent & stationary increments with *no* fixed discontinuities.

NGP – Short Noise Representation

- Lévy-Khintchine representation:

Lévy measure with $\int_0^\infty \min(1, x)\nu(\mathrm{d}x) < \infty$

$$\mathbb{E}[\exp(\mathrm{i}uT_t)] = \exp\left(t \int_0^\infty (\mathrm{e}^{\mathrm{i}ux} - 1)\nu(\mathrm{d}x)\right)$$

- Lévy–Itô decomposition:

Poisson random measure

$$T_t = \int_0^\infty x \cdot N([0, t], \mathrm{d}x)$$

- Short-noise method:

jump location

$$N = \sum_{i=1}^\infty \delta_{(V_i, M_i)}$$

jump magnitude

$$\boxed{T_t = \sum_{i=1}^\infty M_i \mathbb{I}_{\{t \geq V_i\}}}$$

Short-noise representation

- Now, with $\{V_i, M_i\}_i$, we can obtain a realisation of process T_t.

NGP – Inverse Levy Measure Algorithm

- We generally use the *inverse Lévy measure algorithm* to obtain $\{V_i, M_i\}$

$$h(\Gamma_i) = \inf_x \left\{x \in \mathcal{X}: \nu^+(x) = \nu\big([x,\infty)\big) = \Gamma_i\right\} = M_i$$

arrival times randomly simulated from a Poisson process

- Clearly, since Γ_i increases with i, the resulting jump sizes $M_i = h(\Gamma_i)$ decrease.
- When the inverse tail function h is not available in closed form, it becomes infeasible to simulate jump sizes directly via inversion. In such cases, a *thinning method* is employed.
  - Simulate a tractable Poisson point process N_0 with a Lévy measure ν_0 that *dominates* the target measure ν, i.e.,

$$\frac{\nu(\mathrm{d}x)}{\nu_0(\mathrm{d}x)} \leq 1, \qquad \text{for all } x > 0$$

    and for which the inverse tail function of ν_0 is explicitly computable.
  - Jump sizes x are sampled from ν_0, and each proposed jump is accepted with probability $\nu(x)/\nu_0(x)$. The accepted jumps form a thinned Poisson process whose intensity matches the desired Lévy measure ν. These accepted values are then used as the jump magnitudes M_i for the target subordinator.

NGP – Tempered Stable (TS) Process

- Consider the **tempered stable (TS) process**. Its Lévy measure admit closed-form expressions and can be efficiently sampled.
- Generally speaking, for a TS process, the Lévy measure

$$\nu(dx) = \left(\frac{C_+ \exp(-\lambda_+ x)}{x^{1+\alpha}} \mathbb{I}_{\{x>0\}} + \frac{C_- \exp(-\lambda_- x)}{|x|^{1+\alpha}} \mathbb{I}_{\{x<0\}} \right) \mathrm{d}x$$

- For a TS-subordinator process,

positive α-stable process with Lévy density ν_0

$$\nu(dx) = \frac{C}{x^{1+\alpha}} \exp(-\lambda x)\, \mathbb{I}_{\{x>0\}} \mathrm{d}x$$

exponential tempering function

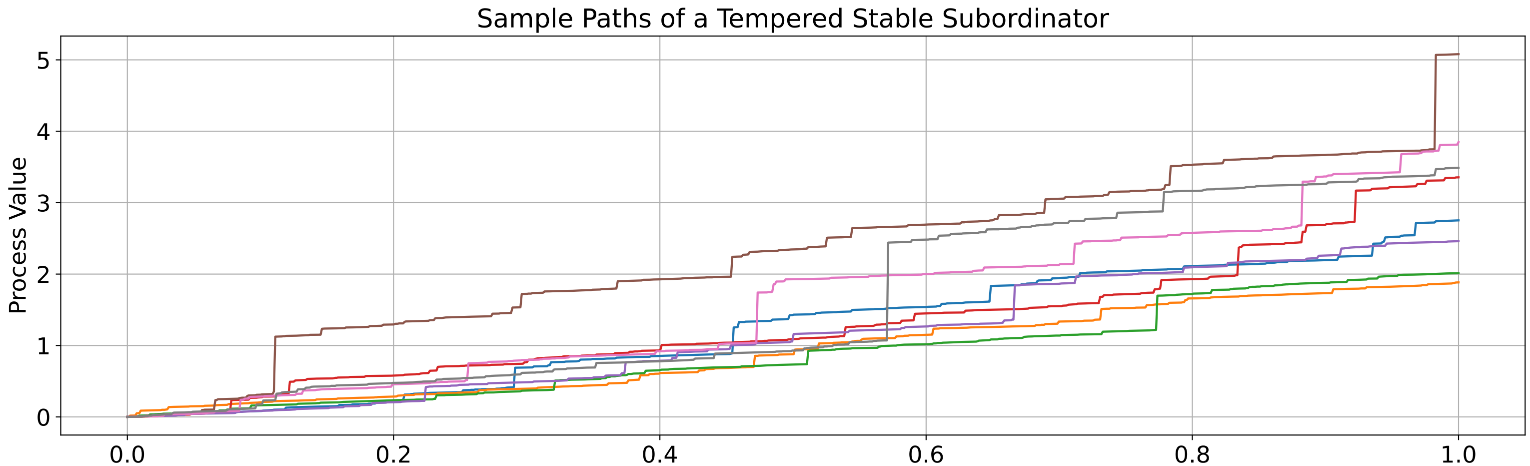
- We can obtain the un-tempered tail mass function of ν, denoted as ν_0^+, then

$$\nu_0^+(x) = \frac{C}{\alpha x^\alpha} \quad \text{and} \quad h(\Gamma) = \left(\frac{C}{\alpha \Gamma} \right)^{\frac{1}{\alpha}}$$

NGP – Tempered Stable (TS) Process Generation

Algorithm 2 Tempered stable (TS) process jumps generation and sampling algorithm

Initialize: $N = \varnothing$.
Generate the epochs Γ_i of a unit-rate Poisson process, i.e., exponential inter-arrival times.
for $i = 1, 2, 3, \cdots$ **do**
  Compute $x_i = h(\Gamma_i)$ using equation (3.4).
  Accept x_i with probability $\exp(-\lambda x_i)$.
  if x_i is accepted **then**
    Add x_i to the process N.
  end if
end for
For each accepted x_i, generate a jump time $V_i \sim \mathcal{U}(0, \tau)$.
Obtain the tempered stable subordinator using (3.2): $T_s = \sum_i x_i \mathbb{I}_{\{V_i \leq s\}}$.



NGP – Batch Inference

Given the observed data $\{x_i, y_i\}_{i=1}^{n}$,

posterior distribution over the latent subordinator process

$$p(f|\boldsymbol{y}_{1:n}) = \int p(f,T|\boldsymbol{y}_{1:n})\mathrm{d}T = \int p(f|T,\boldsymbol{y}_{1:n})p(T|\boldsymbol{y}_{1:n})\mathrm{d}T$$

$$\sim \mathcal{GP}(m_T, K_T)$$

$$= \frac{p(f,\boldsymbol{y}_{1:n}|T)}{p(\boldsymbol{y}_{1:n}|T)} = \frac{p(\boldsymbol{y}_{1:n}|f,T)p(f|T)}{p(\boldsymbol{y}_{1:n}|T)} \sim \mathcal{GP}(\overline{m_T}, \overline{K_T})$$

$$\sim \mathcal{GP}(\widetilde{m_T}, \widetilde{K_T})$$

- $p(\boldsymbol{y}_{1:n}|T)$ containing the *trade-off* between *data-fitness* and *model-complexity* reflecting *how well* the data is represented by the model given a random transformation T and can also be evaluated analytically by a Gaussian Process.

- Notice that the exact inference of $p(T|\boldsymbol{y}_{1:n})$ and thus $p(f|\boldsymbol{y}_{1:n})$ is analytically intractable due to the nonlinearity introduced by the latent transformation.

NGP – Batch Inference

- Motivated by the MCMC Metropolis-Hastings (MH) Algorithm, the Gibbs Sampler and the Birth-Death-Move (BDM) Sampler, we adopt a *Gibbs sampling scheme* for approximating samples from the posterior distribution $p(T|\boldsymbol{y}_{1:n})$.
- The input space $\mathcal{X}$ is partitioned into small disjoint intervals $\tau = (x_j, x_l)$.

Algorithm 3 Adapted BDM sampler procedure for subordinator T from $T^{(k)}$ to $T^{(k+1)}$ in τ.

1: Removing the current jump points in τ from the current sample $T^{(k)}$.
2: Generating a new set of jump locations and magnitudes $\{V_i', M_i'\}$ within τ, with the expected number of jumps governed by the Lévy intensity and the length $\|x_j - x_l\|$. The simulation can be obtained by using Algorithm 2.
3: Proposing a new sample path T' by replacing the points in τ using equation (3.2).
4: Accepting T' using a Metropolis-Hastings acceptance criterion based on the conditional posterior, and the outcome is the requested $T^{(k+1)}$.

- Iterating this procedure across all intervals of $\mathcal{X}$, and repeating over multiple sweeps, yields a Markov chain over subordinator paths whose stationary distribution approximates $p(T|\boldsymbol{y}_{1:n})$
- The acceptance probability

$$\alpha(T', T^{(k)}) = \min\left\{1, \frac{p(y_{1:n}|T')}{p(y_{1:n}|T^{(k)})}\right\}$$

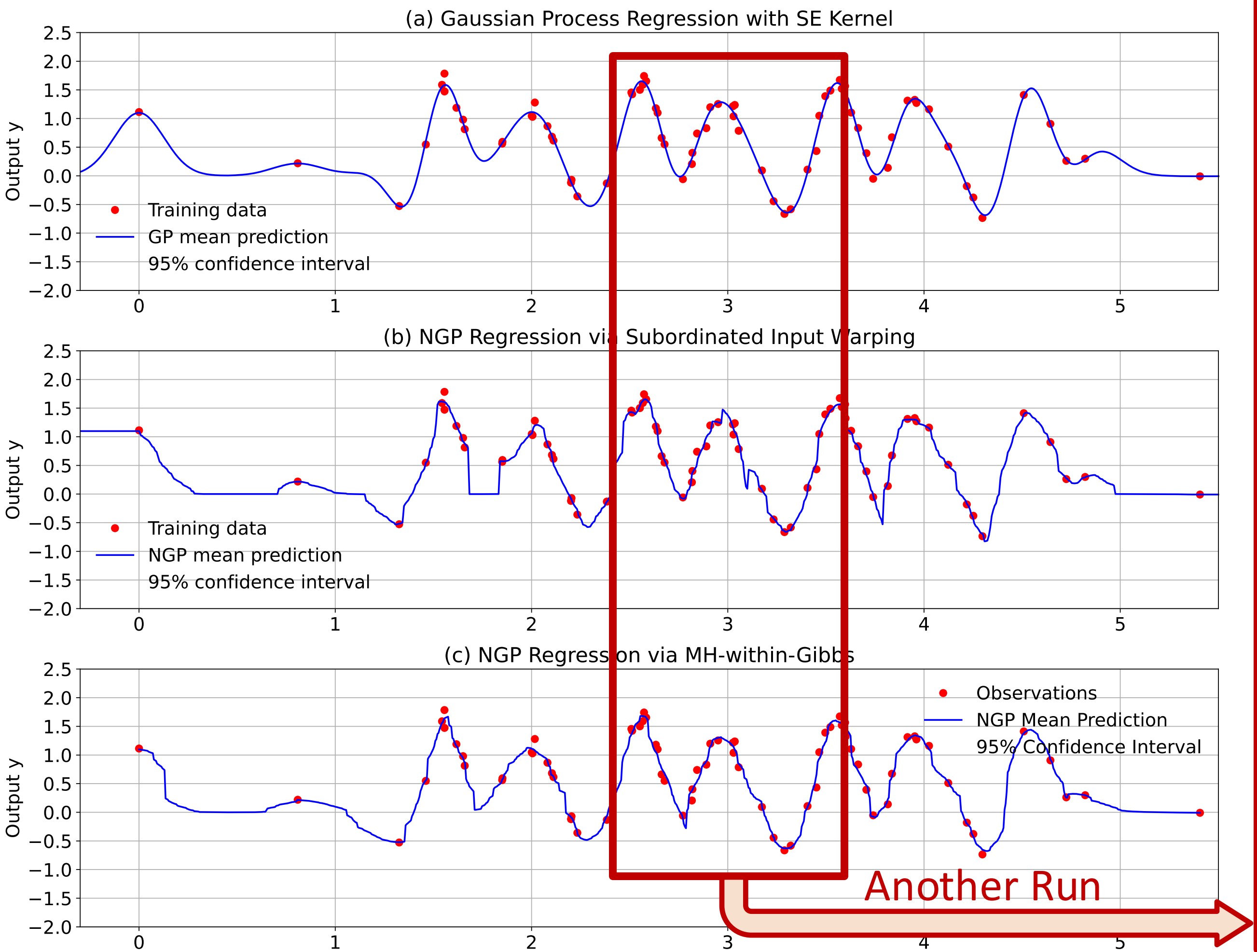
NGP – Batch Inference

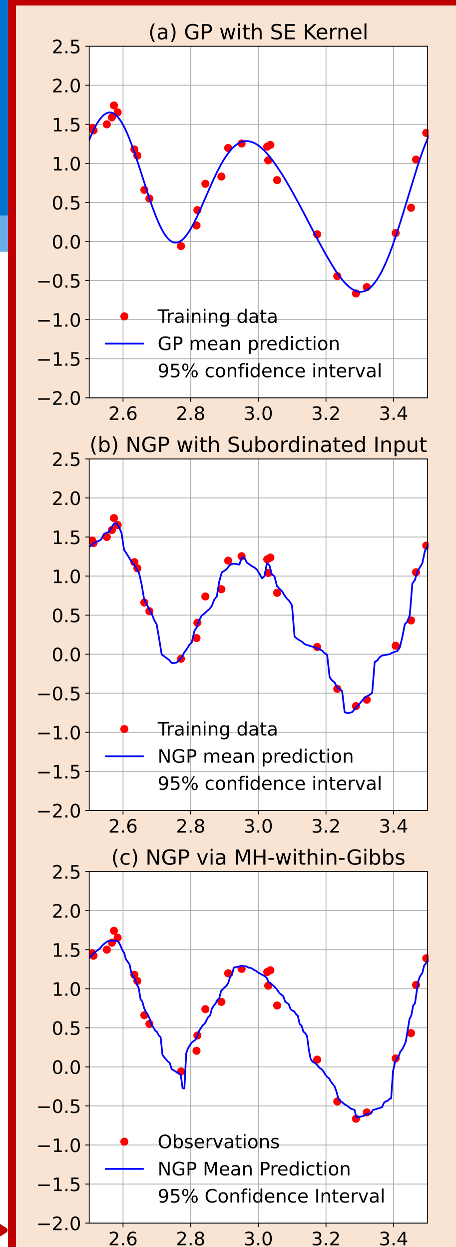
- The overall MH-within-Gibbs sampling procedure:

Algorithm 4 MH-within-Gibbs sampler for $p(T|\boldsymbol{y}_{1:n})$.

1: Initialise $T^{(0)}$ by simulating $\{V_i, M_i\}$ from the associated bivariate point process using Alg. 2.
2: Analytically evaluate $\bar{m}_{T^{(0)}}, \bar{K}_{T^{(0)}}$ which define the conditional GP posterior $p(f|\boldsymbol{y}_{1:n}, T^{(0)})$, and the conditional likelihood $p(y_{1:n}|T^{(0)})$.
3: **for** N times, iterate over $\tau_j \in \mathcal{X}$ where $\bigcup_{j=1}^{J} \tau_j = \mathcal{X}$, i.e., **do**
4: Sample a proposed sample path $T^{(\prime)}$ using Alg. 3, with τ_j and points $\{V_i^{(k)}, M_i^{(k)}\}$ with $T^{(k)}$.
5: Evaluate $\bar{m}_{T^{(\prime)}}, \bar{K}_{T^{(\prime)}}$ and $p(\boldsymbol{y}_{1:n}|T^{(\prime)})$.
6: Accept the proposal with probability $\alpha(T^{(\prime)}, T^{(k)})$, otherwise reject it and set $T^{(k+1)} = T^{(k)}$.
7: **end for**

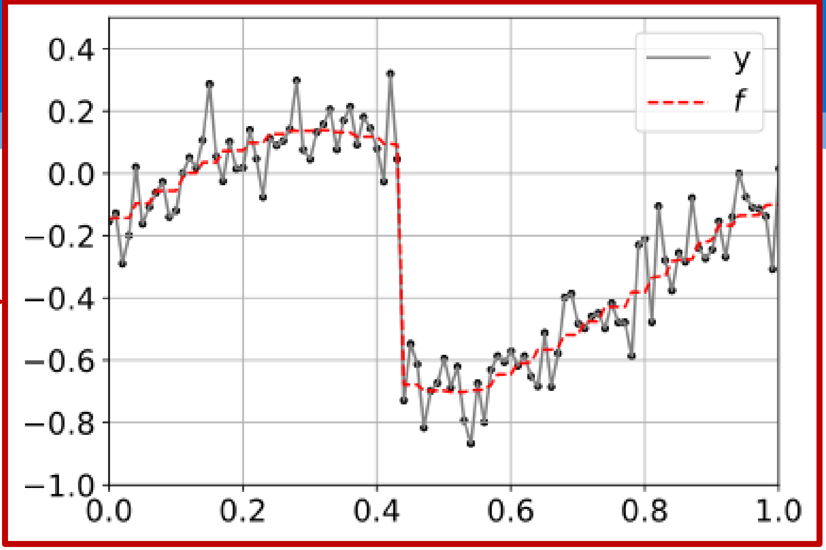
NGP – Regression Initial Outcome

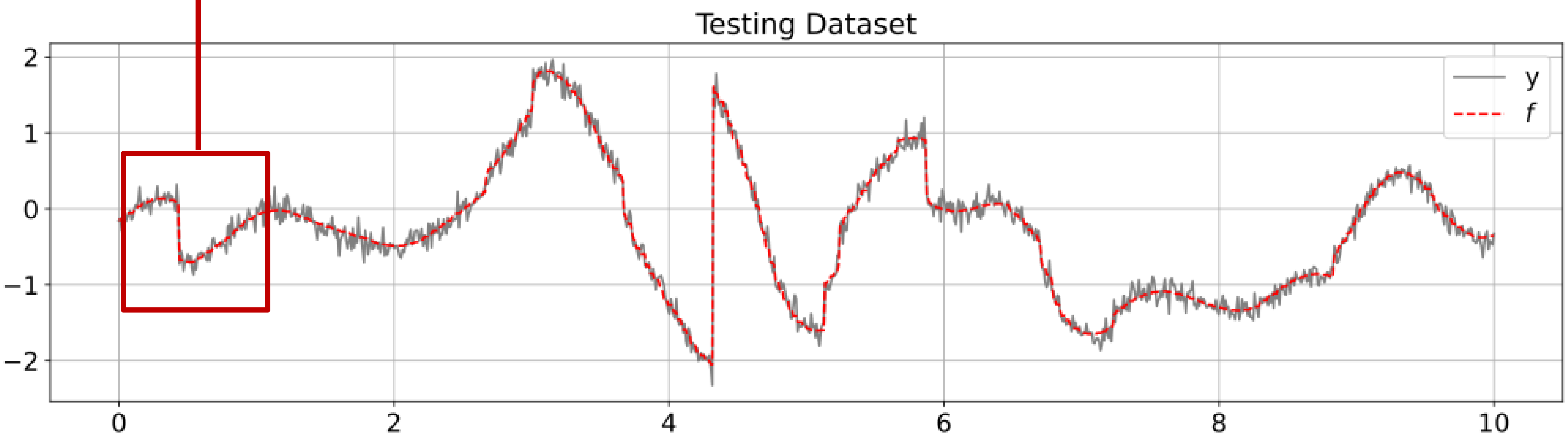




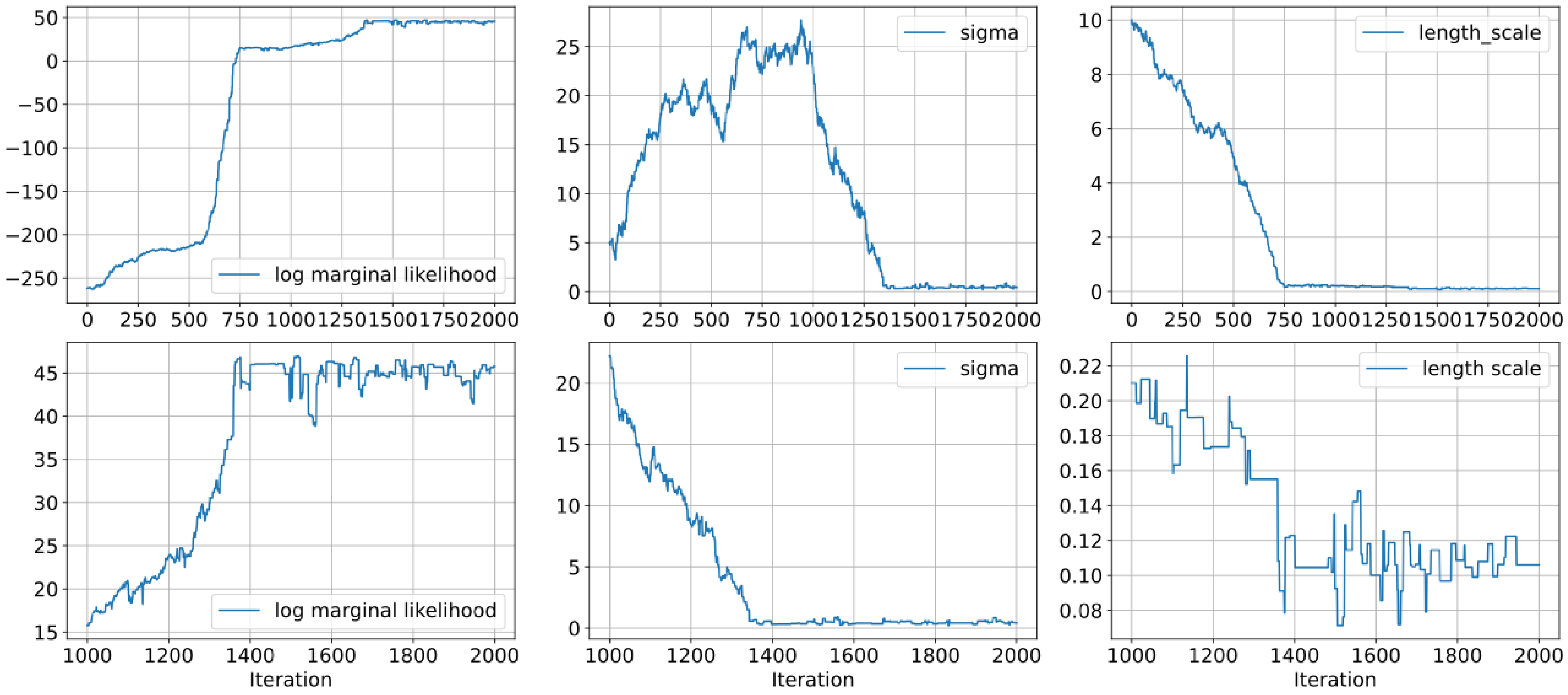
NGP Testing – Dataset0

Experiment Dataset



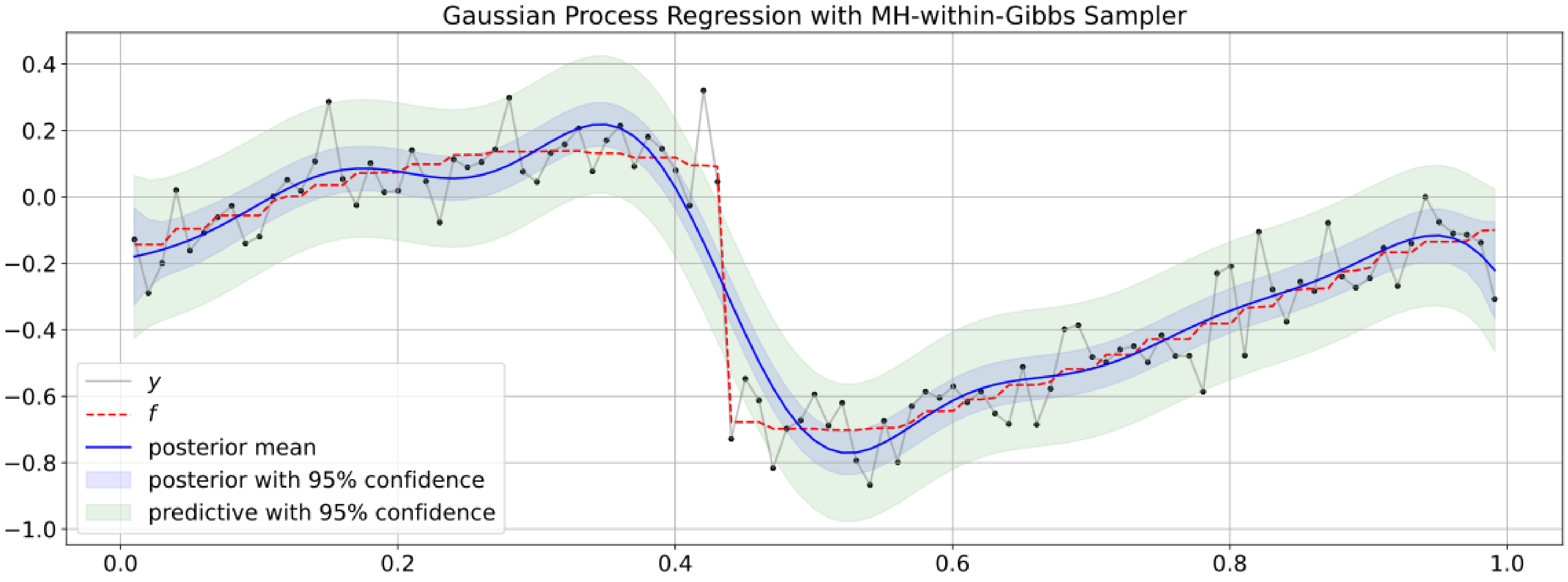


NGP Testing – GP Training with dataset0



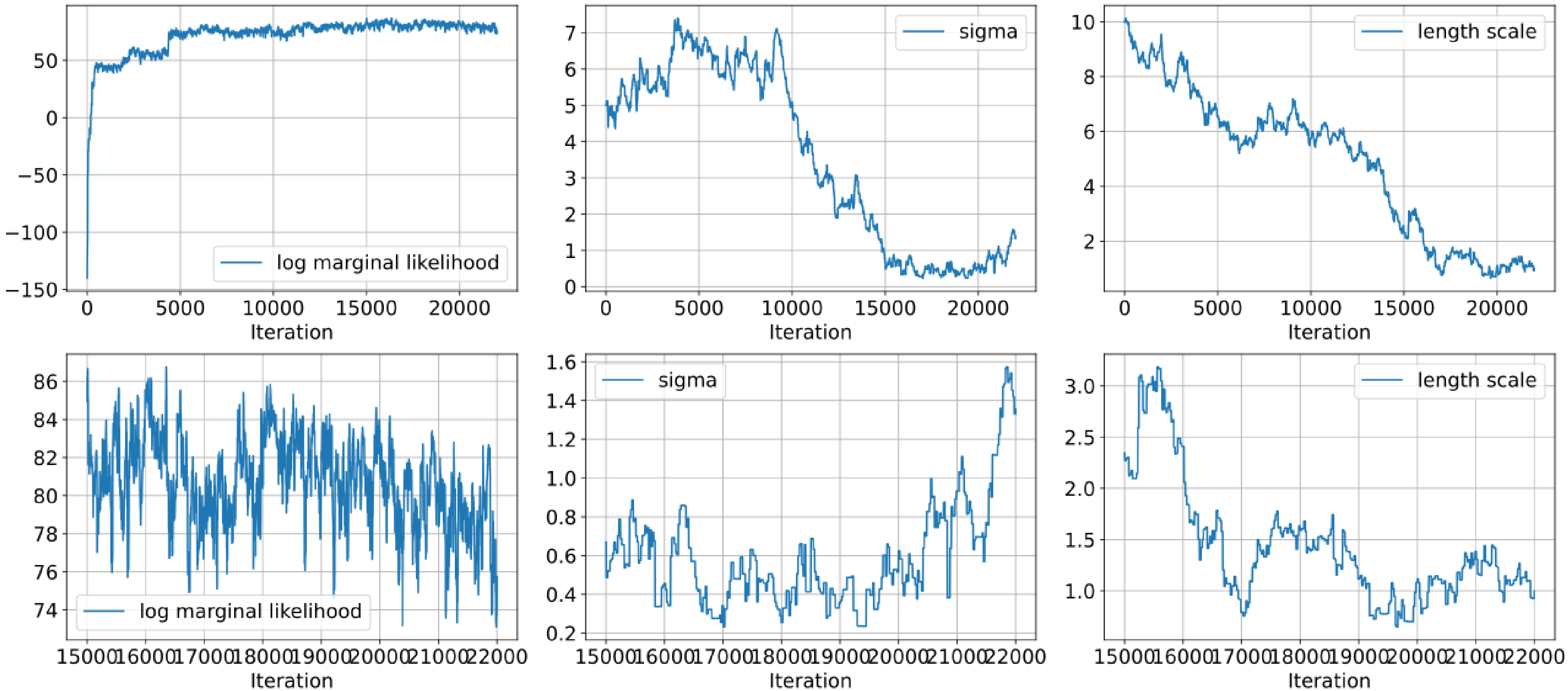
mean(log-marginal-likelihood) after burn-in: 37.35304737
mean(sigma) after burn-in: 3.9242498237457477
mean(length scale) after burn-in: 0.1336047253588278

NGP Testing – GP Regression with dataset0



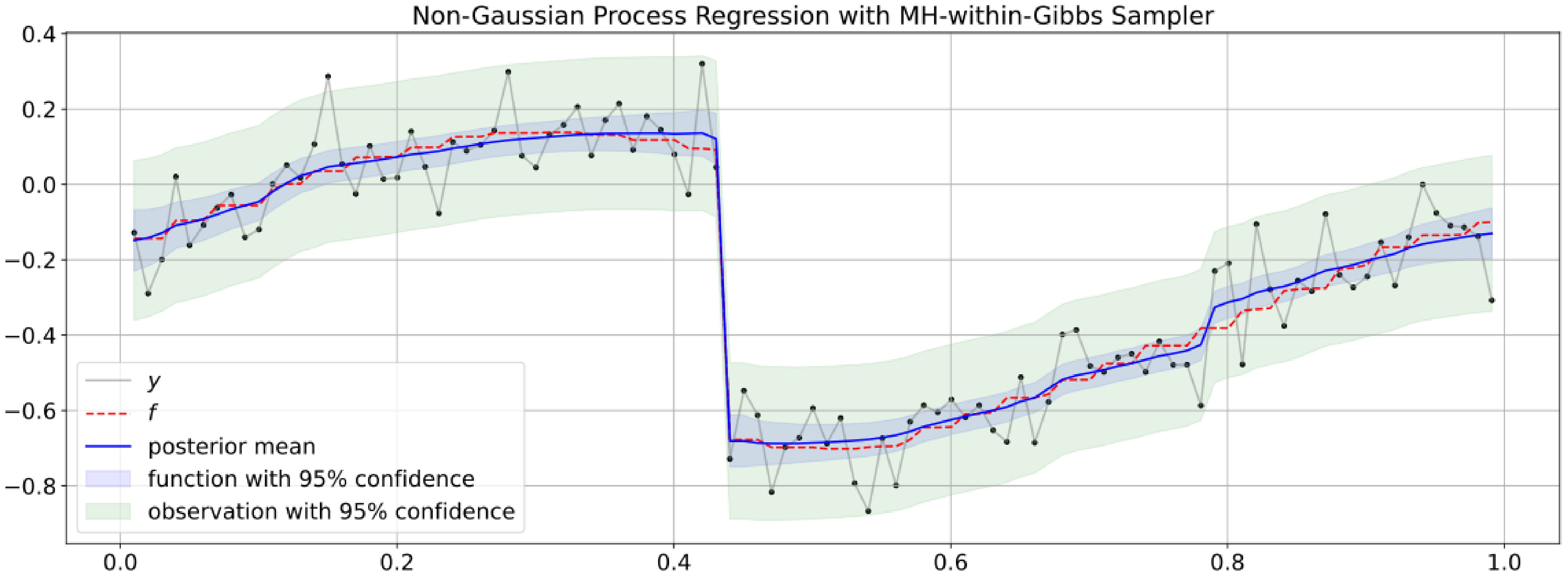
mean(log-marginal-likelihood) after burn-in: 37.35304737
mean(sigma) after burn-in: 3.9242498237457477
mean(length scale) after burn-in: 0.1336047253588278

NGP Testing – NGP Training with dataset0



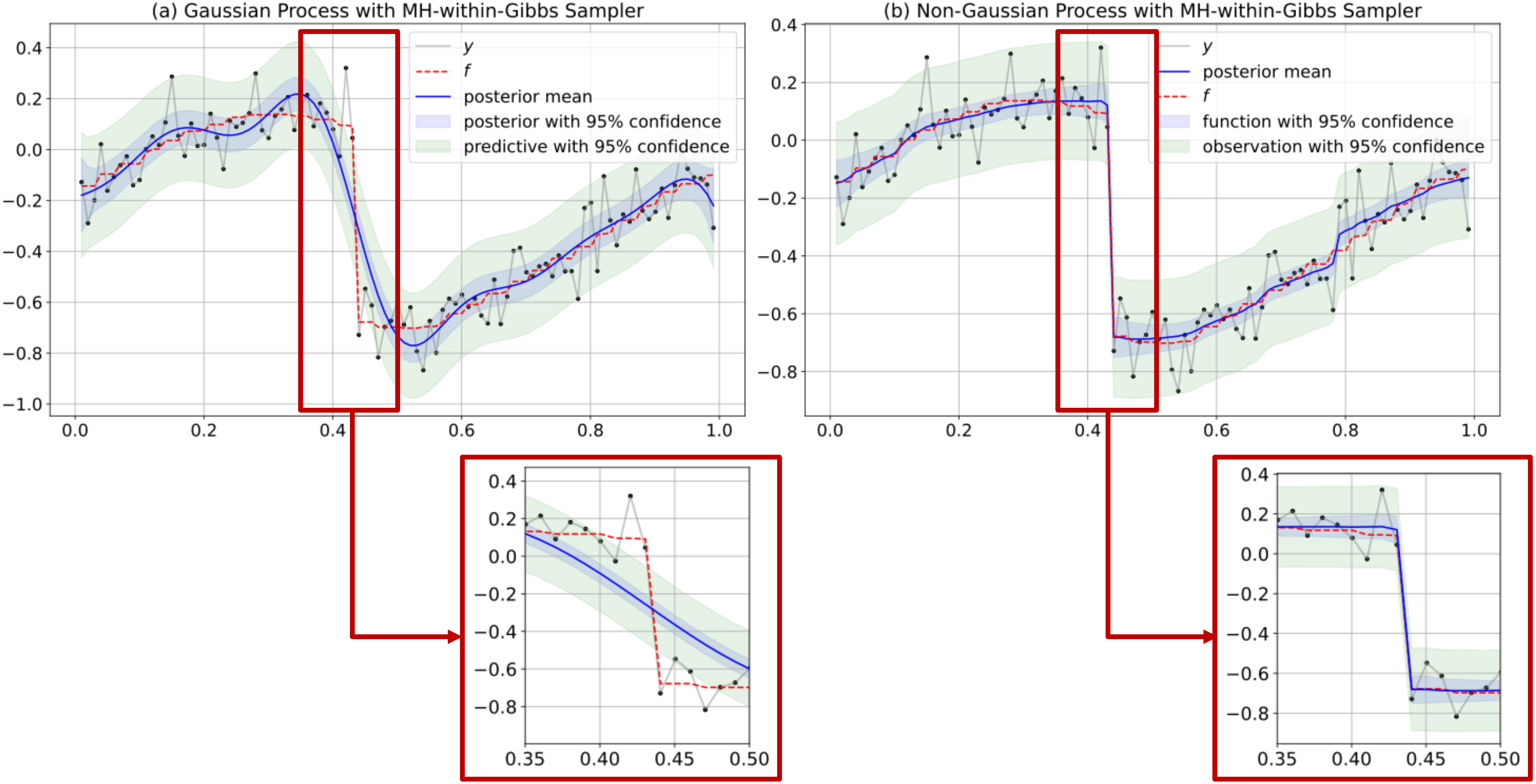
mean(log-marginal-likelihood) after burn-in: 80.68154388197864
mean(sigma) after burn-in: 0.588726
mean(length scale) after burn-in: 1.435514

NGP Testing – NGP Regression with dataset0

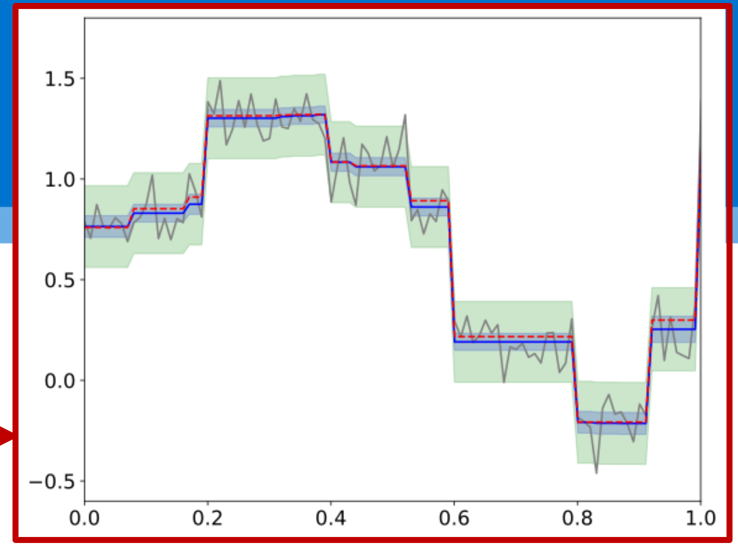


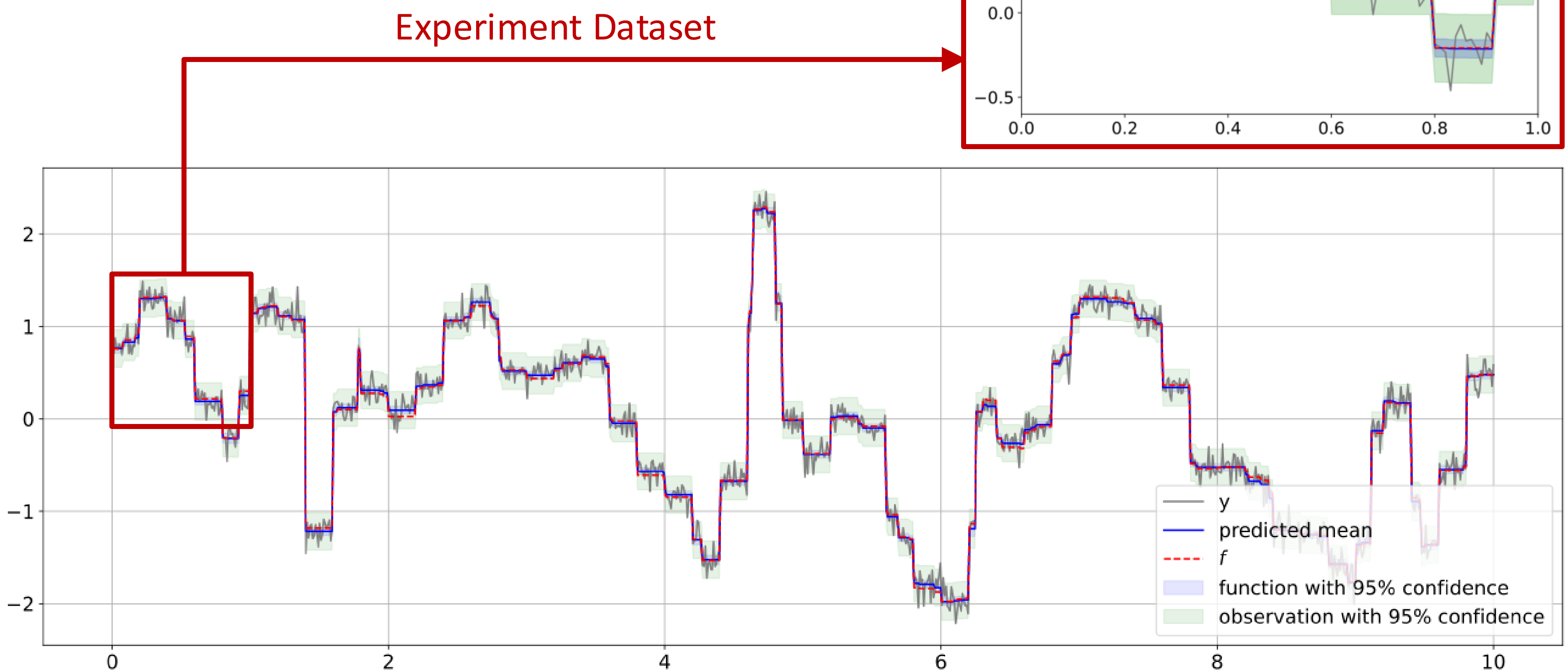
mean(log-marginal-likelihood) after burn-in: 80.68154388197864
mean(sigma) after burn-in: 0.588726
mean(length scale) after burn-in: 1.435514

NGP Testing – GP & NGP with dataset0



NGP Testing – Dataset4

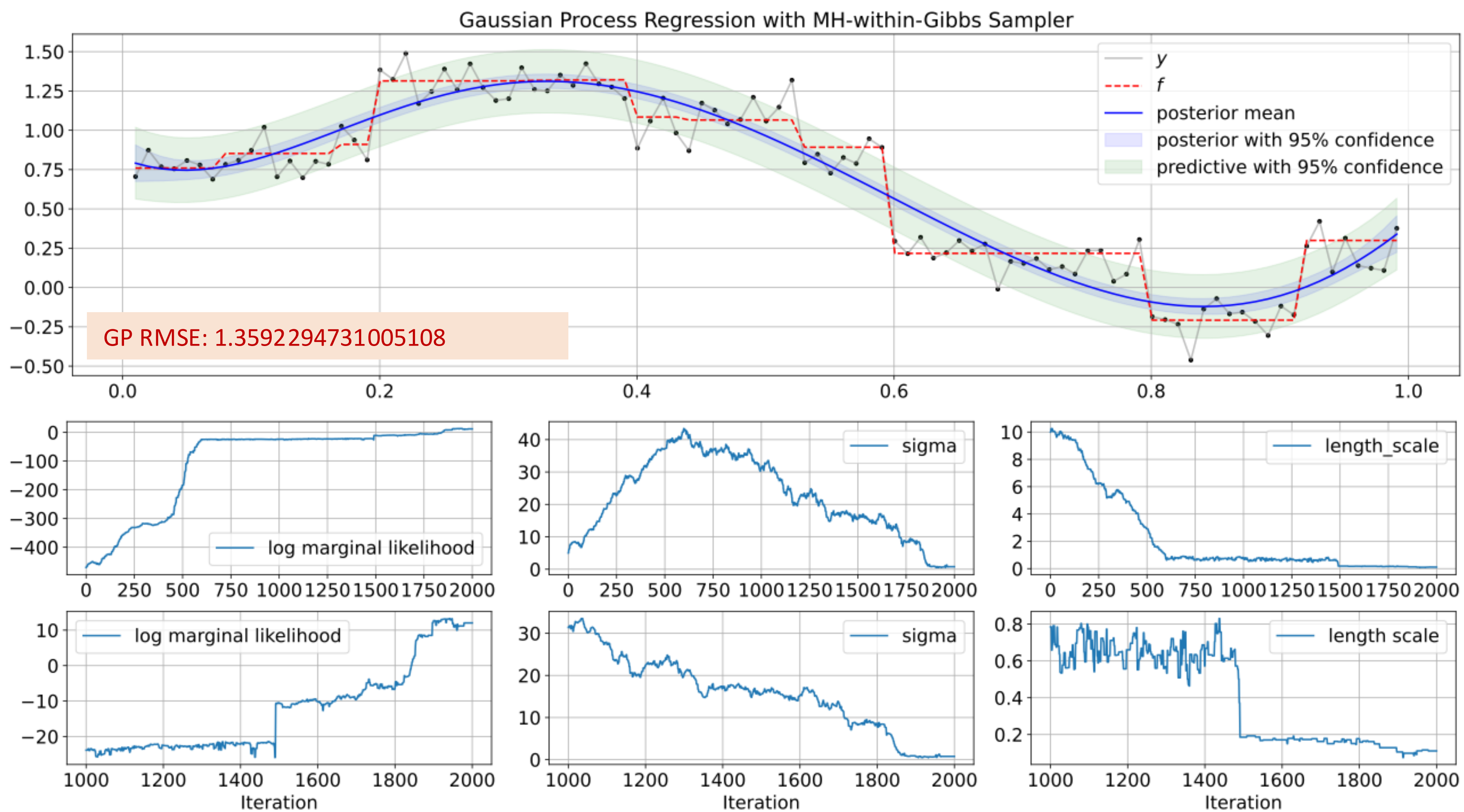




Log_marginal_likelihood: 698.6335686349726

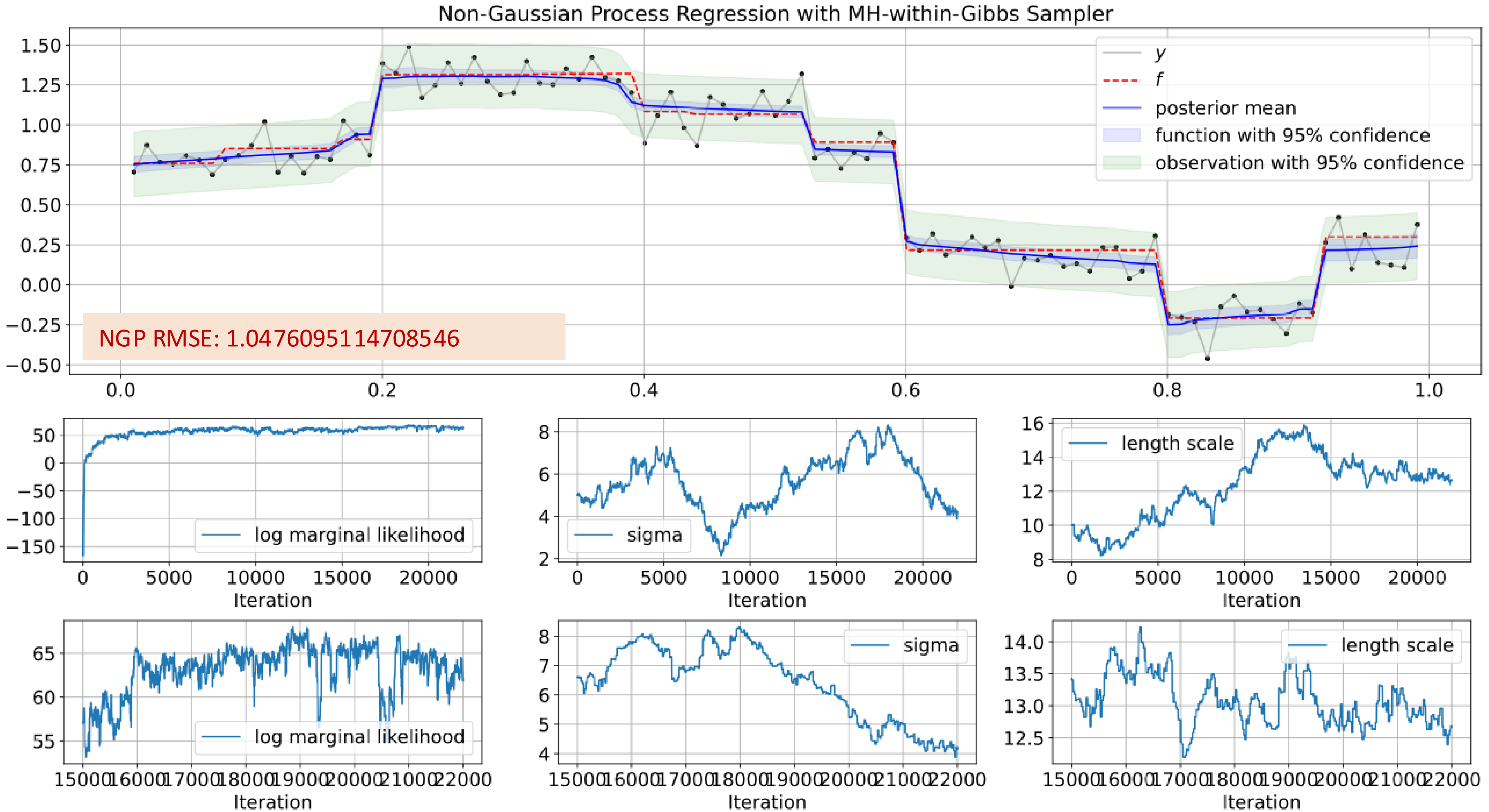
NGP Testing – GP with dataset4

sigma: 45.3558570805834
length_scale: 0.761634553446373
log ML: -25.06967249

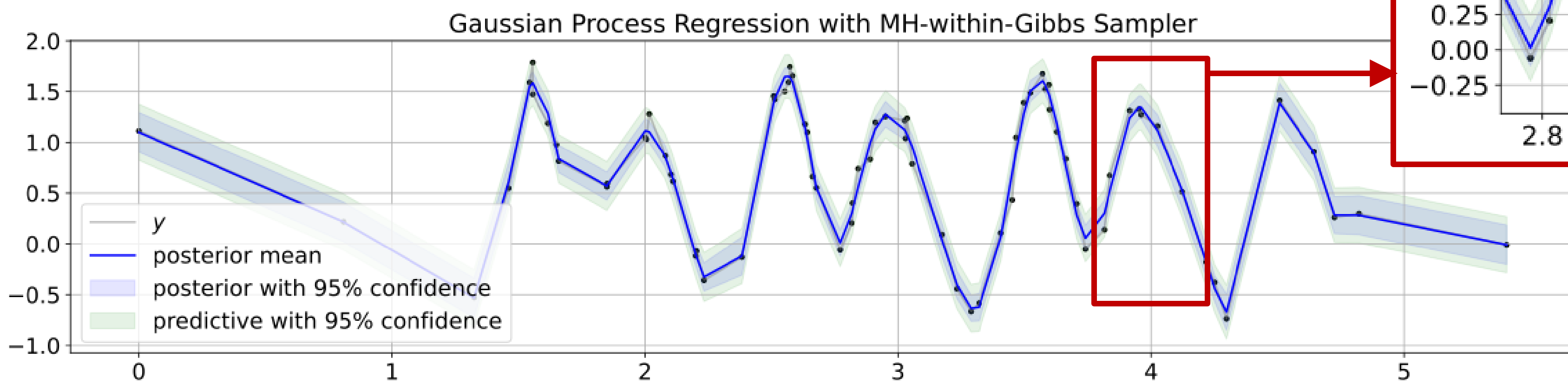


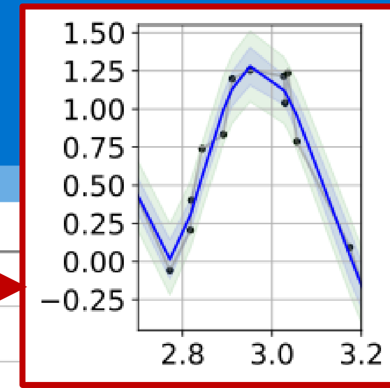
NGP Testing – NGP with dataset4

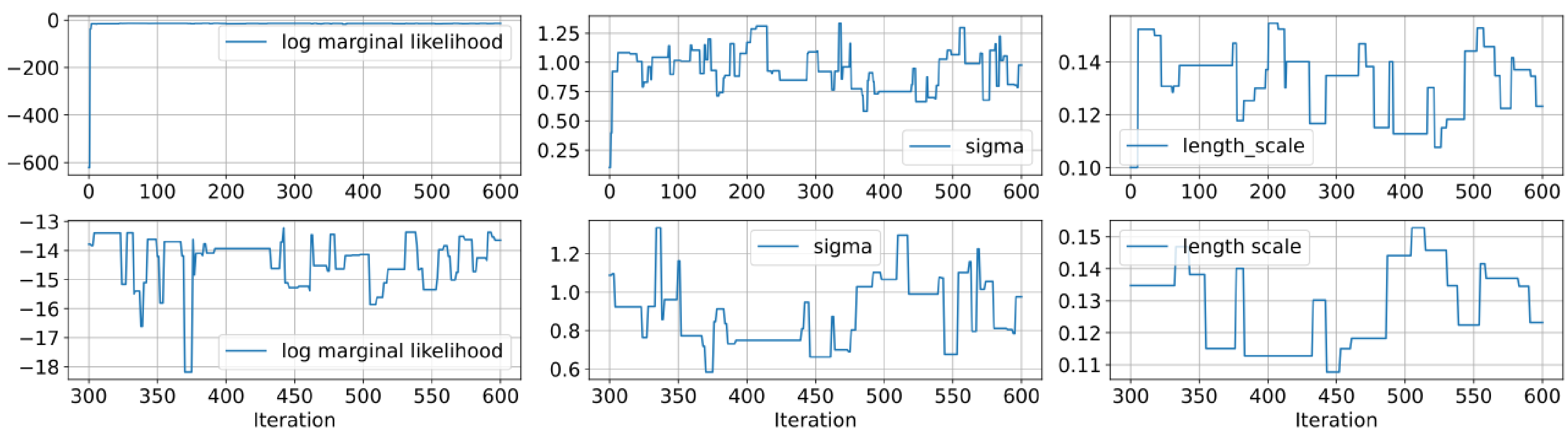
sigma: 6.408088
length_scale: 13.077311
log ML: 63.12223143123806



NGP Application – GP with MH-within-Gibbs



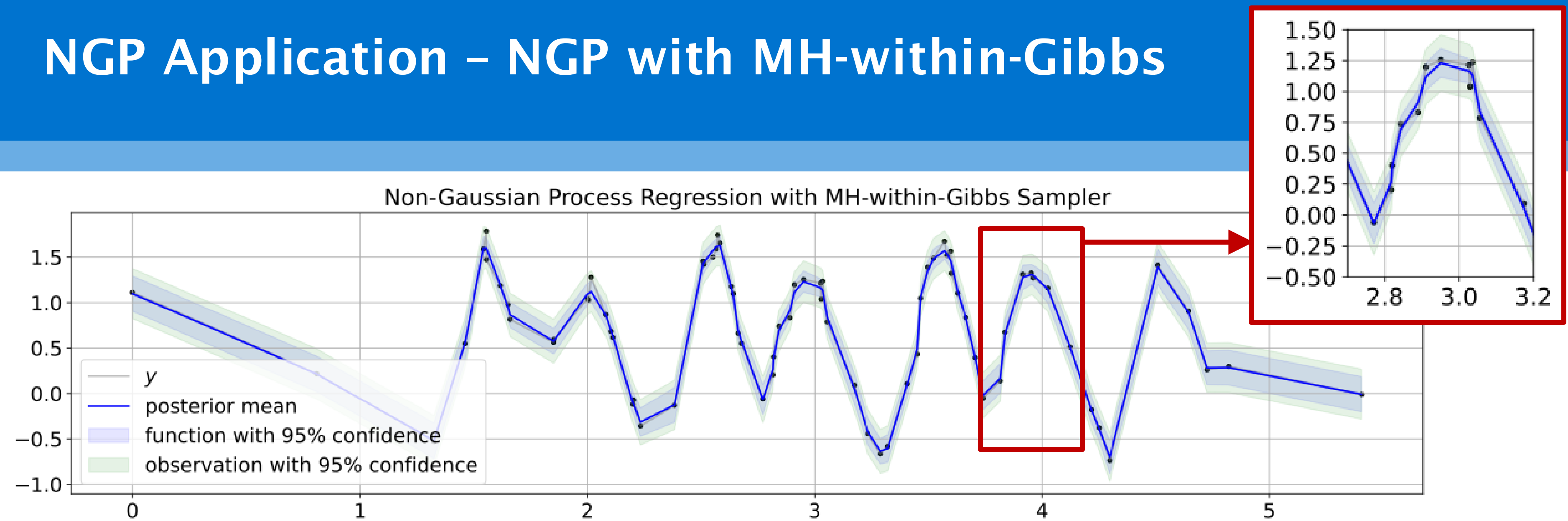


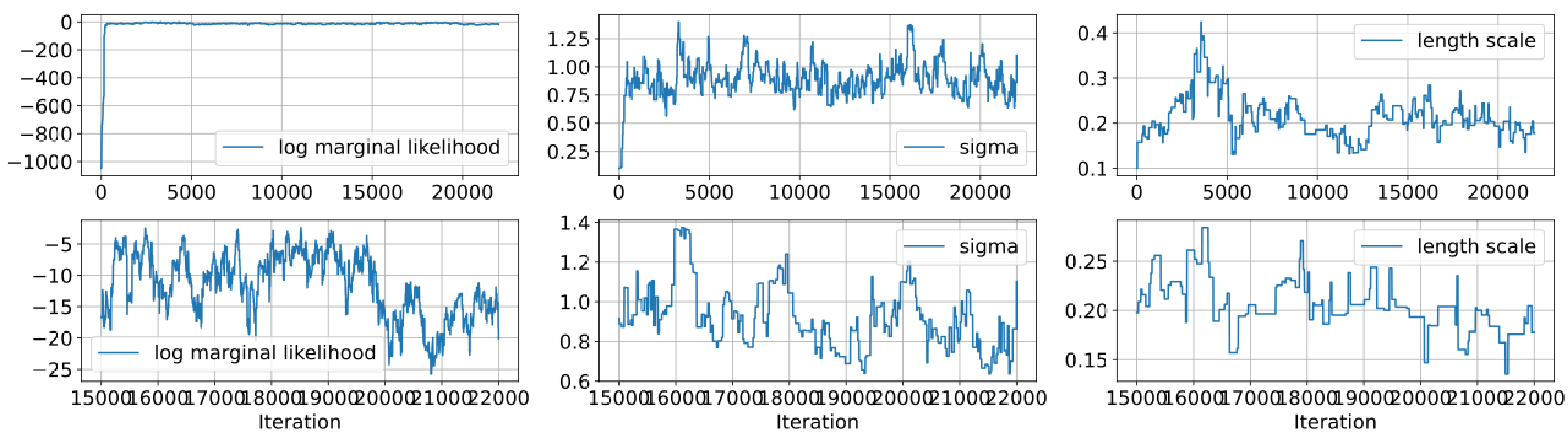


mean(log-marginal-likelihood) -14.35417134;
mean(length scale) after burn-in: 0.12829999647434565

mean(sigma) after burn-in: 0.8845499942841567

NGP Application – NGP with MH-within-Gibbs





mean(log-marginal-likelihood) -11.505880094890763; mean(sigma) after burn-in: 0.918708
mean(length scale) after burn-in: 0.205264

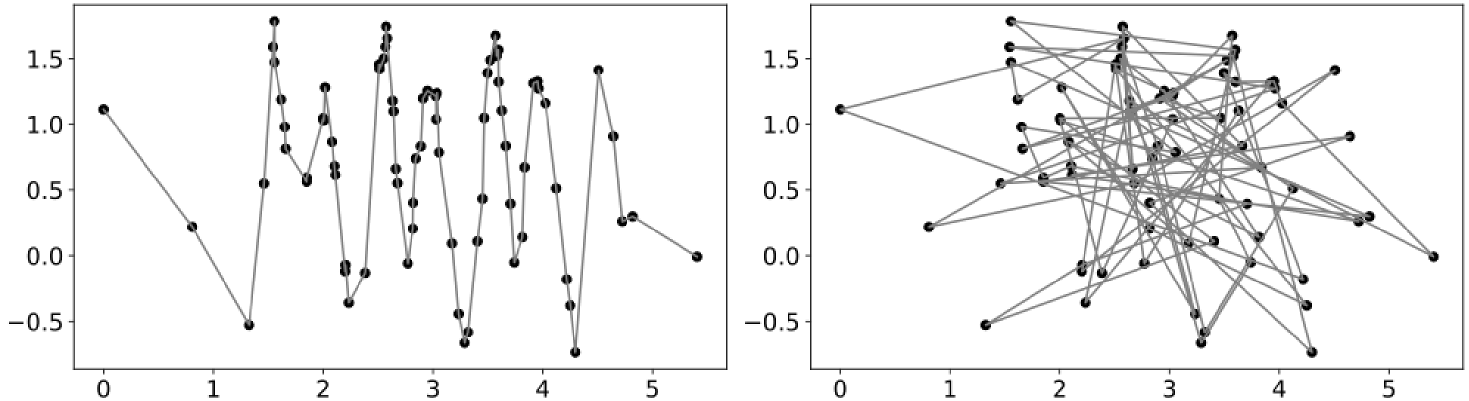
Problems & Future Work

Coding Part:

- DataFrame Saving
- Jupyter Notebook on Server

Next Reproduce:

- Parameter Estimation (learning)
- Sequential Data – tracking objects (on different datasets)



Method Part:

- Gamma Process?
- Residual Approximation Mode