Using Gaussian Processes for Emulation and Calibration

Canadian Celebration of Women in Computing West 2025

Ravleen Bajaj

Simon Fraser University

November 7, 2025

Table of Contents

What is a Computer Model?

- The goal of computational models is to replicate the behavior of real-world physical systems and to do so based on actual, known properties of the system components.
- We can investigate a process by adjusting the input parameters in a simulator and observing the response.
- A computer model streamlines the process of studying a complex physical system by reducing complexities based on time, money, labor, etc. required to **actually** observe a response on a given input simply by simulating it.

Examples

- **Environmental Sciences**
  - Simulators can model the spread and effects of pollutants in the environment, such as water contamination, which can help in constructing a framework for geomorphic risk maps for river migration to assess potential risks to human health and ecosystems **E.G.: [Noh, B., Wani, O., et al.]**

- **Astronomy**
  - Simulators can play a role in performing inference over the structure of black hole accretion discs and hence inform the development of future accretion disk theories. **E.G.: [Griffiths, Ryan-Rhys]**

Challenges

Computer models:

- are often high-dimensional, because they possess large numbers of input parameters;
- can have deterministic or stochastic outputs.

Due to inherent complexity, a single simulation run may take **hours**, if not days or weeks to run. This makes them very difficult to run at every iteration of Markov Chain Monte Carlo (MCMC).

Thus, performing a full uncertainty analysis of model behavior – a critical part of any scientific study that requires model evaluations in a wide variety of parameter combinations – may be unfeasible.

Modeling a model

What do we do now?

We run the simulation a limited number of times and use outputs from those runs to model the computer model/simulator

i.e., we EMULATE

Emulators

- **Emulation:** Given simulation outputs at a set of tried input settings, estimate the output at new, untried input settings.

- **Emulator:**
  - An emulator is a surrogate model which we use in place of a computer model/simulator.
  - It helps us predict uncertainty at unobserved inputs.

- Gaussian Process Regression is commonly used as an emulator model.

What is a Gaussian Process(GP)?

- A stochastic process $\{X_t;\ t \in T\}$ is said to be a **Gaussian Process** if and only if for every finite set of indices $t_1, \ldots, t_k \in T$, the random vector

$$\mathbf{X}_{t_1,\ldots,t_k} = (X_{t_1}, \ldots, X_{t_k})$$

has a multivariate Gaussian distribution.

What is a Gaussian Process(GP)?

- A stochastic process $\{X_t;\ t \in T\}$ is said to be a **Gaussian Process** if and only if for every finite set of indices $t_1, \ldots, t_k \in T$, the random vector

$$\mathbf{X}_{t_1,\ldots,t_k} = (X_{t_1}, \ldots, X_{t_k})$$

  has a multivariate Gaussian distribution.
- Informally, a GP is a (potentially infinte) collection of random variables (RV) such that the joint distribution of every finite subset of those RVs is multivariate Gaussian.

Mathematical Notation of a GP

A GP is a stochastic process which is fully described by a mean function $\mu : \mathcal{X} \to \mathbb{R}$ and covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that

$$f_{\mathrm{GP}}(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right) \tag{1}$$

with $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, with mean vector $\mu(x)$ and covariance matrix as $\Sigma = k(x, x')$.

The covariance function is a measure for the correlation of two states $(\mathbf{x}, \mathbf{x}')$ and is often referred to as *kernel* in the GP literature.

Covariance Kernels

We must specify the form of the kernel based on our prior beliefs about the smoothness of the underlying function we want to model.

Some of the most commonly used kernels are:

- **RBF or Squared Exponential Kernel:**

$$k(x, x') = \sigma^2 \cdot \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right)$$

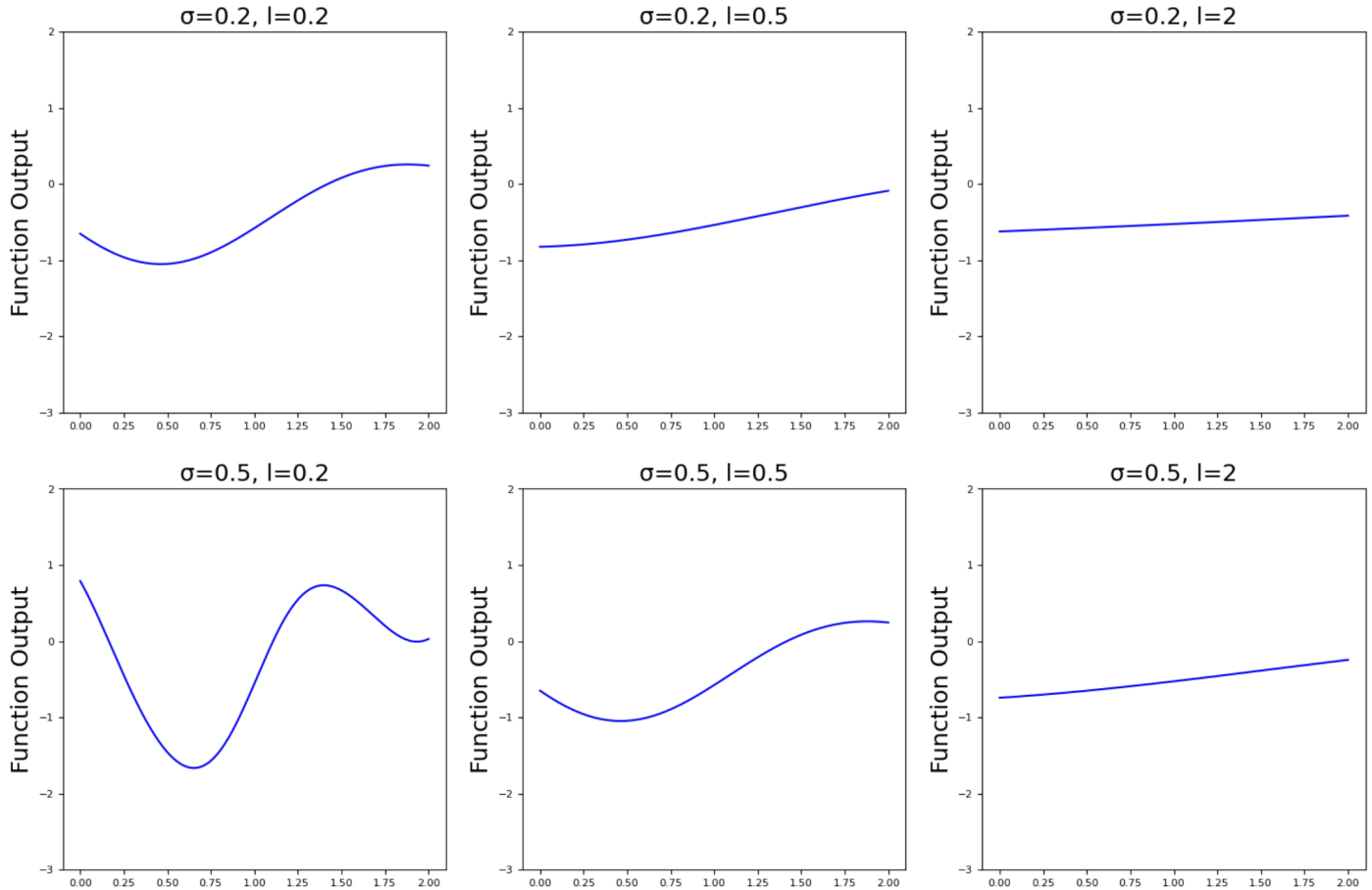
- **Matern Kernel with different amplitudes:**

$$k(x, x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{\ell}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu} r}{\ell}\right)$$

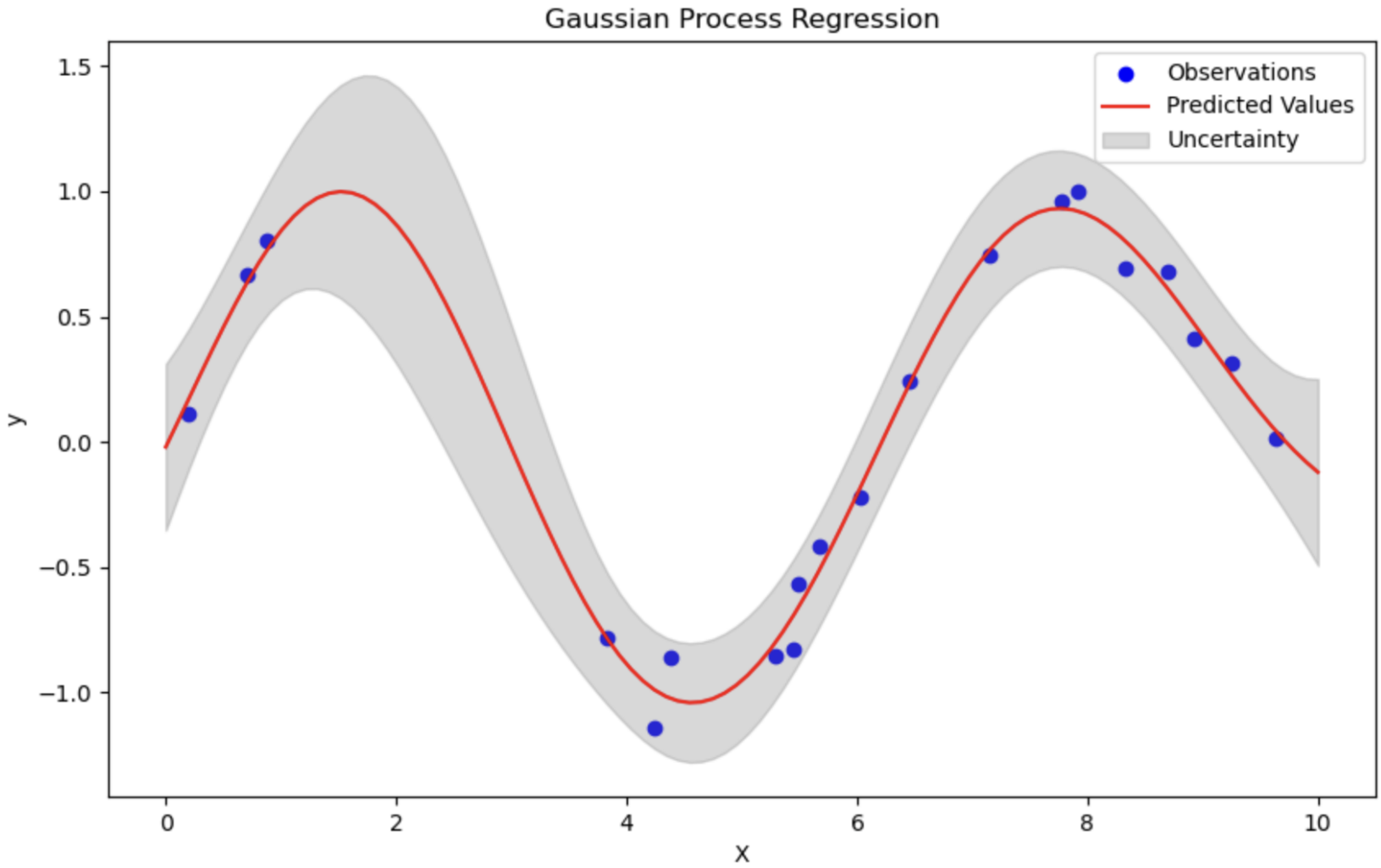
Here, σ^2 is the scale parameter and ℓ is the length scale which controls 'wiggliness'.

Effect of σ^2 and ℓ

- Here, we use the RBF kernel to plot the outputs of a single realization with different choices of σ^2 and ℓ.



Visualization of a GP



An Example

First, we specify a set of inputs to evaluate the computer model:

$$\text{Here, } \boldsymbol{x_i} = (\boldsymbol{x_{i1}}, \boldsymbol{x_{i2}})$$

At various settings for x, observations y are made of the physical system:

$$y(\boldsymbol{x_i}) = \zeta(\boldsymbol{x_i}) + \epsilon(\boldsymbol{x_i}), i = 1, \cdots, n,$$

Statistical Model:

$$y_i = \eta(\boldsymbol{x_i}, \theta) + \epsilon_i \text{ where } \epsilon_i \sim N(0, 0.25^2)$$

and $\boldsymbol{\theta}$ to denote the 'best' or 'true' value of the emulation parameters, which is a quantity about which we wish to infer.

True Function:

$$\boldsymbol{x} = \sin(x_1) + \cos(x_2)$$

An Example

Covariance Function:

Based on prior knowledge, we define: $\alpha = 2$ and $n = 16$

$$Cov(x, x') = \sigma = \frac{1}{\lambda_\eta}\left(exp\{-\sum_{k=1}^{2} \beta_i \|x_{ik} - x'_{ik}\|^\alpha\}\right)$$

$$\Sigma = k(x, x') = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} & \cdots & \sigma_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \sigma_{n3} & \cdots & \sigma_{nn} \end{bmatrix} \tag{2}$$

Mean Function:

We generally, use a constant mean function.

Here, we take: $\mu(\boldsymbol{x}) = 0$

An Example

Likelihood Function of a GP:

$$L(y \mid \lambda, \beta) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{\frac{-y^T \Sigma^{-1} y}{2}}$$

Posterior (Likelihood times Prior):

$$\pi(\lambda, \beta \mid y) \propto L(y \mid \lambda, \beta, \Sigma_y)\, \pi(\lambda)\, \pi(\beta)$$

$$\log L(\beta, \lambda \mid y) = \begin{cases} -\infty & \text{if } \lambda \leq 0 \text{ or } \beta_i \leq 0, \\ \log(y \mid \beta, \lambda) + \log(\beta) + \log(\lambda) & \text{otherwise.} \end{cases}$$

We use MCMC to draw samples from the posterior distribution of the parameters and get point estimates and prediction intervals

MCMC: Metropolis Algorithm

A very simple MCMC implementation uses the Metropolis algorithm and goes as follows:

1. Initialize θ^1 at some value.
2. Given the current realization is θ^t, generate θ^* using proposal distribution.
3. Compute the Metropolis acceptance probability

$$\alpha = \min\left\{1, \frac{\pi(\theta^*|y)}{\pi(\theta^t|y)}\right\}.$$

4. Set

$$\theta^{t+1} = \begin{cases} \theta^* & \text{with probability } \alpha, \\ \theta^t & \text{with probability } 1-\alpha. \end{cases}$$

5. Iterate over steps 2–4.

Acceptance Rates

Our evaluated posterior acceptance rates are evaluated to be as follows:

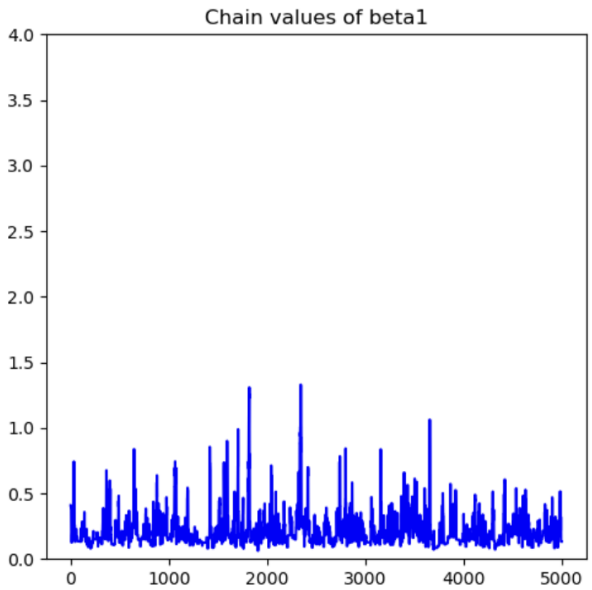
- $\beta_1 = 0.3244$
- $\beta_2 = 0.3066$
- $\lambda = 0.3198$

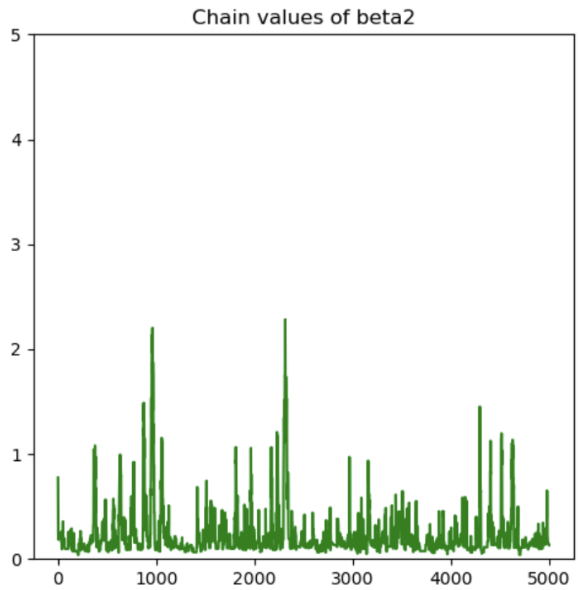
These align well with 'Robert's Thumb Rule'.

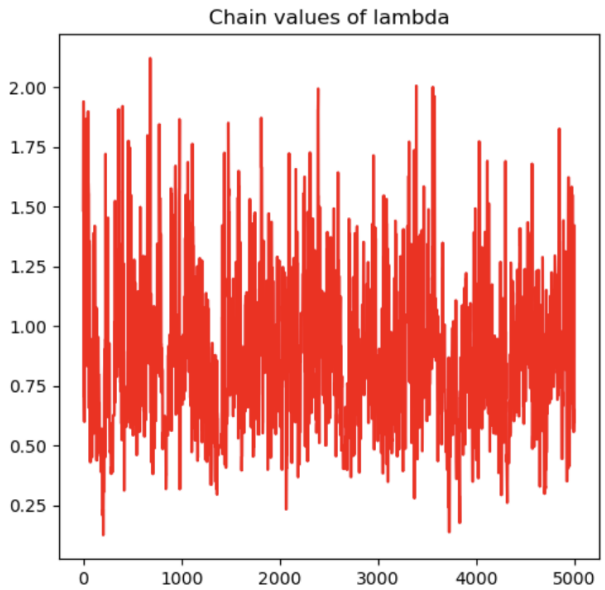
For a random walk Metropolis algorithm in high-dimensional problems, the scale of the proposal distribution should be tuned so that the average acceptance rate is approximately 23.4%. Realistically, we can get observe good mixing in chains with acceptance rates lying between $20\% - 35\%$.

Trace Plots

In an MCMC emulator setting, chain plots (or trace plots) help assess the convergence and mixing of MCMC chains, indicating whether the chains have explored the target distribution.

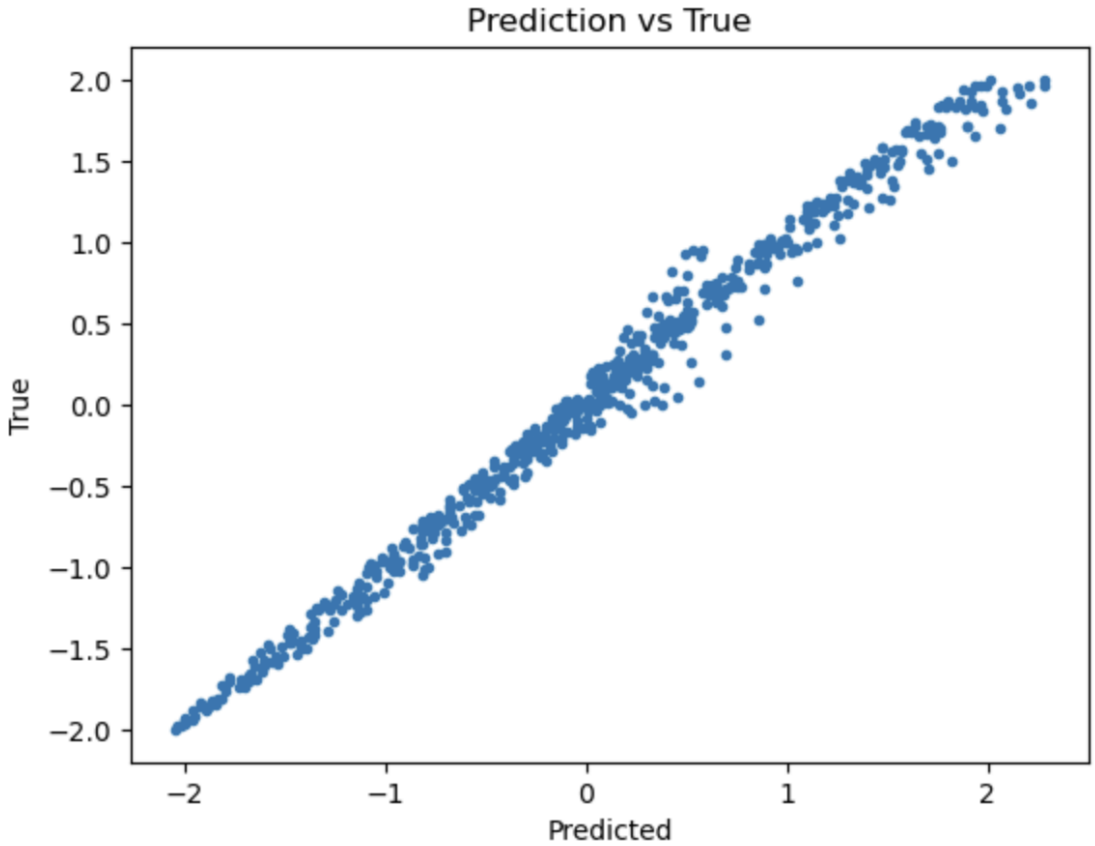






Results (Prediction vs True Plot)

Finally, we generate new values of the true function and plot them against the corresponding predicted values.



This indicates a strong positive correlation between the predicted and true values.

Calibration Setup

- **Calibration** helps to determine the variation in simulation outputs due to uncertainty or changes in input settings.
- We use the computer model output to quantify uncertainty around real-world (field) observations.
- Let $\mathbf{x}$ denote the controllable inputs and $\boldsymbol{\theta}$ the calibration parameter.
- The field observations are modeled as:

$$y_F(\mathbf{x}) = \eta(\mathbf{x}, \boldsymbol{\theta}) + \delta(\mathbf{x}) + \epsilon,$$

where:

- $\eta(\mathbf{x}, \boldsymbol{\theta})$ is the emulator output,
- $\delta(\mathbf{x})$ is the model discrepancy,
- $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is the measurement noise.

Covariance Structure of Emulator

We define the joint covariance between simulation and field runs as:

$$Cov((x,t),(x',t')) = \frac{1}{\lambda_\eta} \exp\left(-\sum_{k=1}^{p} \beta_k |x_{ik} - x'_{ik}|^\alpha - \sum_{k'=1}^{l} \beta_{p+k'} |t_{ik'} - t'_{ik'}|^\alpha \right)$$

where p and l denote the dimensions of inputs and calibration parameters respectively, and we use $\alpha = 2$ which makes the kernel squared-exponential.

Interpretation:

- β_k and $\beta_{p+k'}$ are the scale parameters and control correlation strength for inputs and calibration parameters.
- λ_η is the length scale which controls the overall variability.

Covariance Matrices

Training covariance matrix:

$$\mathbf{K}_{ij} = \frac{1}{\lambda} \exp\left(-\sum_{k=1}^{p} \beta_k |B_{ik} - B_{jk}|^{\alpha} - \sum_{k=1}^{L} \beta_{p+k} |B_{i,p+k} - B_{j,p+k}|^{\alpha} \right)$$

Cross-covariance:

$$\mathbf{K}^{*}_{ij} = \frac{1}{\lambda} \exp\left(-\sum_{k=1}^{p} \beta_k |x^{*}_{ik} - B_{jk}|^{\alpha} - \sum_{k=1}^{L} \beta_{p+k} |x^{*}_{i,p+k} - B_{j,p+k}|^{\alpha} \right)$$

Covariance among field locations:

$$\mathbf{K}^{**}_{ij} = \frac{1}{\lambda} \exp\left(-\sum_{k=1}^{p} \beta_k |x^{*}_{ik} - x^{*}_{jk}|^{\alpha} - \sum_{k=1}^{L} \beta_{p+k} |x^{*}_{i,p+k} - x^{*}_{j,p+k}|^{\alpha} \right)$$

GP Prediction for Emulator

Given the training data $(\mathbf{B}, \mathbf{y_{sim}})$, we predict the emulator mean and variance at field locations:

$$\boldsymbol{\mu}^* = \mathbf{K}^* \mathbf{K}^{-1} \mathbf{y_{sim}}$$

$$\hat{\boldsymbol{\eta}}_{\boldsymbol{cov}} = \mathbf{K}^{**} - \mathbf{K}^* \mathbf{K}^{-1} (\mathbf{K}^*)^T$$

Hence:

$$\hat{\eta} \sim \mathcal{GP}(\mu^*, \hat{\eta}_{cov})$$

Log-Likelihood for Calibration

The calibration likelihood for field data $\mathbf{y_F}$ is given by:

$$\log \mathcal{L}(\theta) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{n}\log(\sigma_i^2) - \frac{1}{2}(y_F - \eta(\theta))^\top(\Sigma)^{-1}(y_F - \eta(\theta)). \tag{3}$$

This assumes:

$$y_{F,i}|\theta \sim \mathcal{N}(\eta(\theta), \Sigma) \tag{4}$$

where

$$\eta(\theta) \sim \mathcal{N}(\mu^*, \Sigma^*) \qquad \Sigma = \tfrac{1}{\lambda_\epsilon} I_n = \sigma^2 I_n$$

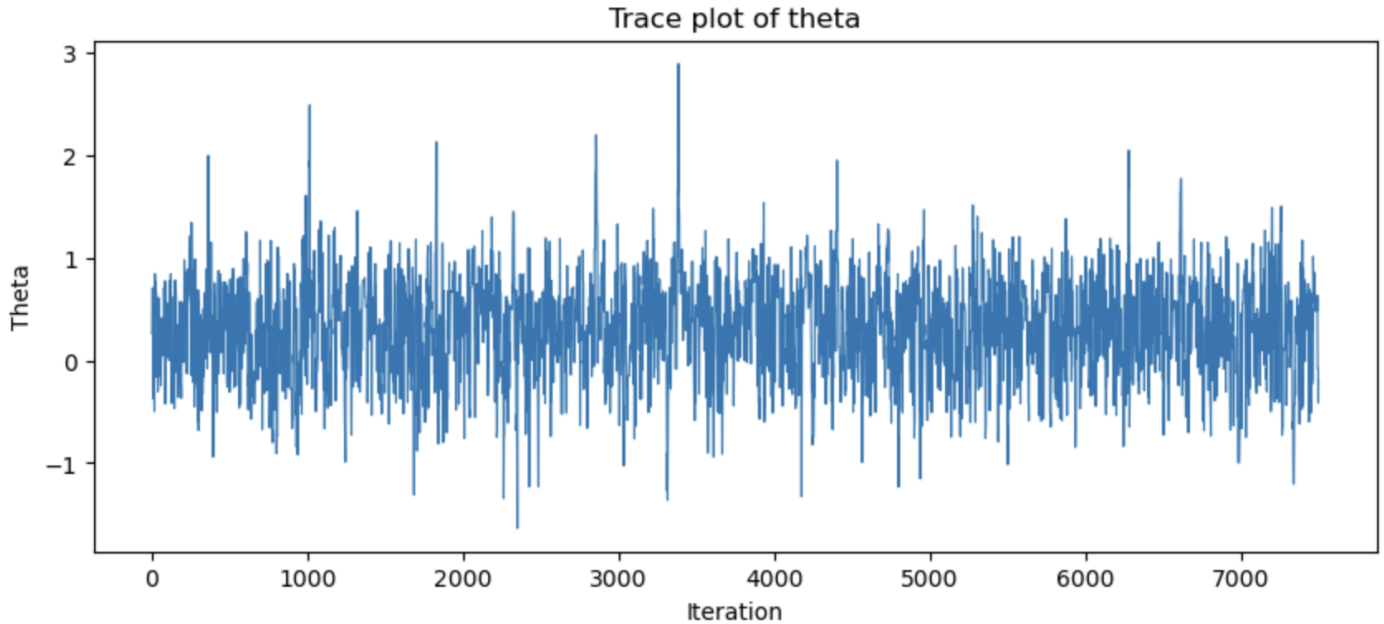
Posterior Inference

Posterior is proportional to likelihood times priors, the target posterior distribution is:

$$\pi(\theta|\mathbf{y}_F) \propto \mathcal{L}(\mathbf{y}_F|\theta)\pi(\theta) \tag{5}$$

where the likelihood $\mathcal{L}(\mathbf{y}_F|\theta)$ is computed using the GP emulator trained on simulation data, and the calibration uses only field observations.

- Samples drawn using MCMC to explore posterior space.
- Convergence assessed via trace and density plots.

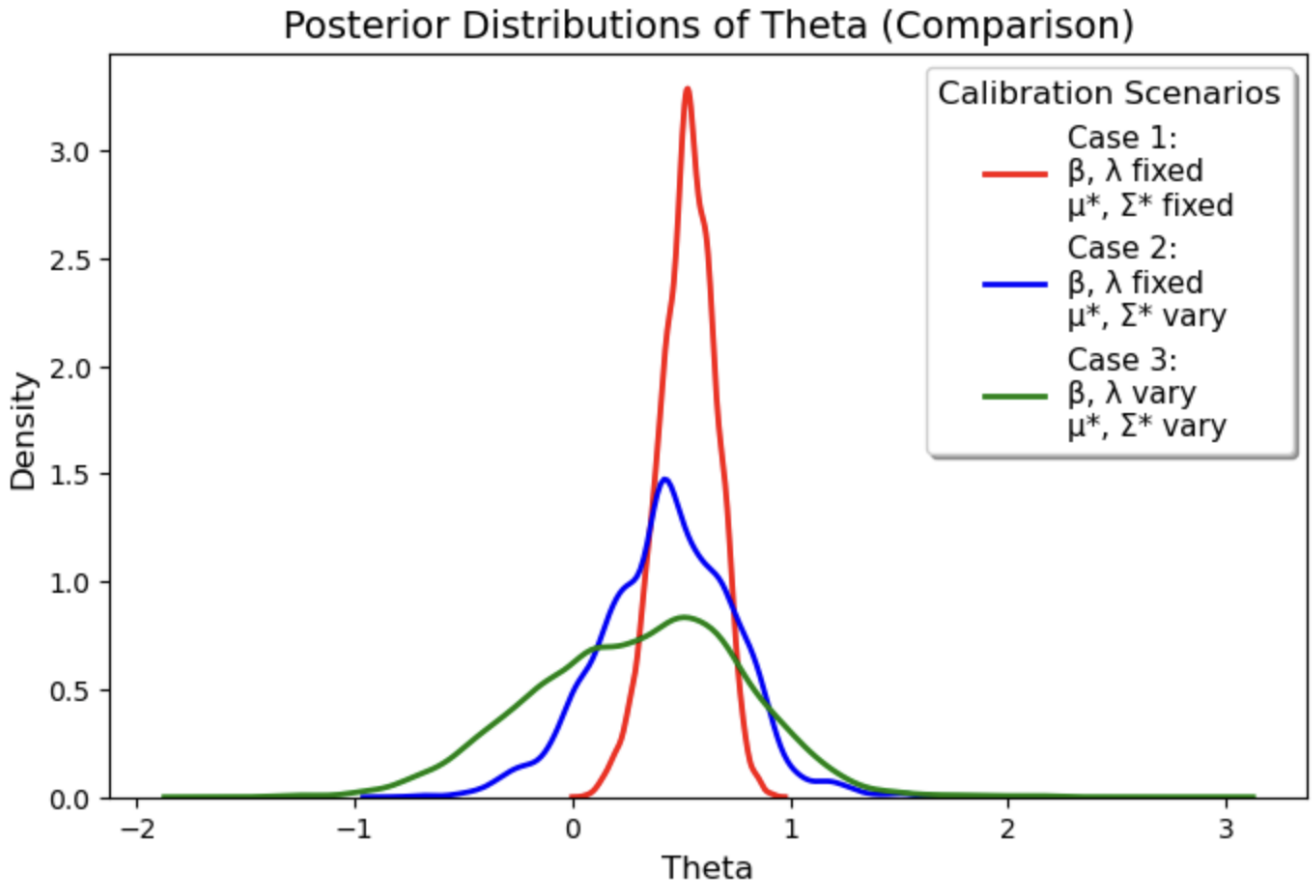


Calibration Cases

- We conduct uncertainty quantification using three different versions of the outputs from the emulator:
  - **Case 1:** We use fixed values of hyperparameters and fixed values of emulator mean and variance for calibration.
  - **Case 2:** We use fixed values of hyperparameters and sample values of from MVN density with mean and variance for calibration.
  - **Case 3:** We sample from posterior densities of hyperparameters and sample values of from MVN density with mean and variance for calibration.
- As we progress from case 1 through 3, we add more variability to calibration model.

Calibration Results

- Uncertainty is minimum when the lowest amount of variability is introduced and increases as variability increases.



Thank You!