Uncertainty Quantification and Calibration In Nuclear Safety Codes Using Gaussian Process Active Learning

by

Eric Nels Fugleberg

B.S., Systems Engineering
United States Naval Academy, 2014

Submitted to the Department of Nuclear Science and Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Nuclear Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

© 2016 Massachusetts Institute of Technology. All rights reserved

Signature of Author ....
Department of Nuclear Science and Engineering
May 6, 2016

Certified by..
Dr. Michael Golay
Professor of Nuclear Science and Engineering, MIT
Thesis Supervisor

Certified by..
Dr. Robert Youngblood
Senior Risk Consultant, Idaho National Laboratory
Thesis Reader

Accepted by ....
Dr. Ju Li
Battelle Energy Alliance Professor of Nuclear Science and Engineering
Chair, Department Committee on Graduate Studies

Uncertainty Quantification and Calibration In Nuclear Safety Codes
Using Gaussian Process Active Learning

by

Eric Nels Fugleberg

Submitted to the Department of Nuclear Science and Engineering
on May 6, 2016 in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Nuclear Science and Engineering

Abstract

Inverse problems and inverse uncertainty quantification (UQ) are challenging issues when dealing with complex and highly non-linear functions. Methods have been developed to decrease the computational burden by using the Gaussian Process (GP) emulator model framework to approximate the input-output relation of a deterministic computer code. The GP emulator can then be used in place of the computer code to perform Bayesian calibration techniques to determine uncertain parameter distribution. The performance of a GP emulator is largely dependent on the quality of the points in its training set; the best emulator exactly replicates the output of the computer code. The uncertain parameter posterior sample space is not known *a priori*, resulting in GP training sets covering as much of the prior sample space as possible in hopes of covering the posterior space well enough. This work improves the performance of the simple GP emulator using an active learning methodology to select additional training points which cover the posterior sample space of the unknown parameters. Furthermore, the effect of the covariance function on the performance of the GP is investigated with recommendations made for future GP emulator applications.

Thesis Supervisor: Dr. Michael Golay
Title: Professor of Nuclear Science and Engineering, MIT

Thesis Reader: Dr. Robert Youngblood
Title: Senior Risk Consultant, Idaho National Laboratory

Acknowledgements

Many thanks to Dr. Joe Yurko whose previous work made this thesis entirely possible. He took the time to teach me about Bayesian Inference, MCMC sampling, and Gaussian Processes. Additionally, he was always available to answer my questions and push me further along in my research.

Thank you Professor Buongiorno for approaching me with this unique and challenging research topic. I'm glad you encouraged me to go out of my comfort zone and had the patience to deal with me along the way.

I wish to thank Professor Golay for taking me on as a new advisee this previous year. He offered a new perspective to this work, and I appreciated his guidance and support along the way.

Thank you to Dr. Youngblood at Idaho National Laboratory for listening to my ideas of how to improve the simple Gaussian Process regression model. I thoroughly enjoyed our intellectual discussions, and the little history lessons that worked their way into our conversations. Your guidance and mentorship was very much appreciated.

Financial support for this research was provided by the Department of Nuclear Science and Engineering and Idaho National Laboratory. Thank you for ensuring I received the necessary funding to complete this research and my Masters degree.

I would like to thank my Mom, Dad, and brother for their continued support

Lastly, thank you to my wife Rebecca for her patience and support these past two years. I love you.

Contents

List of Figures

List of Tables

List of Abbreviations and Mathematical Symbols

Abbreviations

AM	Adaptive Metropolis	
ARD	Automatic Relevance Determination	
BIC	Bayesian Information Criterion	
DF	Driver Fuel	
FFGP	Function Factorization with Gaussian Process Priors	
GP	Gaussian Process	
GPR	Gaussian Process Regression	
LB	Lower Blanket	
LHS	Latin Hypercube Sampling	
Lin.	Linear Covariance Function	
MCMC	Markov Chain Monte Carlo	
MH	Metropolis-Hastings	
MVN	Multivariable Normal	
Pa	Pascal, a unit of pressure	[1 atm $= 101.3 \times 10^3$ Pa]
Per.	Periodic Covariance Function	
RELAP	Reactor Excursion and Leak Analysis Program	
RMSE	Root Mean Square Error	
RQ	Rational Quadratic Covariance Function	
RWM	Random-Walk Metropolis	
SE	Squared Exponential Covariance Function	
UB	Upper Blanket	
UQ	Uncertainty Quantification	

Mathematical Symbols

α_M	MCMC Acceptance Probability	
D	Number of input parameters	
$\mathbb{E}\left[x\right]$	Expected value, or mean, of x	
$f(\mathbf{x})$	Output of computer code or simulator	
$f_{\mathcal{GP}}\left(\mathbf{x}\right)$	Gaussian Process latent function	
$\mathcal{GP}$	Gaussian Process	
$\mathbb{I}$	Identity matrix	
$k(\mathbf{x}_p, \mathbf{x}_q)$	Covariance function between $\mathbf{x}_p$ and $\mathbf{x}_q$	
K_ν	Modified Bessel Function of Second Order ν	
$\mathbf{K}\left(X, X'\right)$	Covariance matrix of matrices X and X'	
l	Characteristic Length-Scale	
$m(\mathbf{x})$	The mean function	
M	Matrix of Characteristic Length-Scales	$[M = \text{diag}\left(l\right)^{-2}]$
N	Number of training points	
N_*	Number of test points	
ϕ	Emulator Hyperparameters	
Φ_S	Friction Factor Correlation Shape Factor	
r_k	Autocorrelation coefficient	
$\hat{\Sigma}$	AM-MCMC expirical covariance matrix	
Σ_t	Experimental Error Covariance Matrix	
s^2	MCMC Tuning Scalar Constant	
$s^2\Sigma$	MCMC Covariance Matrix	
θ	Uncertain input parameters	
$\mathbf{x}_{cv}$	Control variable input parameters	
$\mathbf{x}^o$	Initial sample in MCMC sampling	
$\mathbf{x}^*$	Proposed sample in MCMC sampling	
$\mathbf{x}^t$	Current guess in MCMC sampling	
X	Training set input matrix of size $N \times D$	
X_*	Test matrix of size $N_* \times D$	
$\mathbf{y}$	Training set output matrix of size $N \times 1$	
$\mathbf{y_e}$	Experimental data points	
$\mathbf{y}_*$	Posterior predictive mean	

Chapter 1

Introduction

Uncertainty quantification (UQ) for nuclear reactor safety codes is more of an art than a science due to the highly non-linear system response. Methods have been developed to implement Bayesian inference techniques and calibrate parameter distributions to experimental data. These methods are systematic and statistically rigorous, but come at large computational costs. Bayesian methods are more expensive than other UQ methods because they require the computer code to be run thousands, if not tens of thousands of times in series. Therefore, a fast approximation to the computer code is required to implement Bayesian calibration methods better. Gaussian Process (GP) emulators have been developed to be used in place of the computer code to use Bayesian calibration methods for parameter calibration quickly. The uncertain parameter posterior sample space is not known *a priori,* resulting in GP training sets covering as much of the parameter prior sample space as possible in hopes of covering the parameter posterior space well enough. This naive approach works well with a large number of training points, but if the computer code takes hours to complete one iteration then generating thousands of data points is not realistic. The work reported here seeks to improve the performance of the simple GP emulator by developing an active learning methodology to select training points covering the parameter posterior sample space. Furthermore, the effect of the covariance function on the performance of the GP will be investigated with recommendations made for future GP emulator applications.

1.1 Organization of this Work

Chapter 2 gives an introduction to Bayesian inference and inverse problems. Approximate inference with Markov Chain Monte Carlo (MCMC) is also introduced. This chapter illustrates why emulators are required when performing Bayesian calibration of long running computer codes.

Chapter 3 describes the Gaussian Process emulator used in this work. The standard Gaussian Process Regression (GPR) emulator is introduced and demonstrated to show the flexibility and power of the GPR framework. A friction factor demonstration problem familiarizes the reader with simple GPR emulators.

Chapter 4 discusses the covariance function in greater detail. Various simple and complex covariance functions are introduced, as well as methods to combine the various covariance functions. The rest of this work requires understanding the different covariance functions.

Chapter 5 introduces the active learning methodology created to improve the training set for the GPR emulator. The methodology requires constructing an initial GPR emulator from a space-filling training set and then adding training points which cover the posterior sample space. Ideally, the emulator will exactly reproduce the same output as the computer code. The friction factor demonstration is revisited, with the emulator nearly replicating the directly sampled computer code output. The effects of different covariance functions are investigated with a “best” covariance function chosen for the friction factor problem.

Chapter 6 applies the active learning methodology to two reactor safety code examples with several different covariance functions being tested. The error of the experimental data is assumed to be constant for each experimental data point in the first case; this represents experiments which have a known error that is unresolvable. The second case investigates how the emulator performance changes if the experimental error scales as the mean; this represents the experimental error of the measurement device. The “best” covariance functions are presented for each case with recommendations for future GPR applications.

Chapter 7 summarizes the work reported here, reiterates the lessons learned for using Gaussian Process emulators, and provides areas for future work to improve Gaussian Process emulators further using active learning process.

Chapter 2

Bayesian Inference for Inverse Problems

2.1 Inverse Problems

For inverse problems, we wish to estimate the values of unknown objects, such as parameters or functions, from indirect noisy observations [1]. Consider the function $y = f(x)$ where the output y is considered to be a function of the input variable x. A "forward problem" dictates the presumption that a given input produces a specific output. An inverse problem, on the other hand, wishes to take an observation and determine which x-value produced that particular y output value. To do this, we must compute the inverse function $x = f^{-1}(y)$. Another way to think of this process is as a forward problem using deductive logic: given a cause, one must figure out the consequence, while an inverse problem uses inductive logic: given a consequence, what were the underlying causes [2]. Uncertainty quantification (UQ) works much in a similar manner. Forward UQ takes distributions (uncertainties) on the input parameters and propagates those onto the output. Inverse UQ works in the opposite manner: given an output distribution, what are the distributions on the input parameters. Bayesian inference techniques can be used to perform this inverse UQ operation.

2.2 Bayesian Inference

The fundamental concept of Bayesian analysis is that unknowns are treated as random variables. The power of this approach is that the established mathematical methods of probability theory are applied to develop informative representations of the state of knowledge regarding the unknowns [1]. Bayesian analysis relies on Bayes' theorem, which is a fundamental relationship of conditional probabilities. Each unknown is given a prior probability distribution reflecting the current state of knowledge regarding their possible values. These prior distributions are then "updated" conditional on new information, through the likelihood function. The result is a posterior distribution that represents the new state-of-knowledge, a combination of the prior evidence and the observed data. If $\mathbf{x}$ is a vector of unknown parameters we wish to learn more about and $\mathbf{y}$ is the vector of observed data, Bayes' theorem is written as:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{\int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x}}. \tag{2.1}$$

The numerator of Eq. 2.1 consists of the likelihood function, $p(\mathbf{y}|\mathbf{x})$, which gives the probability of the observations $\mathbf{y}$ given the unknown parameters $\mathbf{x}$, and the prior distribution on the unknown parameters $p(\mathbf{x})$. The denominator is known as the marginal likelihood and integrates out (marginalizes) the unknown parameters. The denominator is therefore equivalent to:

$$p(\mathbf{y}) = \int p(\mathbf{x}|\mathbf{y})p(\mathbf{x})d\mathbf{x}. \tag{2.2}$$

The marginal likelihood does not depend on the unknown parameters and is therefore merely a normalizing constant. Because of this, Bayes' theorem is often written simply as the posterior being proportional to the product of the likelihood and the prior:

$$p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x}). \tag{2.3}$$

The difficulty in implementing Bayes' theorem is computing the marginal likelihood in Eq. 2.2. Evaluating the potentially high-dimensional integral could be intractable analytically and/or very expensive computationally. Analytical intractability results from the fact that the likelihood function may involve non-linear mapping between the unknown parameters and the observed output data. Such is the case for Bayesian calibration of computer models: the output of the computer model is a function of the unknown input parameters and other inputs that are not necessarily "unknown," such as time. Therefore, the likelihood function is not a function of the uncertain parameters themselves, but of the output of the computer code: $p(\mathbf{y}|f(\mathbf{x}))$; the computer model will also be referred to as the compute code and is represented as $f(\mathbf{x})$. With closed form solutions not possible, sampling methods must be used instead. The primary sampling method used is a Markov Chain Monte Carlo (MCMC) sampling strategy that attempts to draw samples from the posterior distribution directly. The next section will cover the basis of an MCMC scheme.

2.3 Approximate Inference with MCMC

The basic premise behind Markov Chain Monte Carlo (MCMC) is to construct a Markov chain whose stationary distribution is the target density that we wish to sample [3]. In the context of Bayesian model calibration, that target density is the posterior distribution of the unknown parameters given the experimental data: $p(\mathbf{x}|\mathbf{y})$. MCMC is similar to the Monte Carlo (MC) sampling technique, except that the samples are correlated in MCMC and no longer independent and identically distributed. The samples are correlated due to the Markov chain, and that is the price that we pay for not knowing the actual posterior distribution. The main benefit of MCMC is that the marginal likelihood, Eq. 2.2, does not need to be computed. This is accomplished by a scientific "guess and check" method which will be discussed later.

All MCMC strategies involve some sort of random walk around the sample space of the unknown parameters of interest. The main components to MCMC sampling are:

1. An initial guess, $\mathbf{x}^o$

2. A proposal (or transition) distribution, $q(\mathbf{x}^*|\mathbf{x})$, where $\mathbf{x}^*$ is the proposed sample for the unknown parameters

3. An accept/reject rule for whether to keep the proposed sample.

The proposal distribution, $q(\mathbf{x}^*|\mathbf{x})$, determines the efficiency of the MCMC sampling; the more naive and random the proposal, the less efficient the MCMC scheme. Efficiency relates to how correlated the MCMC samples are, meaning a more efficient proposal distribution allows the MCMC chain to "forget" the initial state faster [4].

Making a good initial guess in MCMC schemes is difficult. The prior distribution mean is often a good initial guess ($\mathbf{x}^o$), but if the posterior distribution is quite different from the prior distribution, it may take the sampling scheme a long time to shift the prior distribution to the posterior distribution. Even so, the chain will gradually "forget" its initial state and will eventually converge to a unique stationary distribution, which does not depend on the sample number [5]. This means a poor starting point only affects the number of samples required for the chain to "forget" its initial point. However, if a limited number of samples are used, the initial sequence may impact the results. That is why it is common to discard a portion of the sample values at the beginning of the MCMC scheme as "burn-in."

The most common acceptance/rejection criterion is the Metropolis-Hastings (MH) criterion. The MH criterion compares the posterior probability for the initial guess and the proposed sample. If the posterior probability of the sample point increases, then one accepts the proposed sample. But, if the posterior probability of the sample point decreases, then one may randomly choose whether to accept the point with acceptance probability of:

$$\alpha_M = \min\left(1, \frac{p(\mathbf{x}^*|\mathbf{y})}{p(\mathbf{x}^t|\mathbf{y})}\right), \tag{2.4}$$

where t is the current sample. Thus, if the proposed value of $\mathbf{x}^*$ is less probable than the current guess, $\mathbf{x}^t$, the chain may still move there anyway. Instead of greedily moving to only more probable states, we occasionally allow "downhill" moves to less probable states [3]. This kind of behavior ensures that the fraction of time spent at each point $\mathbf{x}$, is proportional to the posterior distribution $p\left(\mathbf{x}|\mathbf{y}\right)$ [4]. One important characteristic of Eq. 2.4 is that because it is a ratio of the posterior probabilities, the marginal likelihood cancels out and does not need to be computed. Hence, samples are drawn from the posterior distribution without computing Eq. 2.2.

2.3.1 Random-Walk Metropolis Sampler

The simplest MCMC scheme is the Random-Walk Metropolis (RWM) sampler. For this scheme, the proposal distribution is centered at the current guess for the unknown parameters, $\mathbf{x}^t$, with specified covariance matrix of $s^2\Sigma$. The scalar s^2 is some constant set to facilitate "rapid mixing" [3]. It is usually assumed that the uncertain parameters are uncorrelated when implementing a RWM sampling scheme, thus the covariance matrix Σ is diagonal. If the unnormalized posterior density is written as $h(\mathbf{x}) = p(\mathbf{y}|f(\mathbf{x}))p(\mathbf{x})$, then the RWM update occurs in the following manner:

1. At current guess, $\mathbf{x}^t$, with current unnormalized posterior $h(\mathbf{x}^t)$, propose a move to $\mathbf{x}^*$ with proposal density: $q\left(\mathbf{x}^*|\mathbf{x}^t\right) = \mathcal{N}\left(\mathbf{x}^t, s^2\Sigma\right)$.
2. Compute the proposed unnormalized posterior $h(\mathbf{x}^*)$.
3. Compute the Metropolis acceptance probability using Eq. 2.4.
4. Set the next sample value to be: $\mathbf{x}^{t+1} \begin{cases} \mathbf{x}^* & \text{, with probability } \alpha_M \\ \mathbf{x}^t & \text{, with probability 1-}\alpha_M \end{cases}$

This process is repeated for the desired number of MCMC samples (including burn-in). The scalar constant s^2 may be tuned in order to equal some optimal sampling rate. If the constant is too large, the proposal sample $\mathbf{x}^*$ will be far away from the initial guess $\mathbf{x}^t$ and the acceptance probability (or chance of the chain moving) will be very low. In this case, the chain will most likely "sit" at the same value for a long time. Conversely, if the proposal is very close to the initial guess then the acceptance rate will be very high, but the information gained from each sample will be very low. The optimal acceptance rate (or mixing rate) is

0.234, meaning that roughly 23% of all proposed samples are accepted [6]. The scalar value s^2 can be tuned, but only during the burn-in phase and must remain fixed for the rest of the sampling. This is because the tuning effects the memoryless-ness nature of the Markov chain [4].

2.3.2 Adaptive Metropolis Sampler

Tuning the scalar constant, s^2, is a simple adaptive scheme that essentially adapts the variance of each element in $\mathbf{x}$ equally; remember that $\mathbf{x} = [x_1, x_2, \ldots, x_{\mathcal{D}}]^{\mathrm{T}}$. It is possible to tune each element of the proposal variance, but that becomes computationally expensive as the dimension $\mathcal{D}$ grows. Haario et al. [7] set up an Adaptive Metropolis (AM) scheme that adapts the entire proposal covariance matrix by using the past samples of the Markov chain. This allows the proposal distribution to better describe the covariance between the different elements in $\mathbf{x}$ and can improve the efficiency of the MCMC. In order to denote the proposal covariance being computed on the fly for the AM sampling scheme, it is labeled $\hat{\Sigma}$. The proposal matrix is now: $q(\mathbf{x}^*|\mathbf{x}^t) = \mathcal{N}(\mathbf{x}^t, \hat{\Sigma})$; the update steps are the same as with the RMW, but using $\hat{\Sigma}$ instead of $s^2\Sigma$.

In implementing the AM scheme, the initial MCMC runs do not update the proposal covariance matrix. After the initial runs, the proposal covariance matrix is updated using a specified number of past samples after a specific number of iterations. For example, after every 1000 samplings the previous 1000 samples are used to compute the empirical covariance matrix, $\hat{\Sigma}$. This is more efficient than computing the empirical covariance matrix at every iteration using all past samples.

2.4 Demonstration Problem

In order to demonstrate Bayesian model calibration using directly sampled MCMC, a method of manufactured solutions is used. Here the experimental data are generated using a computer model with specified true-parameter values [4]. The goal is to see whether the MCMC sampling yields posterior distributions around these true-parameter values. The turbulent friction factor problem in [4] will be used as the "computer model" parameterized as:

$$f = exp\left(b\right) Re^{-exp(c)}, \tag{2.5}$$

where Re is the Reynolds number and b and c are the two uncertain parameters. This model is usually written as $f = B \cdot Re^{-C}$, but the model was transformed such that $B = \exp(b)$ and $C = \exp\left(c\right)$ in order to ensure always positive values for the uncertain parameters. Furthermore, this transformation made it easier to use Gaussian distributions for the priors on the uncertain parameters and give the original parameters, B and C, log-normal prior distributions. The prior mean values were chosen such that $\exp(\mathbb{E}[b]) = 0.184$ and $\exp(\mathbb{E}[c])$=0.2, where $\mathbb{E}\left[x\right]$ denotes the expected (or mean) value of a variable x; expected values correspond to the McAdams' friction factor correlation values [8]. The prior variance for each transformed parameter was assumed so that 95% of the probability covered ±50% around the prior means.

Twenty-five artificial "experimental data" points were generated at the specified true-parameters values of $\exp\left(b\right) = 0.25$ and $\exp\left(c\right) = 0.10$ at evenly spaced intervals between Reynolds numbers of 5000 and 45000. The experimental data points are denoted as $\mathbf{y} = \left[y_1, y_1, \ldots, y_{N_o}\right]^{\mathrm{T}}$ where $N_O = 25$ is the number of experimental data points. These experimental data points are used as if they had came from a physical experiment, but they are artificially created for ease of use in this demonstration.

The simple friction factor model shows that usually multiple types of input variables exist: the b and c parameters are the unknown or uncertain parameters, while the Reynolds number, Re, is a control variable [9]. The uncertain parameters are denoted collectively as $\theta = \left[\mathbf{b}, \mathbf{c}\right]^{\mathrm{T}}$

with $\mathbf{b}$ and $\mathbf{c}$ denoting the vectors of uncertain parameters, and the control variables will be denoted as $\mathbf{x}_{cv} = [Re_1, Re_2, \ldots, Re_{N_O}]^{\mathrm{T}}$. Denoting the prior means as μ_b and μ_c with variances σ_b^2 and σ_c^2, the prior distribution is a multivariate normal (MVN) distribution:

$$p(\theta) = \mathcal{N}\left(\begin{bmatrix} \mu_b \\ \mu_c \end{bmatrix}, \begin{bmatrix} \sigma_b^2 & 0 \\ 0 & \sigma_c^2 \end{bmatrix}\right). \tag{2.6}$$

A Gaussian likelihood function is used to relate the data to the friction factor model:

$$p(\mathbf{y}|f(\mathbf{x}_{cv}, \theta)) = \mathcal{N}(f(\mathbf{x}_{cv}, \theta), \Sigma_t), \tag{2.7}$$

where Σ_t is the experimental error covariance matrix. The experimental error covariance matrix, Σ_t, for this example is assumed to be diagonal with experimental error (variance) to have 95% of the probability covered by $\pm 10\%$ around the mean data values [4]. In a real applications, the experimental error is usually defined by the uncertainty of the measurement device.

Two MCMC schemes were used to sample the posterior distribution directly on the uncertain parameters, $p(\theta|\mathbf{y})$; each scheme used 5×10^4 burn-in samples and 5×10^4 posterior distribution samples. The first is the Random-Walk Metropolis (RWM) sampler with the scalar constant s^2 fixed to $s^2 = (0.5)^2$. The second scheme is the Adaptive Metropolis (AM) sampler, which adapts during the burn-in period after an initial delay of 2×10^3 samples. While adapting, the AM scheme recomputes the empirical covariance matrix every 1000 samplings using the past 1000 samples. Figure 2.1 illustrates the poor mixing of the fixed scalar constant as well as the well-mixed behavior obtained using the Adaptive Metropolis.

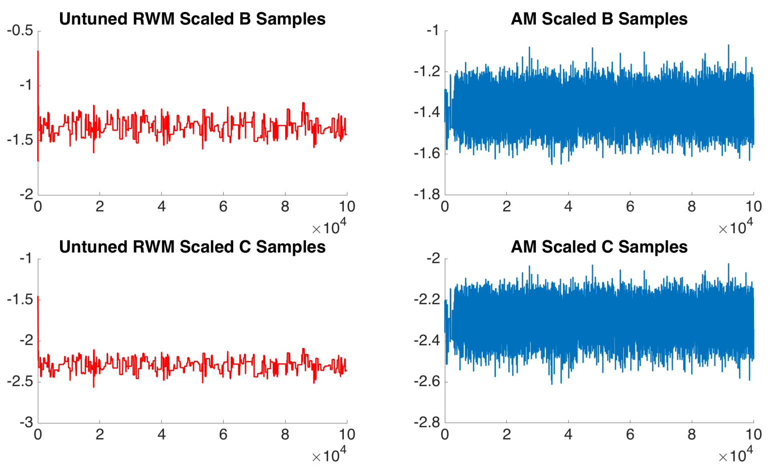

Figure 2.1: MCMC sample histories

This well-mixed behavior can be observed quite clearly in the sample histories, but also through an autocorrelation length over a specified lag as shown in Figure 2.2. The autocorrelation length is the correlation between observations as a function of the time lag between them defined as [10]:

$$r_k = \frac{\sum\limits_{i=1}^{N-k} \left(X_i - \bar{X}\right)\left(X_{i+k} - \bar{X}\right)}{\sum\limits_{i=1}^{N} \left(X_i - \bar{X}\right)^2}, \tag{2.8}$$

for given measurements $X_1, X_2, ..., X_N$ with a lag k over N observations, where $\bar{X} = \frac{1}{N}\sum\limits_{i=1}^{N} x_i$. Autocorrelation is a correlation coefficient not between two different variables, but between two values of the same variable at times X_i and X_{i+k}; autocorrelation is used to detect non-randomness in the behavior of a variable [11]. In Figure 2.2, the autocorrelation is computed with a lag of 100. The untuned RWM scheme has samples that are correlated by roughly 0.8 at best. The AM scheme, on the other hand, has an independent sample drawn nearly every 60 or 70 samples, and corresponds to an autocorrelation of nearly 0. The increased mixing rate from the AM-MCMC scheme also improves the posterior predictions, as shown in Figure 2.3.

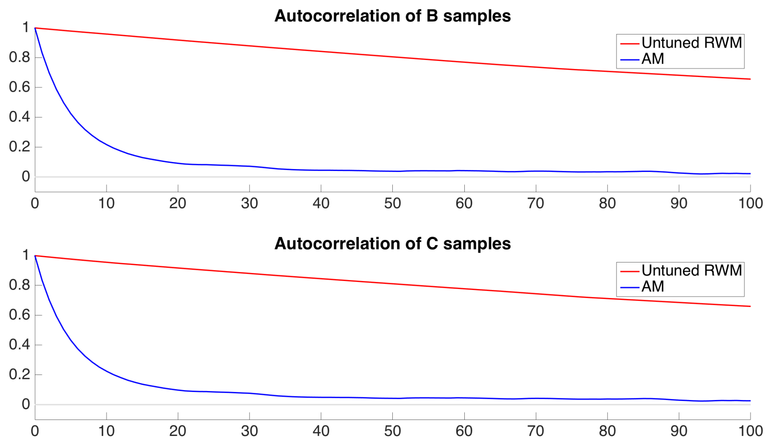

Figure 2.2: Autocorrelation for MCMC sampling.

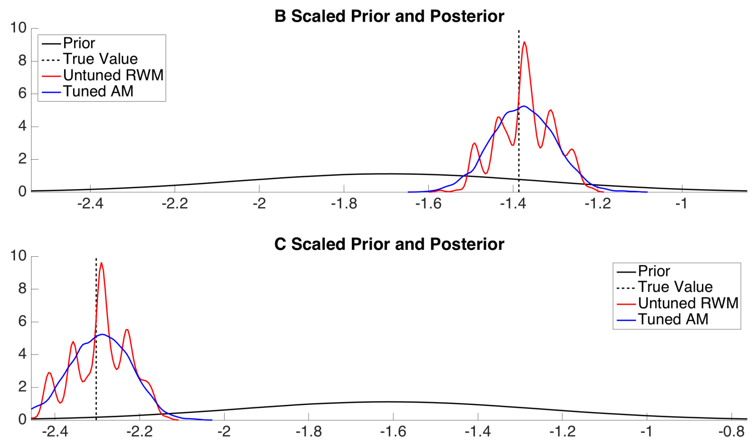

Figure 2.3: Friction factor directly sampled posterior distributions.

The posterior distributions were generated using the MATLAB function, *ksdensity*, over the 5×10^4 posterior distribution samples [12]. The AM-MCMC scheme has a much smoother posterior distribution, attributed to the better mixing rate. Since Bayesian inference and calibration requires tens of thousands of MCMC samples, the limiting factor in the time it takes to perform Bayesian calibration is the speed required to evaluate the computer code, providing numerical values for $f\left(\mathbf{x_{cv}}, \theta\right)$. Using an emulator in place of the computer code can greatly speed up the MCMC sampling process.

Chapter 3

Gaussian Process Emulators

3.1 Non-parametric Emulators

As illustrated in Chapter 2, performing Bayesian inference requires using tens of thousands of samples from the MCMC scheme. Thus, the limiting factor is the speed in which we can draw one sample and compute the likelihood function, $p\left(\mathbf{y}|f(\mathbf{x})\right)$. Even a "fast" computer simulation taking three seconds per iteration to compute $f\left(\mathbf{x}_{cv}, \theta\right)$ would require over 83 hours of run time for producing 10^5 MCMC samples. Most codes require much more time than three seconds per iteration, meaning that they are time consuming and too computationally expensive. A relatively quick method for determining the input/output relation of the model is therefore desirable. In the literature, these non-parametric models are often referred to as emulators as they try to emulate (or reproduce) the behavior of the computer code. A non-parametric model uses the training data to dictate the trends, rather than a modeling choice beforehand. A popular choice for emulators is the Gaussian Process which has been in use since the 1960s, and has recently gained much attention by the machine learning community.

3.2 Gaussian Process

A reactor safety analysis code, such as RELAP[1], is technically deterministic as it will produce the same outputs given the same inputs. However, the output is somewhat unknown until the computer code is run. Therefore, the output, subject to special conditions, can be treated as a random variable. Formally, a Gaussian Process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution [13]. What makes a GP special is that the covariance matrix of this multivariate normal distribution is computed using the training dataset. This feature allows the GP to interpolate the training data, thereby emulating the behavior of the computer code [4].

A GP is completely specified by its mean and covariance functions. The mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}_p, \mathbf{x}_q)$ are defined as:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}\left[f(\mathbf{x})\right], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[\left(f(\mathbf{x}) - m(\mathbf{x})\right)\left(f(\mathbf{x}') - m(\mathbf{x}')\right)\right], \end{aligned} \tag{3.1}$$

where $\mathbf{x}$ is the vector of inputs and the output, $f_{GP}(\mathbf{x})$, is a random variable. The GP is written as:

$$f_{GP}(\mathbf{x}) = \mathcal{GP}\left(m(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right). \tag{3.2}$$

An important part of Eq. 3.2 is that the covariance function is only a function of the inputs, $\mathbf{x}$. The mean function is often taken to be equal to zero for notational simplicity and to not assume any trend of the data (such as linear or quadratic) [13]. A Gaussian process is defined as a collection of random variables. Thus, the definition automatically implies a consistency requirement, which is also sometimes known as the marginalization property [13]. This property simply means that, for example, if the GP specifies $(y_1, y_2) \sim \mathcal{N}(\mu, \Sigma)$, then it must also specify $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$, where Σ_{11} is the relevant submatrix of Σ. This means that examination of a larger set of variables does not change the distribution of the smaller

[1]The Reactor Excursion and Leak Analysis Program (RELAP) is a light water reactor transient analysis code.

set [13].

The covariance function depends on a set of hyperparameters which specify the covariance function and must be learned from the training data; this keeps the GP non-parametric as it only depends on the training data. The most common covariance function used in the literature is the squared exponential (SE) covariance function:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}\left(\mathbf{x}_p - \mathbf{x}_q\right)^{\mathrm{T}} M \left(\mathbf{x}_p - \mathbf{x}_q\right)\right), \tag{3.3}$$

where the subscripts p and q are two different indices of the input $\mathbf{x}$. The hyperparameters are the signal variance σ_f^2 and the matrix M, which is a symmetric matrix given as:

$$M = \operatorname{diag}\left(l\right)^{-2}, \tag{3.4}$$

where l is a vector of positive values, and each element $l_1, \ldots, l_D$ plays the role of the characteristic length-scale for each input parameter [4]. The length-scale, loosely speaking, represents how far a sample needs to move (along a particular axis) in the input space for the function values to become uncorrelated [14]. This formulation of the hyperparameters implements what is known as automatic relevance determination (ARD), meaning that the inverse of the length-scale determines the relevance of that input parameter.

3.3 Gaussian Process Regression (GPR)

The GP model is able to interpolate the training data exactly if there is no noise. However, noise is allowed to prevent ill-conditioning of the covariance matrix. Because of this, the GP model is better labeled a Gaussian Process Regression (GPR) model as it is trying to regress the training dataset within some allowable noise level [4]. The GP prior is therefore placed on a latent (or hidden) function, $f_{\mathcal{GP}}\left(\mathbf{x}\right)$, that we wish to infer from the noisy data, y [4].

While the computer code training output is not noisy, the output y can be related to the GP function, $f_{GP}(\mathbf{x})$, as:

$$y = f_{GP}(\mathbf{x}) + \epsilon, \tag{3.5}$$

where ϵ is the error structure. Assuming additive independent identically distributed Gaussian noise ϵ with variance σ_n^2, the prior on the noisy observations becomes:

$$\operatorname{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 \mathbb{I}. \tag{3.6}$$

This follows from the independence assumption about the noise [13].

3.3.1 GPR Training Set

If there are N training points for D input parameters, all of the parameters are stacked into an $N \times D$ matrix. Each row of X contains the D input parameters. The training output values, $\mathbf{y}$, are in a similar stacked matrix, that is of dimension $N \times 1$:

$$X = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_N^{\mathrm{T}} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}.$$

Since $f_{\mathcal{GP}}(\mathbf{x})$ is a GP and the likelihood function is also Gaussian, the training set prior is also Gaussian as:

$$\mathbf{y} \sim \mathcal{N}\left(0, \mathbf{K}(X, X) + \sigma_n^2 \mathbb{I}\right), \tag{3.7}$$

where $\mathbf{K}(X, X)$ is the training set covariance matrix and $\mathbb{I}$ is the identity matrix.

The training set covariance matrix requires applying the chosen covariance function to each pair of input parameter values:

$$\mathbf{K}(X, X) = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & k(\mathbf{x}_N, \mathbf{x}_2) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}. \tag{3.8}$$

If the SE covariance function in Eq. 3.3 is used, then each diagonal element of $\mathbf{K}(X, X)$ is the signal variance σ_f^2 .

For a GPR model, we want each training point to offer as much new information as possible. That means if several training points have the same x_1 with differing x_2, there is no learning of the relation between x_1 and the output $y(\mathbf{x})$ from those points. This is why an initial training set wishes to cover as much of the prior sample space as possible over all input parameters. Latin hypercube sampling (LHS) as proposed by [15] is a sampling scheme where each input variable $\mathbf{x}_i$ has all portions of its distribution represented in N strata with equal probability of $1/N$ and one sample from each stratum. These form the $\mathbf{x}_{i,n}$ component for $i = 1, \ldots, I$ inputs and the $n = 1, \ldots, N$ stratum. The components of the various $\mathbf{x}_{i,n}$ are then matched so as to not have two points in the same i or n. Further refinement of the LHS is the orthogonal sampling in which the full sample space is divided into further subspaces of equal probability. This could be thought of as another implementation of the LHS within itself. Figure 3.1 shows a visual representation of three different input sampling schemes: random, Latin hypercube, and orthogonal.

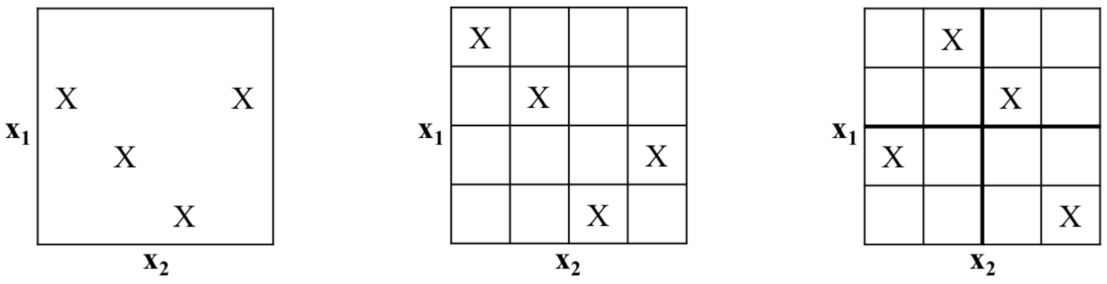

Figure 3.1: Demonstration of different initial sampling schemes (L to R): Random sampling, Latin hypercube sampling, Orthogonal sampling.

3.3.2 GPR Predictions

With a training set and known covariance matrix (assuming, for now, the hyperparameters are known), we wish to make predictions at new input parameter values, or test points. For N_* test points, the test input matrix, X_*, is similar to the training set matrix and is $N_* \times D$. The GP prior is of a similar structure as Eq. 3.7:

$$\mathbf{f}_* \sim \mathcal{N}\left(0, \mathbf{K}\left(X_*, X_*\right) + \sigma_n^2 \mathbb{I}\right). \tag{3.9}$$

The test covariance matrix $\mathbf{K}\left(X_*, X_*\right)$ has the same structure as the training set covariance matrix, except that now the elements in the matrix are computed using the test input parameters. However, Eq. 3.9 provides information of little value as it is not informed about the structure of the training set. Therefore, the test distribution must be conditioned on the training set distribution. First, the joint prior is written as:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{K}\left(X, X\right) + \sigma_n^2 \mathbb{I} & \mathbf{K}\left(X, X_*\right) \\ \mathbf{K}\left(X_*, X\right) & \mathbf{K}\left(X_*, X_*\right) \end{bmatrix}\right), \tag{3.10}$$

where $\mathbf{K}\left(X, X_*\right)$ is the cross-covariance matrix between the training set and the test input of size $N \times N_*$, with $N_* \times N$ being its transposed matrix.

Standard multivariate normal theory allows us to write the conditional distribution $p\left(\mathbf{f}_*|\mathbf{y}\right)$ and give the key predictive equations for the GPR [13]:

$$\mathbf{f}_*|\mathbf{y} \sim \mathcal{N}\left(\bar{\mathbf{f}}_*, \operatorname{cov}\left(\mathbf{f}_*\right)\right) \tag{3.11}$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}\left[\mathbf{f}_*|\mathbf{y}\right] = \mathbf{K}\left(X_*, X\right)\left[\mathbf{K}\left(X, X\right) + \sigma_n^2\mathbb{I}\right]^{-1}\mathbf{y} \tag{3.12}$$

$$\operatorname{cov}\left(\mathbf{f}_*\right) = \mathbf{K}\left(X_*, X_*\right) - \mathbf{K}\left(X_*, X\right)\left[\mathbf{K}\left(X, X\right) + \sigma_n^2\mathbb{I}\right]^{-1}\mathbf{K}\left(X, X_*\right). \tag{3.13}$$

To make a posterior prediction, $\mathbf{y}_*$, in the training data space, the posterior distribution is obtained as:

$$\mathbf{y}_* \sim \mathcal{N}\left(\bar{\mathbf{f}}_*, \operatorname{cov}\left(\mathbf{f}_*\right) + \sigma_n^2\mathbb{I}\right). \tag{3.14}$$

Eq. 3.12 and Eq. 3.13 give insight to the power of GPR models: First, for predictions at the training points ($X_* = X$), the predictive uncertainty shrinks to the allowable error tolerance, as shown in Eq. 3.12. Second, the posterior predictive covariance shrinks the prior covariance by the subtraction of the first and second terms in Eq. 3.13. The computational expense is completely determined by the cost to invert the training set covariance matrix. Furthermore, if the training set covariance matrix is ill-conditioned, the inverse cannot be computed. Thus, the noise is allowed in order to ensure that $\left[\mathbf{K}\left(X, X\right) + \sigma_n^2\mathbb{I}\right]^{-1}$ is always invertible [4].

3.4 Building the Emulator

When introducing GPR predictions before, the training set hyperparameters were assumed to be known. Using the SE covariance function and noise structure as before, the complete set of hyperparameters are $\phi = \left\{l, \sigma_f^2, \sigma_n^2\right\}$. The posterior predictive distribution is now more formally written as $\mathbf{f}_*|\mathbf{y}, \phi \sim \mathcal{N}\left(\bar{\mathbf{f}}_*, \operatorname{cov}\left(\mathbf{f}_*\right)\right)$; the posterior predictive mean and covariance are the same as in Eq. 3.12 and Eq. 3.13.

Building the emulator requires the learning of the hyperparameters, ϕ, from the training set. Two different approaches to building the emulator are the empirical Bayes and the full Bayesian approaches. The empirical Bayes approach uses optimization schemes to find point estimates to the hyperparameters, while the full Bayesian approach uses the MCMC process to draw samples from the posterior distribution of the hyperparameters [4]. The empirical Bayes process is much faster than the MCMC process, but lacks cross-validation to determine whether the "optimized" hyperparameters yield the "best" results.

Empirical Bayes Process

Empirical Bayes optimizes the hyperparameter values by maximizing the marginal likelihood, Eq. 2.2. Given the notation for a training set X and training outputs $\mathbf{y}$, the marginal likelihood becomes:

$$p\left(\mathbf{y}|X\right) = \int p\left(\mathbf{y}|\mathbf{f}, X\right) p\left(\mathbf{f}|X\right) d\mathbf{f}. \tag{3.15}$$

The prior, $p\left(\mathbf{f}|X\right)$, is just the GP prior specified above with the likelihood, as before, being the Gaussian likelihood:

$$\mathbf{y}|\mathbf{f} \sim \mathcal{N}\left(\mathbf{f}, \sigma_n^2 \mathbb{I}\right). \tag{3.16}$$

Integrating out the latent function gives the function that must be maximized during the Empirical Bayes optimization as:

$$\log\left[p\left(\mathbf{y}|X\right)\right] = -\frac{1}{2}\mathbf{y}^{\mathrm{T}}\left[\mathbf{K}\left(X,X\right) + \sigma_n^2\mathbb{I}\right]^{-1}\mathbf{y} - \frac{1}{2}\log\left[\left|\mathbf{K}\left(X,X\right) + \sigma_n^2\mathbb{I}\right|\right] - \frac{N}{2}\log\left|2\pi\right|. \tag{3.17}$$

Eq. 3.17 is referred to as the log-marginal likelihood and has easily interpretable terms. On the right hand side of the equation, the first term represents the data fit, the second term represents the complexity penalty which depends only on the covariance function and the inputs, and the last term is a normalizing constant [4].

Full Bayesian Process

In the full Bayesian approach a prior is placed on the hyperparameters, $p\left(\phi\right)$, with the goal being to sample the posterior distribution conditioned on the training data [4]. The joint posterior is written as:

$$p\left(\mathbf{f},\phi|\mathbf{y},X\right) = \frac{p\left(\mathbf{y}|\mathbf{f},\phi,X\right)p\left(\mathbf{f}|\phi,X\right)p\left(\phi\right)}{\int p\left(\mathbf{y}|\mathbf{f},\phi,X\right)p\left(\mathbf{f}|\phi,X\right)p\left(\phi\right)d\mathbf{f}d\phi}, \tag{3.18}$$

since the latent function is also unknown. Since the latent function variables can be integrated out, the posterior distribution on the hyperparameters can be written as:

$$p\left(\phi|\mathbf{y},X\right) \propto p\left(\mathbf{y}|\phi,X\right)p\left(\phi\right). \tag{3.19}$$

The likelihood, $p\left(\mathbf{y}|\phi,X\right)$, is given by Eq. 3.7, with the condition on ϕ and X showing that it is for constructing the emulator [4]. The hyperparameter prior must still be specified, and the easiest prior to implement is a "flat" prior, where $p\left(\phi\right) \propto 1$ [4]. This means there is no bias, *a priori*, on the hyperparameter values. Drawing samples from the hyperparameter posterior distribution in Eq. 3.19 can be done with the MCMC techniques mentioned earlier.

3.5 GPR Emulator Uncertain Parameter Calibration

Once the GPR emulator is constructed, it can be used to calibrate the uncertain parameters in lieu of the computer code. In Chapter 2 the uncertain parameter calibration with MCMC was demonstrating a mapping function between the uncertain parameters, θ, and the computer code results. The experimental data points are denoted as $\mathbf{y_e}$, with the computer code output now denoted as $\mathbf{y}$ in order to prevent confusion between the latent variables and the computer code results. The total likelihood can now be broken into two parts: the likelihood between the experimental data and the computer code predictions, $p\left(\mathbf{y_o}|\mathbf{y}\right)$, and the likelihood between the computer code predictions and the uncertain parameters, $p\left(\mathbf{y}|\mathbf{x}_{cv},\theta\right)$. The posterior distribution is now written as the joint-posterior distribution between the uncertain parameters and the computer code predictions, conditioned on the experimental data [4]:

$$p\left(\mathbf{y},\theta|\mathbf{y_o}\right) \propto p\left(\mathbf{y_o}|\mathbf{y}\right) p\left(\mathbf{y}|\mathbf{x}_{cv},\theta\right) p\left(\theta\right). \tag{3.20}$$

Using N_O experimental data points, the experimental data points and computer code predictions vectors are written as $\mathbf{y_e} = \left[y_{e,1}, y_{e,2}, \ldots, y_{e,N_O}\right]^{\mathrm{T}}$ and $\mathbf{y} = \left[y_1, y_2, \ldots, y_{N_O}\right]^{\mathrm{T}}$, respectively. If the GPR emulator were already built from a given training set and the hyperparameters were determined using one of the previously mentioned approaches, the joint-posterior distribution between the emulator predictions, $\mathbf{y}_*$, and the uncertain parameters, θ, would be:

$$p\left(\mathbf{y}_*,\theta|\mathbf{y_o},\left\{X,\mathbf{y}\right\},\phi\right) \propto p\left(\mathbf{y_o}|\mathbf{y}_*\right) p\left(\mathbf{y}_*|\left\{\mathbf{x}_{cv},\theta\right\},\left\{X,\mathbf{y}\right\},\phi\right) p\left(\theta\right). \tag{3.21}$$

Because the likelihood $p\left(\mathbf{y_o}|\mathbf{y}\right)$ and emulator posterior predictive distribution are both Gaussian, the emulator predictions can be integrated out of Eq. 3.21. The likelihood is now the GPR emulator modified likelihood, which is simply the emulator posterior predictive distribution with the measurement error added to the predictive variance, yielding the result:

$$\mathbf{y}_*|\left\{\mathbf{x}_{cv},\theta\right\},\left\{X,\mathbf{y}\right\},\phi \sim \mathcal{N}\left(\bar{\mathbf{f}}_*, \operatorname{cov}\left(\mathbf{f}_*\right) + \sigma_n^2\mathbb{I}\right). \tag{3.22}$$

The (integrated) posterior distribution, conditioned upon the experimental data is then [4]:

$$p\left(\theta|\mathbf{y_o},\left\{X,\mathbf{y}\right\},\phi\right) \propto p\left(\mathbf{y_o}|\left\{\mathbf{x}_{cv},\theta\right\},\left\{X,\mathbf{y}\right\},\phi\right) p\left(\theta\right). \tag{3.23}$$

3.6 GPR Demonstration

The same friction factor problem from Section 2.4 will be used to demonstrate use of the GPR model. The friction factor equation, Eq. 2.5, uses the same parameterization. Twenty-five artificial “experimental data” points were generated at the specified true-parameters values of $\exp(b) = 0.25$ and $\exp(c) = 0.10$ at evenly spaced intervals between Reynolds numbers of 5000 and 45000. The general process for using GPR is as follows:

1. From the uncertain parameter prior distribution, choose lower and upper bounds on each of the uncertain parameters.
2. Create the training set using a Latin hypercube sampling scheme.
3. Generate the training output by running the computer code at each training point within the training set.
4. Build the emulator.
5. Calibrate the uncertain parameters using MCMC, using the emulator in place of the computer code.

A total set of 50 training points was used to build the GPR emulator. A Gaussian prior was used for the uncertain parameters, B and C, as shown in Table 3.1.

Table 3.1: Gaussian prior mean and variance for the friction factor uncertain parameters.

Uncertain Parameter	Mean	Variance	Minimum	Maximum
B Coefficient	log(0.18)	0.179	-2.539	-0.846
C Exponent	log(0.20)	0.162	-2.414	-0.805

In order to construct the training set, the LHS sampling scheme in MATLAB called *lhsdesign* was used [12]. The output of *lhsdesign* is a set of scaled values between 0 and 1, as shown in Figure 3.2. The scaling minimum and maximum values of the uncertain parameters using the Gaussian priors were set at $\pm 2\sigma$ from the mean value. The training points were then built by turning the scaled output from *lhsdesign* into raw B and C values; this same process was also completed for the Reynolds numbers between 5000 and 45000. The raw training points then were used as the inputs to the computer code (friction factor formula) to produce the raw training set shown in Figure 3.3.

The GP emulator was built using AM-MCMC sampling with 2×10^4 burn-in samples and 2×10^4 covariance matrix samples with the AM proposal matrix, $\hat{\Sigma}$, being computed every

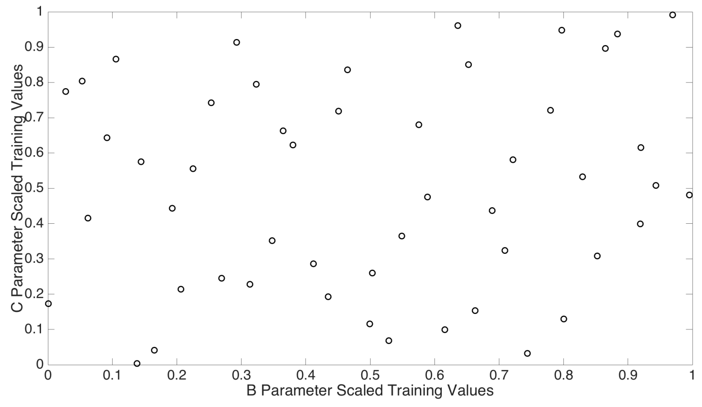

Figure 3.2: Training set scaled values of B and C parameters.

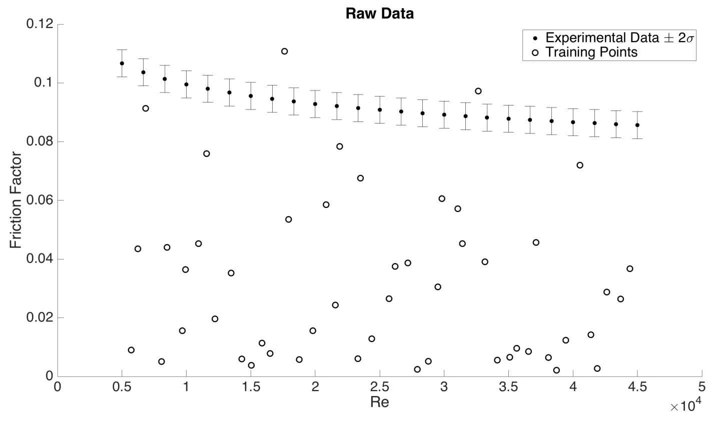

Figure 3.3: GPR training set with 50 training points.

1000 samplings using the past 1000 samples. The posterior distributions were then sampled using another AM-MCMC algorithm using 5×10^4 burn-in samples and 5×10^4 posterior distribution samples. The resulting posterior predictions of the emulator are shown in Figure 3.4, with the posterior distributions of the uncertain parameters shown in Figure 3.5. As reference, the directly sampled MCMC predictions and posterior distributions from Section 2.4 are also shown. The perfect GP emulator exactly reproduces the directly sampled MCMC predictions and posterior distribution. The performance of the GPR is strongly dictated by the training set, and Figure 3.6 shows that the training set covers the entire prior sample space well, but not the posterior sample space; the posterior sample space is defined by $\pm 2\sigma$ of the true, directly sampled B and C posterior distributions.

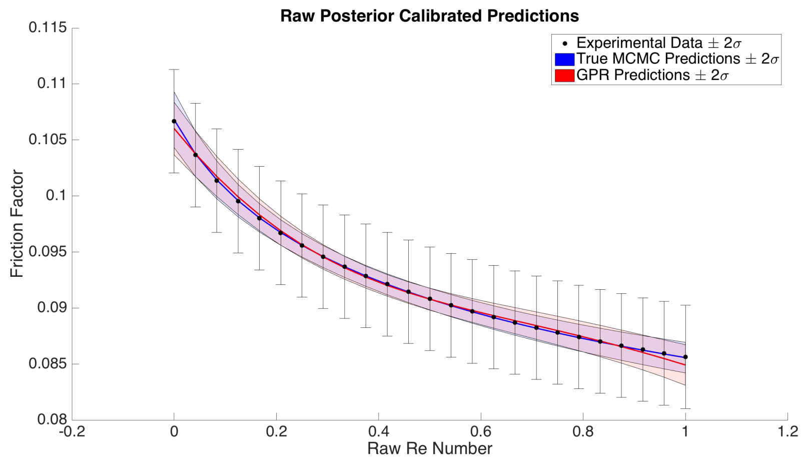

Figure 3.4: Posterior predictions from direct MCMC sampling and GP Regression.

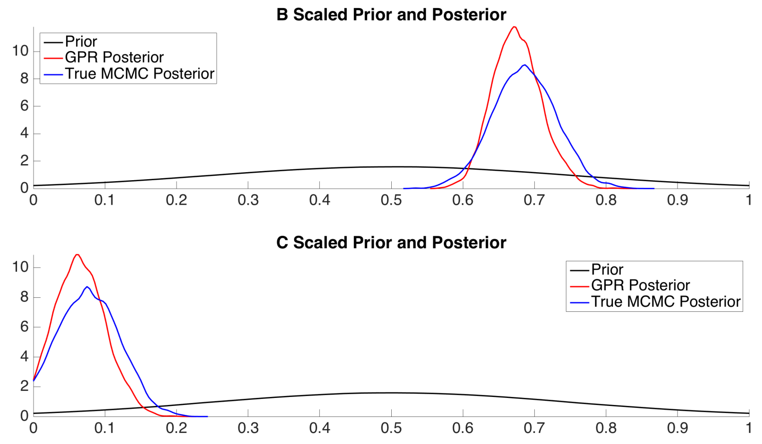

Figure 3.5: Posterior distributions from direct MCMC sampling and GP Regression.

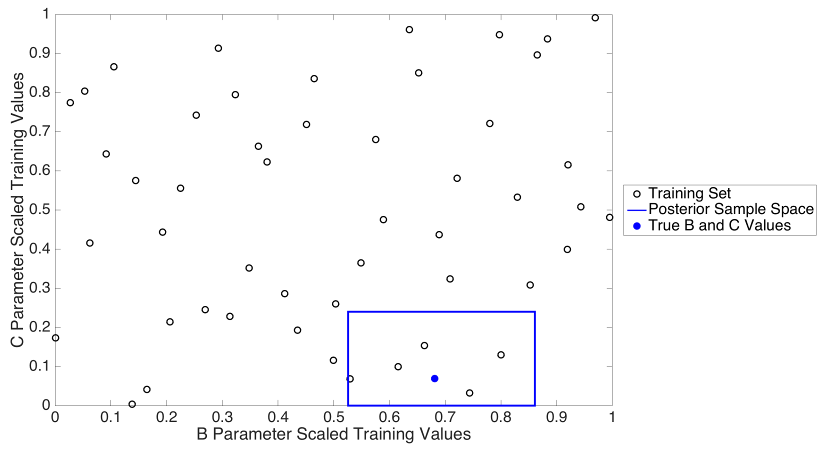

Figure 3.6: Training set with posterior sample space and known true B and C values.

Chapter 4

Covariance Functions

The flexibility of GP models raises the question of which covariance function to use for a given problem. The covariance function is a vital part of the Gaussian Process emulator, inherently encoding some important prior assumptions about the underlying function that we wish to model. The notion of similarity between data points is important: a basic assumption is that points with inputs $\mathbf{x}$ which are close to one another are likely to have similar target outputs, y [13]. In the Gaussian Process framework, the covariance function defines this similarity, or closeness. The covariance shorthand is defined as:

$$\operatorname{Cov}\left[f\left(\mathbf{x}\right), f\left(\mathbf{x}'\right)\right] = k\left(\mathbf{x}, \mathbf{x}'\right). \tag{4.1}$$

4.1 Covariance Function Properties

A covariance function is a positive-definite function of two inputs, $\mathbf{x}$ and $\mathbf{x}'$ [16]. A stationary covariance function depends only on the difference $\mathbf{x} - \mathbf{x}'$. Furthermore, if the covariance is only a function of $|\mathbf{x} - \mathbf{x}'|$ then it is called isotropic. This means that the probability of observing a particular dataset remains the same even if the $\mathbf{x}$ values are all moved by the same amount [16]. In contrast, non-stationary covariance functions will produce different predictions if the data are moved similarly. All of the following covariance functions have a σ_f^2 term at the beginning that specifies the signal covariance for each $\mathbf{x}$ and $\mathbf{x}'$.

4.2 Basic Covariance Functions

Linear (Lin.)

The linear (Lin.) covariance function is of the form:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2(\mathbf{x}_p - c)(\mathbf{x}_q - c), \tag{4.2}$$

where c is a hyperparameter used as an offset constant to determine which points all the lines in the posterior distribution go through; at this point, the function will have zero variance. The linear covariance function is non-stationary and therefore is dependent on the absolute location of the inputs. The linear covariance is too simple for most applications, but it is useful when combined with other covariance functions.

Squared Exponential (SE)

The squared exponential (SE) covariance function is defined as:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}\left(\mathbf{x}_p - \mathbf{x}_q\right)^{\mathrm{T}} M \left(\mathbf{x}_p - \mathbf{x}_q\right)\right), \tag{4.3}$$

where the subscripts p and q represent two different indices of the input $\mathbf{x}$. The hyperparameters are the signal variance σ_f^2 and the matrix M, which is a symmetric matrix usually given as:

$$M = \operatorname{diag}\left(l\right)^{-2}, \tag{4.4}$$

where l is a vector of positive values and each element, $l_1, \ldots, l_D$ plays the role of the characteristic length-scale for each input parameter [4].

The squared exponential (SE) function has several properties that make it the default covariance function for many Gaussian Process applications. The SE function has relatively few parameters to estimate, with each one easily interpretable. This function is also capable of learning any continuous function given enough data, under some conditions; this property makes the SE a universally applicable covariance function [16]. The SE function is also infinitely differentiable, meaning a Gaussian Process using this covariance function is very smooth [13].

Periodic (Per.)

The periodic (Per.) covariance function is written as:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 exp\left(-2M\sin^2\left(\pi\frac{|\mathbf{x}_p - \mathbf{x}_q|}{p}\right)\right), \tag{4.5}$$

where p determines the distance between repetitions of the function, and M is the matrix of characteristic length-scales [16].

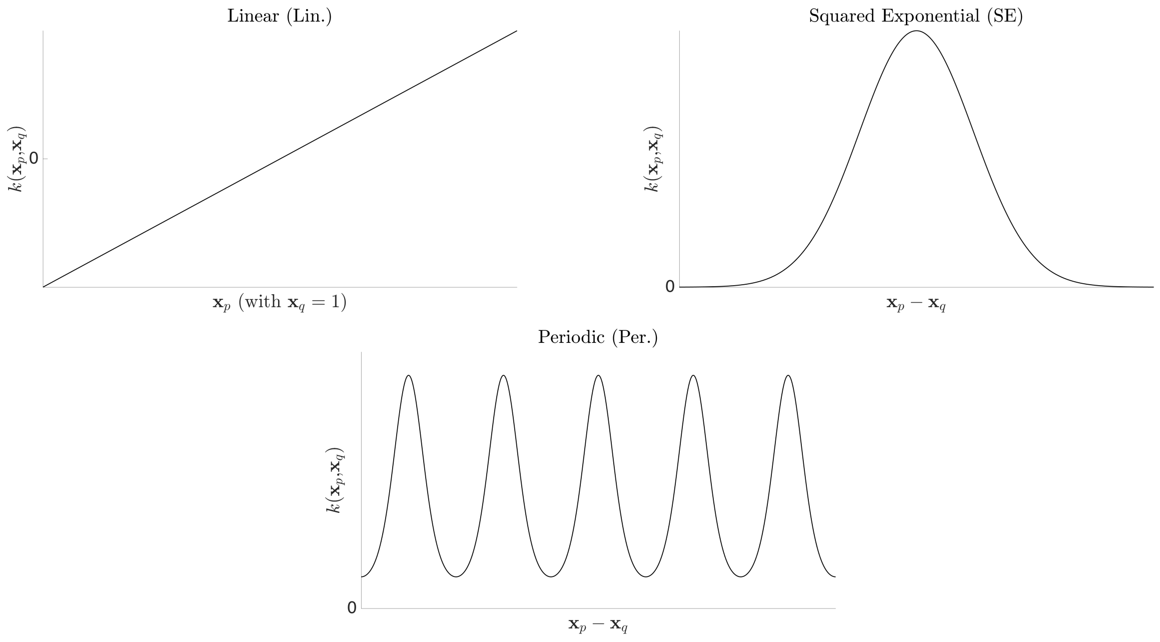

Figure 4.1: Basic covariance functions with arbitrary units.

4.3 Advanced Covariance Functions

Matérn Class

The Matérn class of covariance functions is given by:

$$k_{Matérn}(\mathbf{x}_p, \mathbf{x}_q) = \frac{\sigma_f^2}{\Gamma(\nu)2^{\nu-1}} \left[\sqrt{2\nu}\,(\mathbf{x}_p - \mathbf{x}_q)\right]^{\nu} K_\nu M \left[\sqrt{2\nu}\,(\mathbf{x}_p - \mathbf{x}_q)\right], \tag{4.6}$$

where σ_f^2 is the signal variance, Γ is the Gamma function, K_ν is a modified Bessel function [17] of the second kind of order ν and M is the matrix of characteristic length-scales [13]. It was named for the work of Bertil Matérn by Michael Stein [18]. The Matérn covariance function has an interesting feature in that it is $[\nu - 1]$ times differentiable [19]. Therefore, the hyperparameter ν can control the degree of smoothness. Special cases of the Matérn covariance function exist when ν is a half-integer: $\nu = p + 1/2$, where p is a non-negative integer [13]. For these cases, the covariance function is a product of an exponential and a polynomial of order p [13]. The first three forms are:

$$k_{\nu=1/2}(r) = \sigma_f^2 exp\left(-r\sqrt{M}\right)$$

$$k_{\nu=3/2}(r) = \sigma_f^2 \left(1 + \sqrt{3M}r\right) exp\left(-\sqrt{3M}r\right) \tag{4.7}$$

$$k_{\nu=5/2}(r) = \sigma_f^2 \left(1 + \sqrt{5M}r + \tfrac{5}{3}r^2 M\right) exp\left(-\sqrt{5M}r\right),$$

where $r = \mathbf{x}_p - \mathbf{x}_q$ and M is the matrix of characteristic length-scales [13]. As $\nu \to \infty$, the Matérn covariance function becomes the squared exponential covariance function. For Gaussian Process applications, $\nu = 1/2$ becomes very rough, and for $\nu \geq 7/2$ it becomes very hard to distinguish between values of $\nu \geq 7/2$ unless there is explicit prior knowledge about existence of the higher order derivates [13].

Rational Quadratic

The rational quadratic (RQ) covariance function:

$$k_{RQ}(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \left(1 + \frac{1}{2\alpha} \left(\mathbf{x}_p - \mathbf{x}_q\right)^{\mathrm{T}} M \left(\mathbf{x}_p - \mathbf{x}_q\right)\right)^{-\alpha}, \tag{4.8}$$

with $\alpha, M > 0$ can be a scale mixture (infinite sum) of squared exponential functions with different length-scales l. In this parameterization, the RQ covariance for $\alpha \to \infty$ becomes the SE covariance function [13]. Similar to the Matérn class, the RQ covariance has a tunable parameter α. Conversely, the RQ is infinitely differentiable for all α and is therefore closer in similarity to the SE covariance than the Matérn class covariance.

4.4 Combining Covariance Functions

A single covariance function can be used to model the data for the Gaussian Process emulator, but the power of the covariance function lies in the ability to combine covariance functions to reflect various aspects of the data. Combining covariance functions allows properties from each covariance function to influence the GP predictions and uncertain parameter calibration. In the end, combining covariance functions allows one the ability to include as much high-level structure as necessary for the data [16]. Two ways of combining covariance functions are through addition and multiplication. The notation for these is a shorthand without input arguments with the understanding that:

$$k_a \times k_b = k_a(\mathbf{x}_p, \mathbf{x}_q) \times k_b(\mathbf{x}_p, \mathbf{x}_q) \tag{4.9}$$

and

$$k_a + k_b = k_a(\mathbf{x}_p, \mathbf{x}_q) + k_b(\mathbf{x}_p, \mathbf{x}_q). \tag{4.10}$$

4.4.1 Multiplication of Covariance Functions

Multiplying two positive-definite covariance functions always results in another positive-definite covariance function [16]. Working with the covariance function allows the expression of high-level properties of functions that do not necessarily have a simple parametric form. A few examples are:

- **Polynomial Regression**

  Multiplying L linear covariance functions together will create a polynomial of degree L without having to specify a particular polynomial covariance function.

- **Functions with Growing Amplitude**

  Multiplying by a linear covariance function means that the marginal standard deviation of the modeled function grows linearly away from the location given by the hyperparameter c [16].

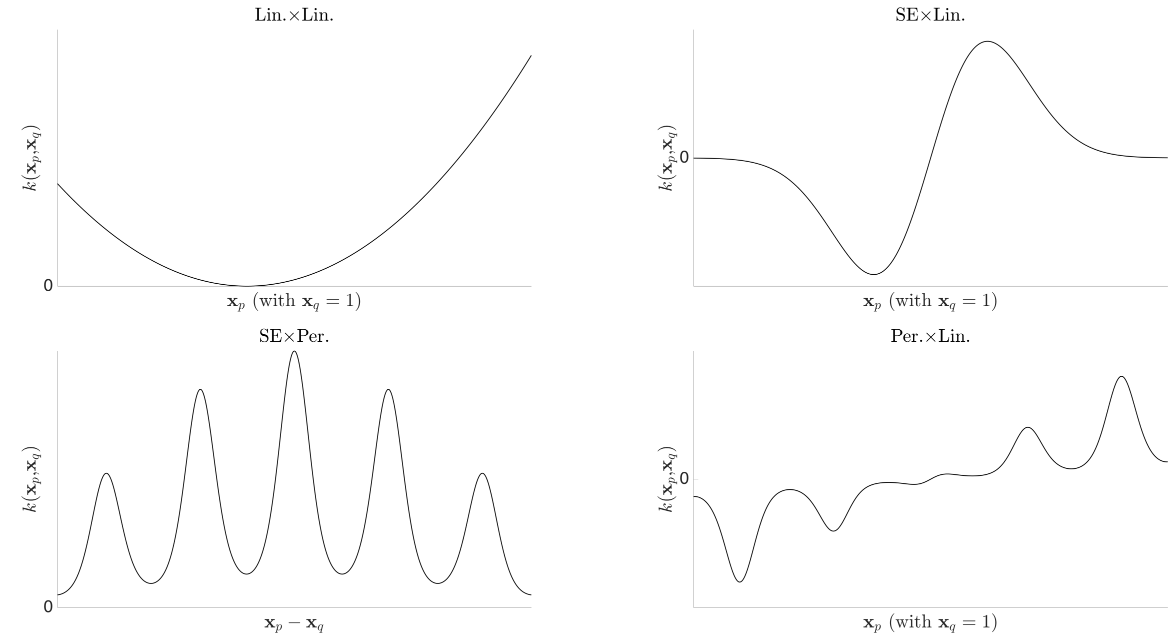

Figure 4.2: Examples of multiplication of covariance functions for the expression of different high-level properties.

Each of the covariance functions contributes particular characteristics when multiplied with another covariance function, as described in [16]:

- **Multiplication by SE** removes long range correlations from the original model, since $\mathrm{SE}(x, x')$ decreases monotonically toward 0 as $|x - x'|$ increases.
- **Multiplication by Lin.** is the same as multiplying the modeled function by a linear function such that $x \times f(x) \sim \mathrm{GP}(0, \mathrm{Lin.} \times k)$; this causes the standard deviation to vary linearly.
- **Multiplication by Per.** keeps the correlation between all pairs of functions within one period apart, thus allowing variation within each period.

4.4.2 Sum of Covariance Functions

Additivity is a useful modeling assumption that allows for strong assumptions about the individual components that make up the total sum [16]. Encoding the additivity into GP models is simple, too. If f_a and f_b are drawn independently from GP priors:

$$f_a \sim \mathcal{GP}(\mu_a, k_a) \tag{4.11}$$

$$f_b \sim \mathcal{GP}(\mu_b, k_b), \tag{4.12}$$

then the distribution of the sum of the functions is another GP:

$$f_a + f_b \sim \mathcal{GP}(\mu_a + \mu_b, k_a + k_b), \tag{4.13}$$

which is identical to Eq. 3.2. The covariance functions used to generate k_1 and k_2 can be of different classes, allowing for different underlying assumptions about the data, giving the user greater flexibility with their GP models. Figure 4.3 shows several combinations of the basic covariance functions summed together, but there is no limit to the number of covariance functions that can be used this way.

One of the benefits of summation of covariance functions compared to the product of covariance functions is the difference in ability to extrapolate away from the training set. The product of covariance functions is more limited because it allows a different function value for every combination of inputs, making the GP uncertain about the function at values away from the training points [16].

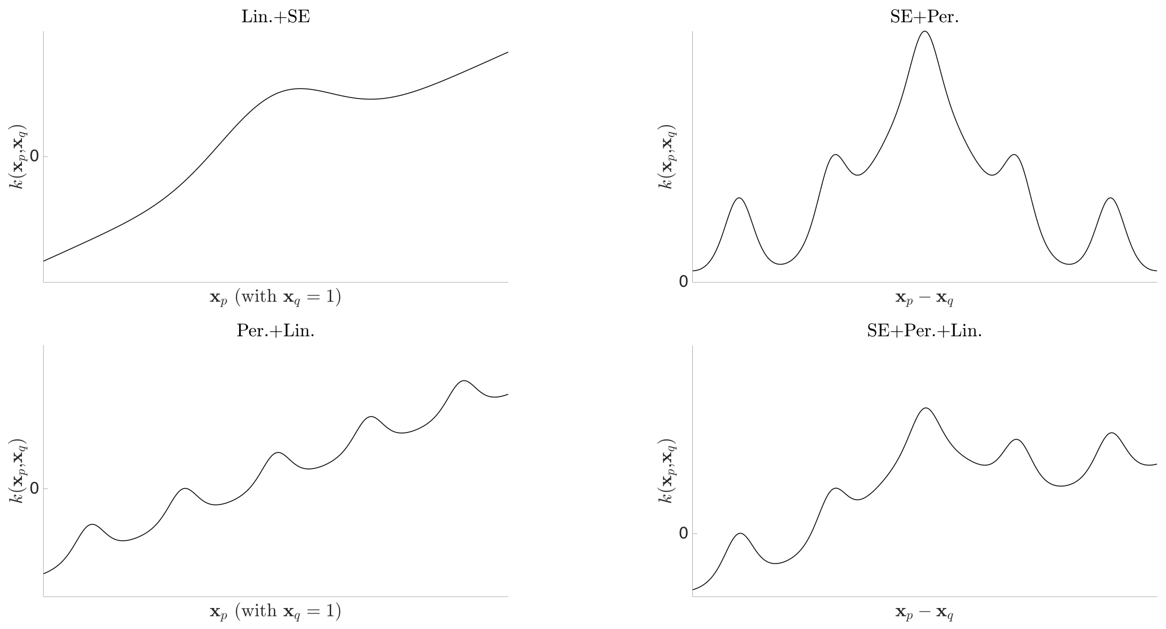

Figure 4.3: Sum of various combinations of covariance functions.

Chapter 5

Active Learning For Gaussian Process Regression

The Gaussian Process Regression (GPR) model provides a simple, flexible framework for modeling data. However, this simplicity does have its drawbacks. Primarily, the GPR's performance is limited by the ability of the training set to cover the posterior sample space. As mentioned before, this problem is unique in that because we are running a deterministic code we do not know the results until we run the computer code. The initial training set must cover the prior sample space as much as possible to give an accurate posterior sample space for the future placement of additional training points. Previous active learning methodologies using Gaussian Process applications have been used to decide how to categorize a data point (classification problems) or to reduce the number of data points used in the training set (sparse Gaussian Processes) [20, 21]. In both of these instances, the domain was known *a priori* when applying the active learning methodology. What makes this form of active learning unique in this instance is that the posterior domain is not known *a priori*.

5.1 Methodology

The general methodology for GPR active learning is as follows:

1. Choose lower and upper bounds on each of the uncertain parameters.
2. Create the training set using a Latin hypercube sampling scheme.
3. Generate the training output by running the computer code at each training point within the training set.
4. Build the emulator.
5. Calibrate the uncertain parameters using MCMC, using the emulator in place of the computer code.
6. Determine which uncertain parameter values to assign the new training point(s) and at which control variable location to place the new training point(s).

Steps 4 through 6 are iterated upon a certain number of times to arrive at a final posterior predictive mean and variance and uncertain parameter posterior distributions. The process workflow structure is outlined in Figure 5.1.

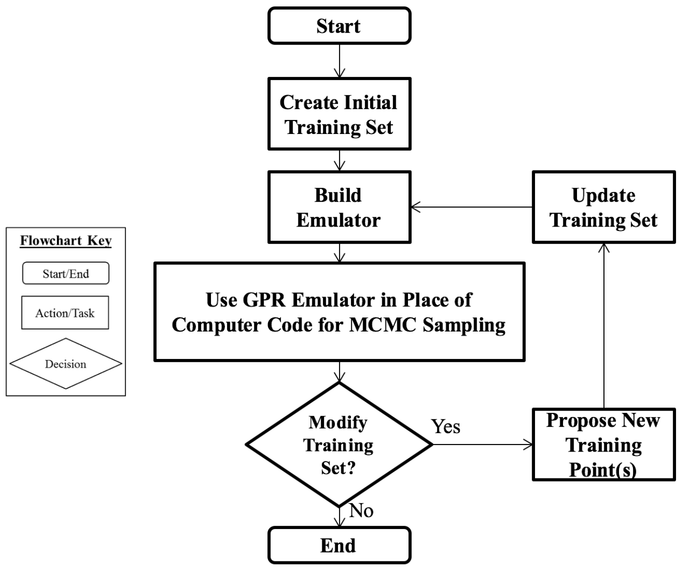
Start
Create Initial Training Set
Build Emulator
Update Training Set
Flowchart Key
Start/End
Action/Task
Decision
Use GPR Emulator in Place of Computer Code for MCMC Sampling
Modify Training Set?
Yes
Propose New Training Point(s)
No
End

Figure 5.1: Workflow for Gaussian Process Active Learning.

5.2 Control Parameter Training Values

The training matrix, X, consists of all the input parameters, which can be subdivided into the uncertain parameters and the "control" parameters. The control parameter is a variable for which we already have an experimental sample space of interest and have the explicit ability to control. For the friction factor problem, the input parameter is the Reynolds number because the Reynolds number can easily be changed by an experimental operator changing the mass flow rate. The decision criterion for the control parameter should favor using locations where the emulator predictive mean is far from the experimental data values and the predictive variance is large. Thus, the decision criterion to determine the location of the new input control parameter is defined as [22]:

$$\mathbf{x}_{*,\mathbf{cv}} = \text{argmax } \left[|\mathbf{y}_* - \mathbf{y_o}| \cdot \sqrt{var\left(\mathbf{y}_*\right)} \right], \tag{5.1}$$

since it accounts for differences between the emulator predictive mean and the experimental data, plus the emulator predictive variance.

5.3 Uncertain Parameters Training Values

Each uncertain parameter must provide information to the emulator, as described in section 3.3.1, to improve its utility. The method proposed by Gorodetsky and Marzouk designs experiments which minimize the integrated posterior variance (IVAR) of the Gaussian Process[23]. This method aims to minimize the integrated variance of the emulator posterior predictive variance:

$$\mathbf{x}_{*,\mathbf{unc}} = \text{argmin } \int_X k\left(x|\mathbf{x}\right) d\mu(x). \tag{5.2}$$

Evaluating Eq. 5.2 can be done using quadrature or Monte Carlo (MC) sampling techniques. Since the variance is usually a smooth function, quadrature schemes may work accurately for low-dimensional models. Monte Carlo schemes are generally better and more robust for higher order dimensionality problems and were the schemes of choice for applications of this methodology. A Monte Carlo scheme replaces the integral with a summation:

$$\int_X k\left(x|\mathbf{x}\right) d\mu(x) \approx \frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} k\left(x_i|\mathbf{x}_{*,\mathbf{cv}}\right), \tag{5.3}$$

where N_{mc} is the number of MC samples and $x_i \sim \mu$ [23].

5.4 Greedy and Batch Training

Addition of the active learning training points can be done either in a greedy or batch fashion. "Greedy" refers to the best single point being added each time. A greedy algorithm, for instance, always makes the choice that is locally optimal in the hopes that this choice will lead to a globally optimal solution [24]. Greedy implementations are easy to implement overall and much cheaper computationally than exhaustive searches. However, there is no guaranteed way to recognize whether a problem can be solved using a greedy algorithm. On the other hand, batch training, or dynamic programming, solves problems by combining the solutions to subproblems [24]. In the context of our work, this means that training points are added as groups of points instead of as single points at a time. This approach is typically applied to optimization problems having many possible solutions. Batch training is advantageous because interactions between the training points are taken into account directly [23].

Adding greedy training points for GPR often results in point "pile-up," with several training points being added at, or near, another training point value. Greedily adding T training points also requires retraining the emulator and then recalibrating the uncertain parameters before adding another training point, thus, a total of $T + 1$ iterations. But, each additional training point adds information to the covariance matrix and, in principle, provides a more accurate covariance matrix for better future predictions and training point additions.

Adding training points in batch allows for several alternative benefits as well: avoiding pileup, reducing the number of iterations needed to retrain the emulator and calibrate the uncertain parameters, and providing the freedom to modify the batch size. The added freedom to modify the batch size allows for tuning of the batch size to allow for more exploration or exploitation. For example, smaller batch sizes can be chosen when the hyperparameters are changing rapidly between iterations (exploration) and batch sizes can be increased once the hyperparameters converge (exploitation) [23]. Tuning of the batches was not explored in our work.

5.5 Friction Factor Revisited

The friction factor problem from Section 3.2 was repeated with the active learning methodology implemented. In the first GPR demonstration, 50 training points were selected from a LHS scheme using the *lhsdesign* MATLAB function [12]. The active learning methodology uses 20 initial training points (10 per uncertain parameter) and is created using the same LHS scheme in MATLAB. The SE covariance function was used to demonstrate the improvements obtained from using the active learning alone, while not modifying the covariance function. The GP emulator was built using AM-MCMC sampling with 2×10^4 burn-in samples and 2×10^4 covariance sampling samples with the AM proposal matrix, $\hat{\Sigma}$, being computed every 1000 samplings using the past 1000 samples. The posterior distributions were then sampled using another AM-MCMC algorithm with 5×10^4 burn-in samples and 5×10^4 posterior distribution samples. After performing GPR with the initial 20 training points, an additional 20 training points were added using the active learning methodology described in Chapter 5. These additional points were added in a single batch of 20 points rather than in a greedy fashion. The emulator was then rebuilt and the posterior predictions were sampled using the uncertain calibrated parameters. Figure 5.2 shows the initial training points (black) and the additional training points (red) with the experimental data given as point estimates with their 95% confidence interval given by the error bars. Notice how the initial training points hardly cover the experimental data. This is a product of the LHS scheme trying to cover the B and C parameter sample space as much as possible. The 20 additional training points placed using the IVAR method in Eq. 5.2 at locations specified by Eq. 5.1 cover the experimental data very thoroughly. Thus, the active learning algorithm is able to propose new training points that cover the posterior sample space very well. But, how well does the addition of these new training points improve the performance of the emulator?

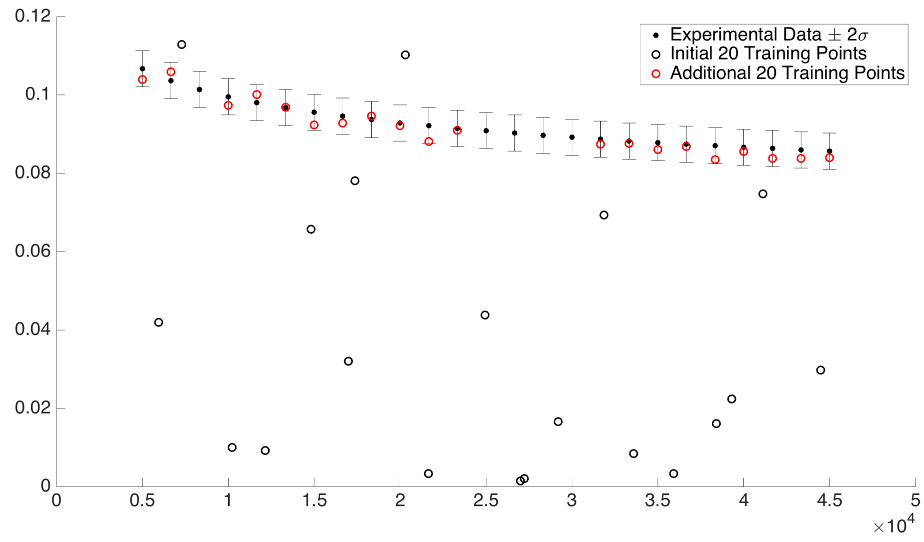

Figure 5.2: Active learning training set with the initial 20 training points (black) and the additional 20 training points (red) using the squared exponential (SE) covariance function.

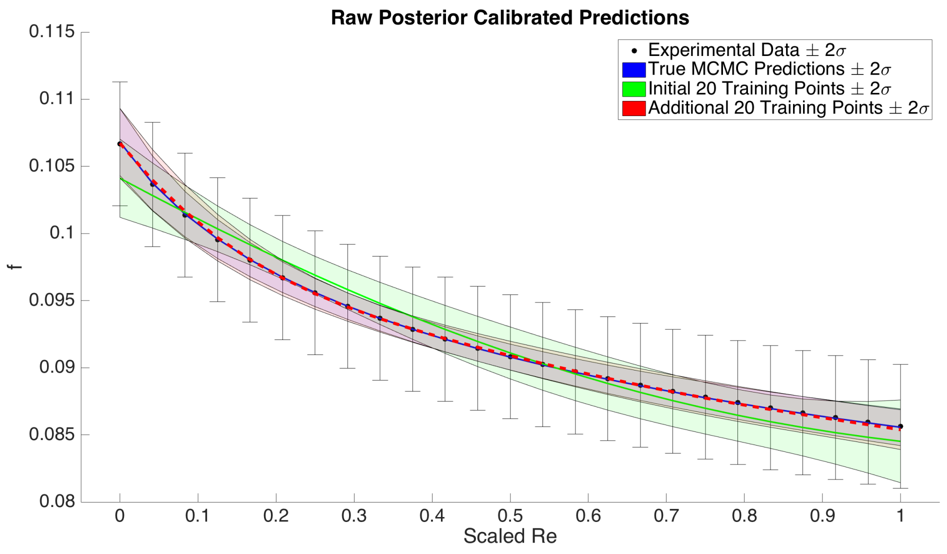

Figure 5.3: Active learning posterior predictions of the GP with the initial training points (green), additional training points (red), and the true MCMC sampled predictions (blue) using the squared exponential (SE) covariance function.

The posterior predictions are shown in Figure 5.3 with the true directly sampled MCMC predictions from Section 2.4 shown in blue, the emulator predictions from the initial training set in green, and the emulator predictions with the 20 additional training points shown in red. Clearly, the 20 additional training points improve the predictive performance of the emulator and match the directly sampled MCMC predictions. The 95% confidence intervals for the predictions, shown with the transparent area of the same color as the mean prediction, narrow when the additional training points are added and match the uncertainty present in the directly sampled MCMC example. Therefore, this active learning methodology produces an emulator which produces the same predictive quality as if the true computer code were being directly MCMC sampled.

For the desired applications of this methodology, the uncertain parameter posterior distributions are greatly important to the performance of the emulator. Figure 5.4 shows the uncertain parameter distributions for both B and C parameters with their respective priors (black) and true MCMC sampled posteriors (blue); the sample color scheme is the same as before for the initial training points and the additional training points. Notice that the initial posterior distributions have similar precision to the true posteriors, but the accuracies are poor with each distribution being roughly 0.10 scaled distance away from the true posterior mean. The posteriors from the additional 20 training points are nearly identical to the true posteriors. There are small deviations from the true posterior distribution at the modes and towards the tails, but the overall accuracy and precision of the emulator posterior distribution matches the true, directly sampled MCMC posterior distribution quite well. Figure 5.5 also demonstrates how the additional training points chosen by the IVAR criterion cover the posterior sample space well. The linear appearance of the additional training points is due to the B and C uncertain parameters being highly correlated. Thus, the active learning methodology improves the performance of the simple Gaussian Process emulator, and even uses fewer training points than the comparable demonstration described in Section 3.6.

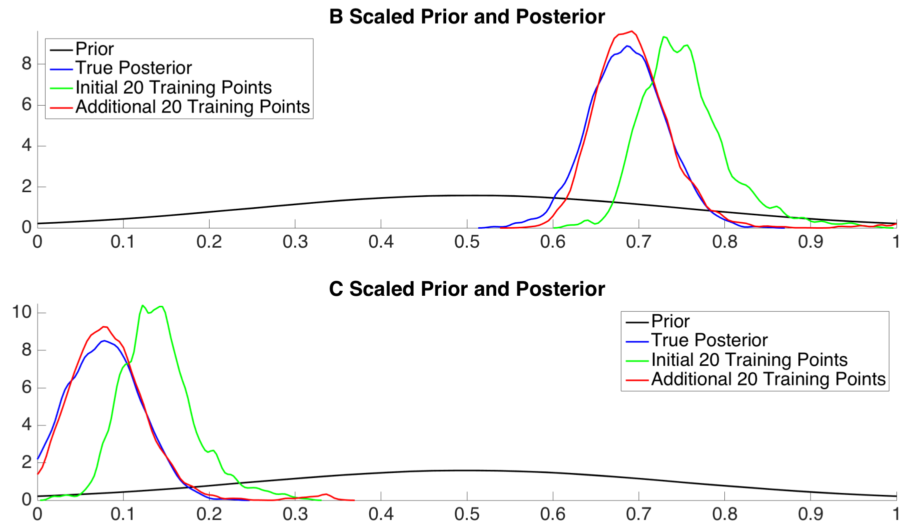

Figure 5.4: Active learning posterior distributions from the initial training points (green), additional training points (red), and the true MCMC sampled posterior distribution (blue) using the squared exponential (SE) covariance function.

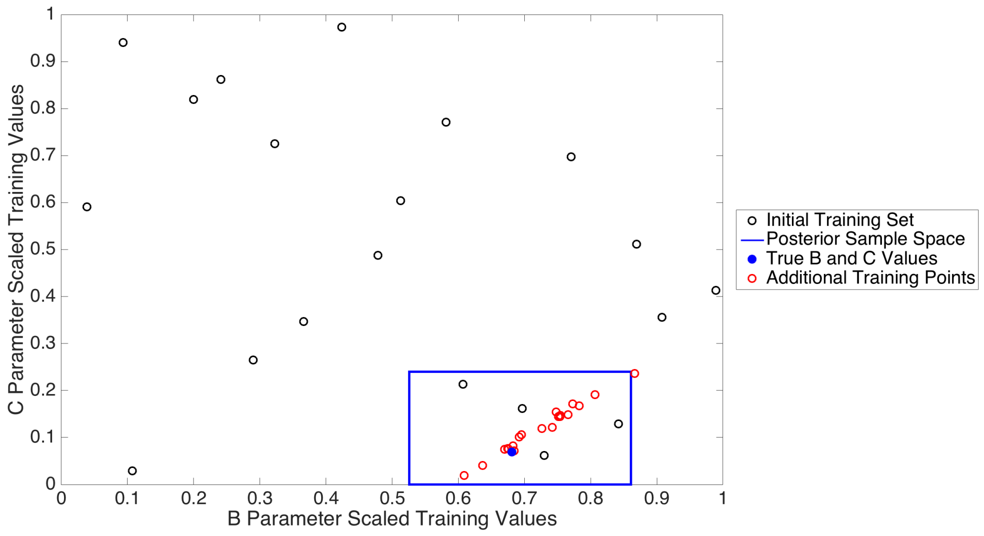

Figure 5.5: Initial and additional training points with posterior sample space and known true B and C values.

5.6 Influence of the Covariance Function

After demonstrating that the active learning methodology does improve the performance of the GP emulator, the next step is to investigate the impact of modifying only the covariance function. As stated before, the covariance function encodes the beliefs about the computer code being emulated.

There are, in fact, several methods for comparing covariance functions. The marginal likelihood is the chosen criterion here since it balances the fit and complexity of a model [25]. Conditioned upon kernel parameters, the marginal likelihood of a GP can be computed analytically by Eq. 2.2. In addition, if one compares GP models by the maximum likelihood value obtained after optimizing their hyperparameters, then all else being equal, the model having more parameters will be chosen; this introduces a bias in favor of more complex models. Overfitting can be avoided by integrating the marginal likelihood over all parameters, but this integral is difficult to compute in general. Instead, the integral can be loosely approximated using the Bayesian information criterion (BIC) [26]:

$$\mathrm{BIC}\left(X\right) = \log\left[p\left(\mathbf{y}|X\right)\right] - \frac{1}{2}\left|X\right| \log N, \tag{5.4}$$

where $\log\left[p\left(\mathbf{y}|X\right)\right]$ is the log-marginal likelihood from Eq. 2.2 with the optimized hyperparameters, $|X|$ is the number of hyperparameters for the covariance function, and N is the number of training points. This provides an adequate estimation of the “value” of each covariance function with a penalty term for the complexity of the covariance function.

In addition to BIC, the predictive capability of the emulator is also of great importance. Each covariance function combination requires comparing the emulator predictive mean to the experimental data using the root mean square error (RMSE) method:

$$\mathrm{RMSE}\left(\mathbf{y}_{*}, \mathbf{y}_{\mathbf{o}}\right) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{y}_{*,i} - \mathbf{y}_{\mathbf{e},i}\right)^{2}}, \tag{5.5}$$

where $\mathbf{y}_{*}$ and $\mathbf{y}_{\mathbf{e}}$ are the predictive and experimental means, respectively. Table 5.1 shows the calculated BIC and RMSE for several covariance functions applied to the friction factor problem. Additionally, because the friction factor example is not computationally expensive, a comparison between the true directly sampled MCMC posterior distributions and the GP uncertain parameter posterior distributions is possible, appearing as the final column in Table 5.1 with the B and C parameters RMSE, respectively. The bolded values in the table correspond to the four best values obtained for the BIC and RMSE; a higher BIC is better, and a lower RMSE is better.

Table 5.1: Effect of covariance function on Gaussian Process emulator performance. A larger Bayesian information criterion (BIC) values is better, while a smaller root mean square error (RMSE) value is better. The three "best" values for each category are bolded, while the three "worst" values are underlined.

			RMSE for:	
Covariance Function		BIC	Predictive Mean	Posterior Distributions $[B, C]$
Squared Exponential (SE)		78.24	1.609×10^{-4}	$[\mathbf{0.379}, \mathbf{0.369}]$
Rational Quadratic (RQ)		**78.27**	2.155×10^{-4}	$[0.852, 0.817]$
Matérn($\nu = 3/2$)		$\underline{53.72}$	$\underline{6.568 \times 10^{-4}}$	$\underline{[1.080, 1.098]}$
Matérn($\nu = 5/2$)		74.18	1.368×10^{-4}	$[0.819, 0.777]$
Product	{SE,Linear}	69.91	$\underline{7.610 \times 10^{-4}}$	$\underline{[3.837, 1.678]}$
	{SE,Matérn($\nu = 5/2$)}	70.15	1.632×10^{-4}	$[\mathbf{0.195}, \mathbf{0.207}]$
	{RQ,Linear}	$\underline{54.86}$	2.807×10^{-4}	$\underline{[1.333, 1.221]}$
	{RQ,Matérn($\nu = 5/2$)}	77.48	1.399×10^{-4}	$[\mathbf{0.399}, \mathbf{0.393}]$
	{Matérn($\nu = 5/2$),Linear}	$\underline{39.82}$	$\underline{3.988 \times 10^{-4}}$	$[0.903, 0.818]$
Sum	{SE,Linear}	69.10	1.703×10^{-4}	$[0.647, 0.641]$
	{SE,SE}	69.70	$\mathbf{1.008 \times 10^{-4}}$	$[\mathbf{0.468}, \mathbf{0.442}]$
	{SE,Matérn($\nu = 5/2$)}	**88.92**	$\mathbf{1.041 \times 10^{-4}}$	$[0.857, 0.812]$
	{RQ,Matérn($\nu = 5/2$)}	**87.44**	$\mathbf{1.253 \times 10^{-4}}$	$[0.869, 0.811]$
	{Matérn($\nu = 5/2$),Linear}	75.30	1.463×10^{-4}	$[0.714, 0.701]$

The covariance function which had the smallest RMSE error for the uncertain parameter posterior distributions was the product of the SE and the Matérn with $\nu = 5/2$ covariance functions. For comparison to the simple SE covariance function, the training set, posterior predictions, and uncertain parameter posterior distributions are shown in Figures 5.6, 5.7, and 5.8, respectively. An interesting observation from Figure 5.6 is that the additional training points are not directly on top of the experimental data as they were with the SE covariance function in Figure 5.2. However, the BIC and RMSE values are comparable for the two covariance functions, with the {SE,Matérn($\nu = 5/2$)} having the better RMSE value on the posterior distributions. This helps illustrate how the "best" additional training points can vary between covariance functions. The posterior distributions for the uncertain parameters in Figure 5.8 with the additional points are nearly identical to the directly sampled MCMC of the friction factor function. Overall, both sets of figures help demonstrate the power of the active learning methodology in properly replicating the same results of direct MCMC sampling of the true friction factor computer code.

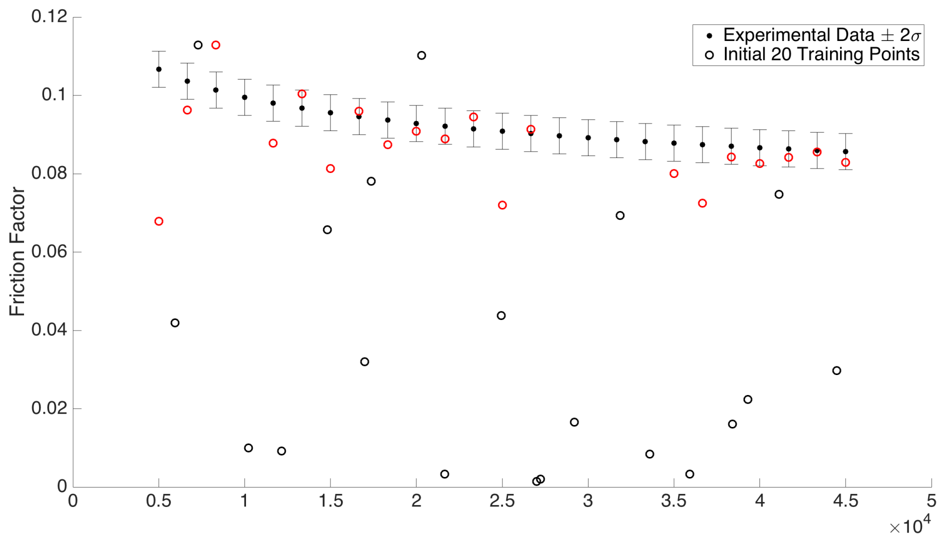

Figure 5.6: Training set from the product of the SE and Matérn with $\nu = 5/2$ covariance functions with the initial 20 training points (back) and the additional 20 training points (red).

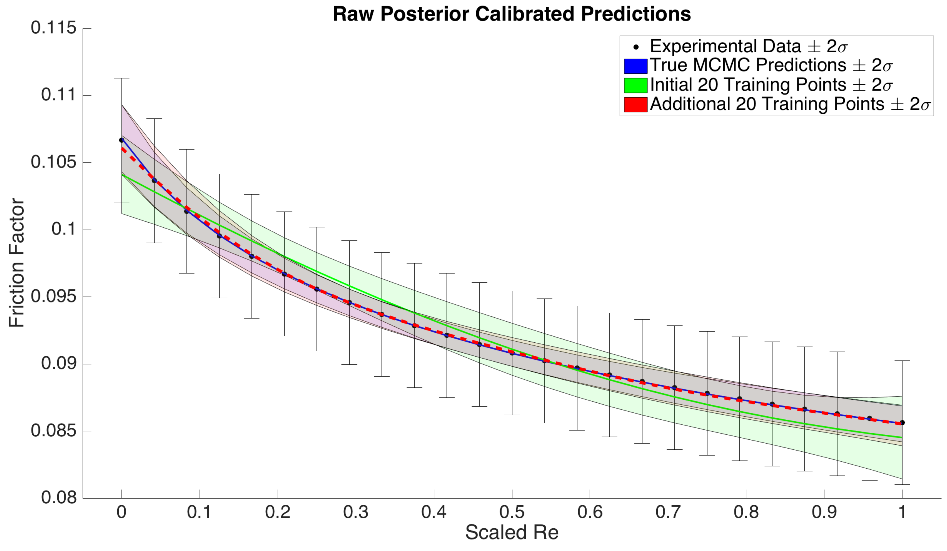

Figure 5.7: Posterior predictions from the product of the SE and Matérn with $\nu = 5/2$ covariance functions with the initial training points (green), additional training points (red), and the true MCMC sampled predictions (blue).

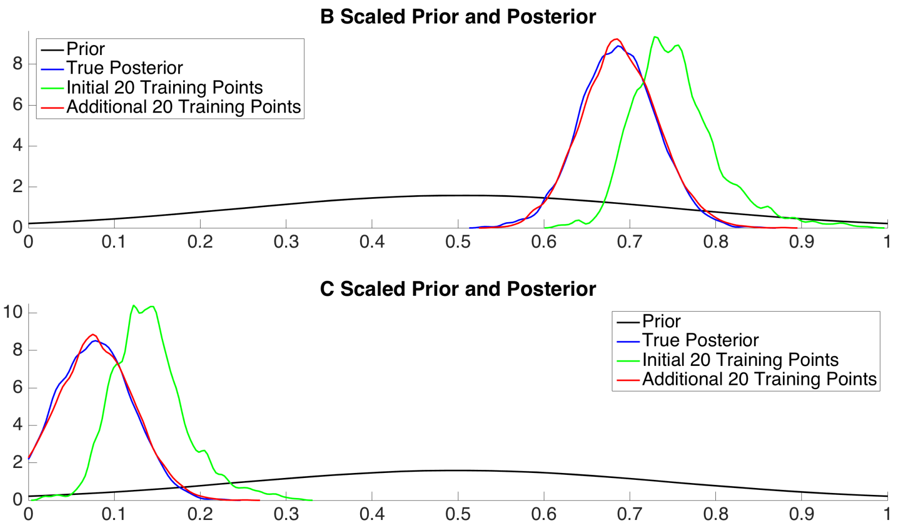

Figure 5.8: Posterior uncertain parameter distributions from the product of the SE and Matérn with $\nu = 5/2$ covariance functions from the initial training points (green), additional training points (red), and the true MCMC sampled posterior distribution (blue).

Chapter 6

Demonstration of Active Learning GPR with RELAP Models

The friction factor model previously used is helpful for demonstration purposes, but does not adequately represent the complex reactor safety analysis codes that this GPR active learning methodology was developed. Problems that can be modeled with system codes such as RELAP are the reason for GPR applications. Two RELAP simulations from Yurko [4] were chosen to demonstrate the GPR active learning methodology and to analyze the influence of the covariance function: the Cheng & Todreas friction factor correlation [27] and the Gapalakrishnan & Gillette (1973) data [28]. Both of these examples use the same generic RELAP model shown in Figure 6.1; dimensions and flow rates change for each problem, but the overall layout is still the same. As Yurko [4] describes, the inlet boundary condition is a time dependent junction which sets the mass flow rate. This inlet boundary condition is attached to a dummy volume, labeled as component 200 in Figure 6.1. Dummy volume 200 is attached to the bundle/channel of interest labeled component 400 via junction 201. When applicable to the simulation, Junction 201 is where the inlet nozzle loss coefficient is applied. Component 400 is attached to another dummy volume, labeled as component 100. Component 100 is attached to the outlet pressure boundary condition; both dummy volumes 200 and 100 have the wall friction turned off. Therefore, the pressure drop across the entire model is determined by subtracting the pressures within dummy volumes 200 and 100. With the outlet pressure fixed, the pressure from the time dependent inlet mass flow boundary condition adjusts depending upon the specified mass flow rate and the frictional characteristics of the components. The resulting pressure drop is considered to be the RELAP ΔP value [in pascals (Pa)].

The Cheng & Todreas RELAP simulation considers channel component 400 as a single channel called the driver fuel (DF). Furthermore, there is assumed to be no inlet nozzle loss coefficient. On the other hand, the Gapalakrishnan & Gillette (G&G) RELAP simulation breaks channel component 400 into three sections: the lower blanket (LB), the driver fuel (DF), and the upper blanket (UB). Additionally, the G&G simulation does include a nozzle loss coefficient applied at junction 201.

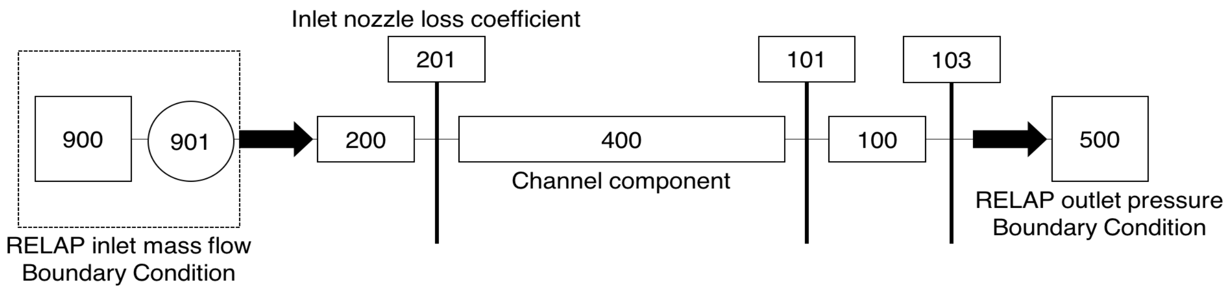
Inlet nozzle loss coefficient
201
101
103
900
901
200
400
100
500
Channel component
RELAP outlet pressure Boundary Condition
RELAP inlet mass flow Boundary Condition

Figure 6.1: Generic RELAP model illustration for the two RELAP simulations; replotted from [4].

RELAP User Defined Friction Factor Correlation

The RELAP user defined friction factor correlation is given for the laminar, transition, and turbulent regimes as [4]:

$$f^L = \frac{64}{Re \cdot \Phi_S}; \qquad \text{for } 0 \leq Re \leq 2200 \tag{6.1}$$

$$f^+ = \left(3.75 - \frac{8250}{Re}\right)\left(f^T_{3000} - f^L_{2200}\right) + f^L_{2200}; \qquad \text{for } 2200 < Re < 3000 \tag{6.2}$$

$$f^T = B \cdot Re^{-C}; \qquad \text{for } Re \geq 3000 \tag{6.3}$$

where Φ_S is the laminar shape factor, f^L is the laminar friction factor, f^+ is the transition regime friction factor, f^T is the turbulent friction factor, and f^T_{3000} and f^L_{2200} are the turbulent and laminar friction factors calculated at $Re = 3000$ and $Re = 2200$, respectively. The turbulent friction factor correlation is the unparameterized form of Eq. 2.5. The B and C parameters are referred to collectively as the turbulent friction factor parameters, and B, C, and Φ_S parameters are collectively referred to as the friction factor parameters [4].

GPR Emulator Characteristics

The emulators used for the two RELAP models were constructed in the same way as in Chapter 5. However, a key difference between the friction factor example demonstrated in Chapter 5 and the emulators built in this chapter are the uncertain parameters' prior distributions. The friction factor demonstration problems used a Gaussian prior with a mean and known variance. These priors were reparameterized using exponententials to avoid ever having negative values, which physically aren't possible. From now on, all uncertain parameters prior distributions will be uniform priors between lower and upper values taken from Yurko [4]. The uniform priors force the uncertain parameter values to stay within the prior bounds since any value outside of the prior has a density of zero.

6.1 Cheng & Todreas (C&T) RELAP Simulation

The Cheng & Todreas (C&T) RELAP simulation calibrates the three friction factor parameters in the driver fuel (DF) of the RELAP user-defined friction factor correlation. As before, the initial training set requires 10 initial training points for each uncertain input parameter, for a total of 30 initial training points. Twenty-five experimental data points are generated ranging from 1% of nominal core channel flow to 100% core channel flow. The experimental data used are generated from the Cheng & Todreas wire-wrapped bundle average friction factor correlation, hence the naming convention for this simulation [27]. The experimental error of the data is assumed to be that 95% of the probability was covered by $\pm 33\%$ around the mean data value of the first training point; this error was kept constant for each of the 25 training points. Figure 6.2 shows the 25 experimental data points with the initial training set. The error on the experimental data is so small that it only appear as a single line at each data point.

Table 6.1: Prior bounds for the Cheng & Todreas uncertain parameters. Values are taken from [4].

Uncertain Parameter	Minimum Value	Maximum Value
DF B Coefficient	0.138	0.23
DF C Exponent	0.15	0.25
DF Φ_S Shape Factor	0.125	1.875

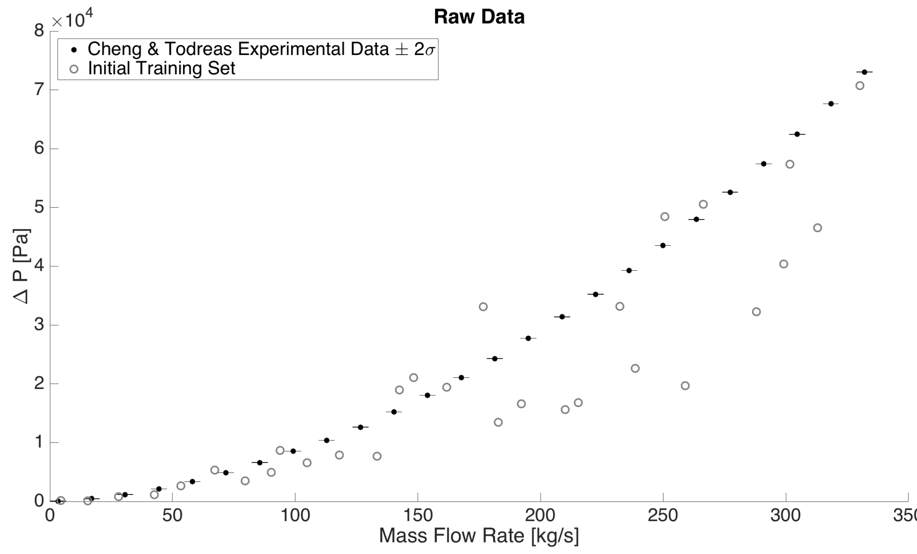

Figure 6.2: Cheng & Todreas (C&T) experimental data and initial GPR training set.

As before, the emulator is built using AM-MCMC sampling and the posterior distributions for the uncertain parameters are sampled using AM-MCMC. An additional 30 training points are added using the active learning methodology from Chapter 5 for a total of 60 training points. The emulator is rebuilt with the new training set of 60 points, and the uncertain parameters are calibrated. This procedure is repeated for the 13 different covariance functions tested, as shown in Table 6.2. The three best BIC and RMSE values are bolded, and the three worst values are underlined. The product of the SE and Linear covariance functions had the lowest predictive RMSE out of all the tested covariance functions. The squared exponential (SE) and rational quadratic (RQ) covariance functions have two of largest BIC values, yet the worst RMSE for the the predictive mean.

Table 6.2: C&T BIC and RMSE values with constant error. A larger Bayesian information criterion (BIC) values is better, while a smaller root mean square error (RMSE) value is better. The three "best" values for each category are bolded, while the three "worst" values are underlined. The product of the SE and Linear covariance functions had the lowest predictive RMSE out of all the tested covariance functions. The squared exponential (SE) and rational quadratic (RQ) covariance functions have two of largest BIC values, yet the worst RMSE for the the predictive mean.

			Predictive Mean
Covariance Function		BIC	RMSE
Squared Exponential (SE)		**196.5**	$\underline{258.2}$
Rational Quadratic (RQ)		**204.9**	$\underline{306.1}$
Matérn($\nu = 3/2$)		129.8	$\underline{229.5}$
Matérn($\nu = 5/2$)		168.7	151.5
Product	{SE,Linear}	$\underline{73.44}$	**74.14**
	{SE,Matérn($\nu = 5/2$)}	**198.4**	**120.6**
	{RQ,Linear}	$\underline{100.4}$	**112.8**
	{RQ,Matérn($\nu = 5/2$)}	171.2	150.3
	{Matérn($\nu = 5/2$),Linear}	$\underline{82.87}$	197.3
Sum	{SE,Linear}	196.0	187.0
	{SE,SE}	190.1	146.7
	{SE,Matérn($\nu = 5/2$)}	180.8	168.8
	{Matérn($\nu = 5/2$),Linear}	165.2	140.6

The uncertain parameter posterior distributions in Figure 6.5 are well formed and nearly Gaussian in appearance. These results are similar to the results in Yurko where he was using Function Factorization with Gaussian Process Priors (FFGP) and 50 training point values[4]. In his results, the C exponent was well defined with a scaled value of roughly 0.50, but the B coefficient and Φ_S shape factor were compressed to the prior maximum and minimum, respectively. The fact both the B and Φ_S parameters are well defined within the sample space of this active learning could explain why the C exponent has shifted slightly left to a mean of approximately 0.38. Ultimately, the active learning methodology achieved similar results using a much simpler methodology and 60 training points.

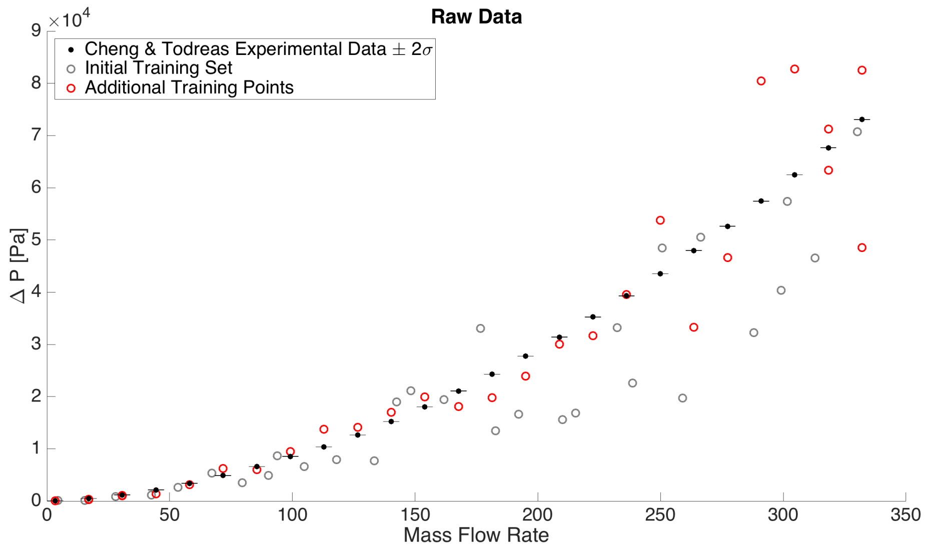

Figure 6.3: C&T additional training points from product of SE and Linear covariance functions.

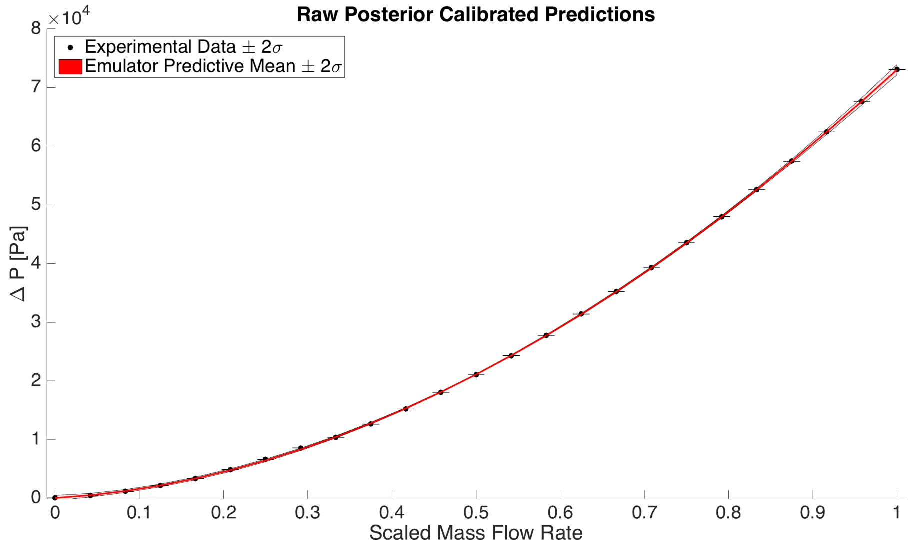

Figure 6.4: C&T emulator posterior predictions of the emulator from product of SE and Linear covariance functions.

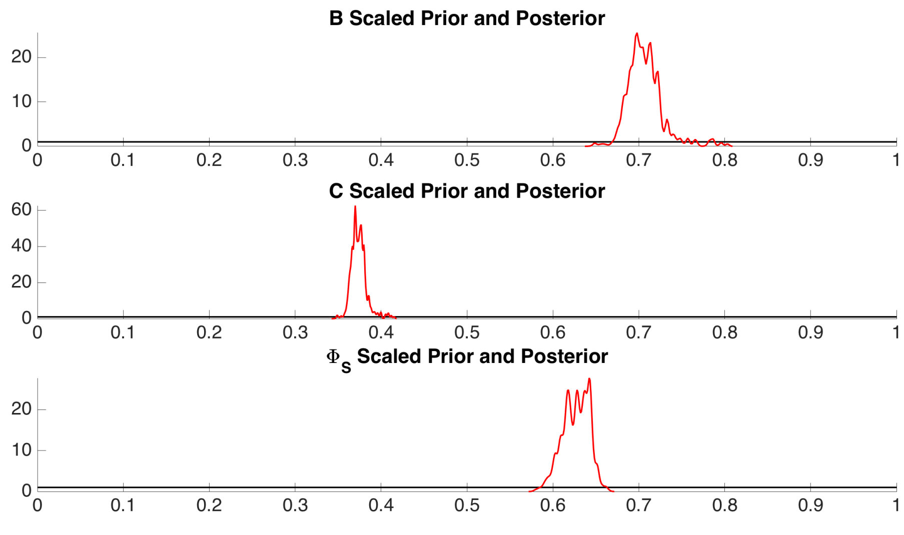

Figure 6.5: C&T posterior distributions of B, C, and Φ_S uncertain parameters from product of SE and Linear covariance functions.

6.2 Gopalakrishnan & Gillette (G&G) RELAP Simulation

The Gopalakrishnan & Gillette (G&G) RELAP Simulation has the same general layout as the Cheng & Todreas simulation shown in Figure 6.1, except that component 400 is no longer a single driver fuel section but three sections: the lower blanket (LB), driver fuel (DF), and upper blanket (UB). The LB and UB are assumed to be the same geometry, and thus have the same B, C, and Φ_S values. This means that three additional uncertain parameters are added to the RELAP model for the LB and UB friction factor parameters. An inlet nozzle loss coefficient uncertain parameter is also added since the G&G data is for the pressure drop across the entire channel and not a single bundle like the Cheng & Todreas simulation. Therefore, a total of seven uncertain parameters will be calibrated using the GPR: the three DF friction factor parameters, three LB and UB friction factor parameters, and the inlet nozzle loss coefficient.

The experimental data for the G&G simulation comes from Gopalakrishnan & Gillette [28]. Their data comes from hydraulically scaled water tests, making it difficult to tell whether the results used here come directly from experimental data or from empirical corrections. The "data" were taken from their paper and are shown in Figure 6.6.

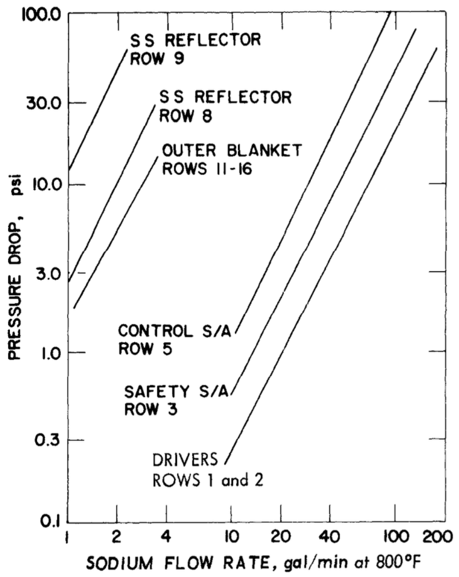

Figure 6.6: Gopalakrishnan & Gillette pressure drop experimental "data"

Table 6.3: Prior bounds for the Gopalakrishnan & Gillette uncertain parameters. Values are taken from [4].

Uncertain Parameter	Minimum Value	Maximum Value
LB/UB B Coefficient	0.138	0.23
LB/UB C Exponent	0.15	0.25
LB/UB Φ_S Shape Factor	0.125	1.875
DF B Coefficient	0.138	0.23
DF C Exponent	0.15	0.25
DF Φ_S Shape Factor	0.125	1.875
Inlet Nozzle Loss Coefficient	5	40

Only six experimental data points were taken from Figure 6.6 due to the inherent difficulty of getting accurate "data" from Figure 6.6. The experimental error of the data is assumed to be that 95% of the probability was covered by ±33% around the mean data value of the first training point; this error was kept constant for each of the 6 data points. Figure 6.7 shows the 6 experimental data points with the initial training set, with the error bars barely noticeable since they are so small.

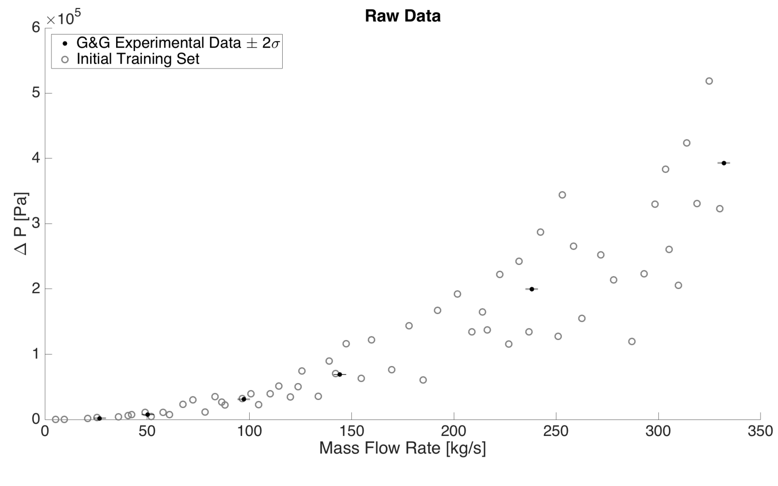

Figure 6.7: Gopalakrishnan & Gillette (G&G) experimental data and initial GPR training set.

Table 6.4: G&G emulator BIC and RMSE values with constant error.

Covariance Function		BIC	Predictive Mean RMSE
Squared Exponential (SE)		523.0	<u>8406</u>
Rational Quadratic (RQ)		**551.3**	4630
Matérn($\nu = 3/2$)		377.2	4215
Matérn($\nu = 5/2$)		498.2	4530
Product	{SE,Linear}	<u>332.5</u>	**3128**
	{SE,Matérn($\nu = 5/2$)}	498.8	4461
	{RQ,Linear}	<u>375.6</u>	**3199**
	{RQ,Matérn($\nu = 5/2$)}	530.3	4578
	{Matérn($\nu = 5/2$),Linear}	<u>366.9</u>	<u>4776</u>
Sum	{SE,Linear}	**541.6**	4313
	{SE,Matérn($\nu = 5/2$)}	519.1	<u>5777</u>
	{Matérn($\nu = 5/2$),Linear}	**540.6**	**3409**

The initial GPR emulator is built from an initial training set of 70 points (10 per uncertain parameter) and using AM-MCMC sampling. The posterior distributions for the uncertain parameters are also sampled using AM-MCMC. An additional 70 training points are added using the adaptive learning methodology from Chapter 5 for a total of 140 training points. The emulator is rebuilt with the new training set of 140 points and the uncertain parameters are calibrated. This procedure is repeated for the 13 different covariance functions tested, as shown in Table 6.2. The three best BIC and RMSE values are bolded, and the three worst values are underlined.

The product of the SE and Linear covariance functions had the lowest predictive RMSE out of all the tested covariance functions. The final training set, emulator posterior predictions, and uncertain parameter posterior distributions are all shown in Figures 6.8, 6.9, and 6.10, respectively. Unlike the Cheng & Todreas simulation, the G&G only had six experimental data points. This creates several non-ideal situations for the emulator. First, there are fewer locations to place additional training points. As shown in Figure 6.8, many of the training points are nearly on top of one another. As explained before in Section 3.3.1, if several training points have a shared input x_1 with other differing $x_{2,3,\ldots,D}$ values, there is no learning of the relation between x_1 and the output $f(\mathbf{x})$ from those points. Therefore, it is advantageous to have more experimental data points to help prevent the additional training points from being placed nearly on top of one another. Another shortfall of the few experimental data points is that it forces the emulator to become piecewise linear. The covariance functions tested all have varying degrees of smoothness associated with them, but none of them are designed to model a piecewise linear function. Furthermore, a piecewise linear covariance function should only be used if the true underlying function is believed to be a piecewise linear function. This clearly is not the case, and goes to demonstrate the necessity for more experimental data points in order to provide reliable GP results.

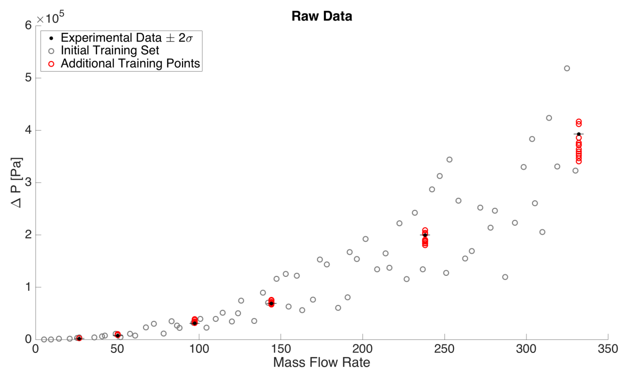

Figure 6.8: G&G data additional training points from product of SE and Linear covariance functions.

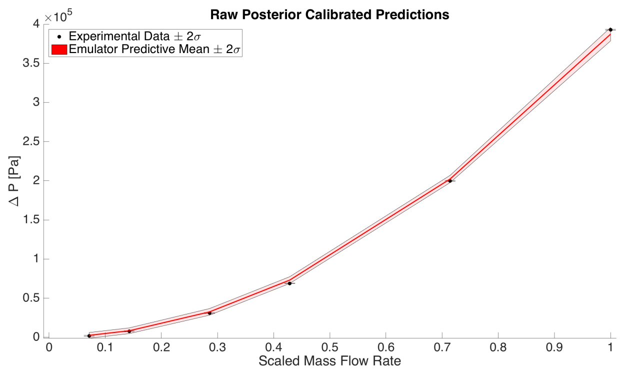

Figure 6.9: G&G emulator posterior predictions of the emulator from product of SE and Linear covariance functions.

The posterior distributions in Figure 6.10 further confirm this due to their lack of a clear Gaussian form. Only the inlet nozzle loss coefficient is clearly defined by the data, with the DF B and C parameters slightly less defined; the DF Φ_S is pushed up against the edge and not well defined with an extended distribution tail. The LB/UB parameters are hardly defined at all from the data, and there is little confidence in the ability of the GP to accurately determine those parameter values. This can also be contributed to the lack of experimental data points from the original G&G experiment. It should be noted that even obtaining the six experimental data points in the first place, as stated before, was difficult when there was confusion on what the data represented in the original 1973 paper from Gopalakrishnan and Gillette. Ultimately, any use of Gaussian processes should ensure that a proper quantity of experimental data points are obtained in order to prevent the piecewise linear function obtained from the G&G RELAP simulation.

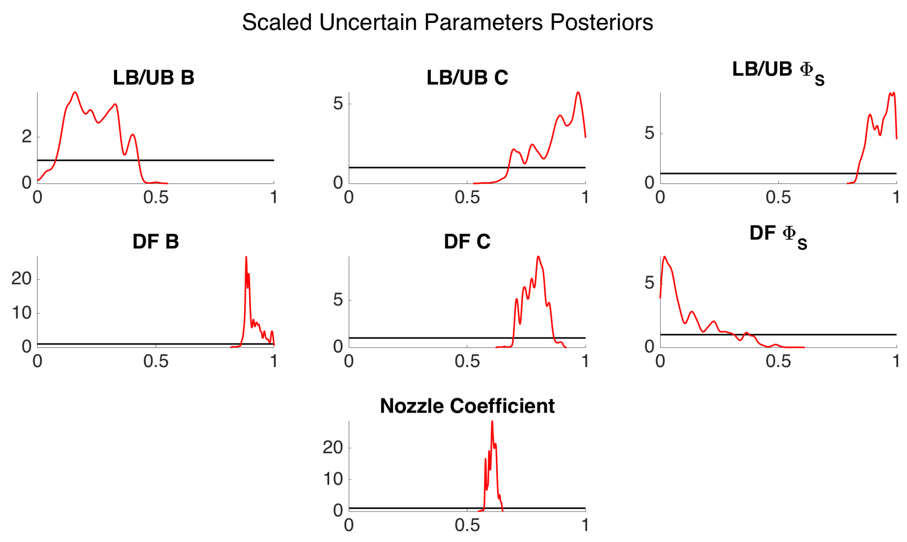

Figure 6.10: G&G data posterior distributions of the seven uncertain parameters from product of SE and Linear covariance functions.

6.3 Experimental Error Scaling as the Mean

In the previous simulations, the experimental error was assumed to be the same for each of the experimental data points. In certain situations where the experimenters are unable to resolve some constant error, this is an adequate assumption for the error. However, if the experimental error is designed to model the uncertainty presented by a measurement device, such as a pressure gauge, then a fixed experimental error is an incorrect assumption. The error of industrial pressure gauges is often given as a fraction of the mean measured value. While some measurement techniques are able to have uncertainty as low as 0.025%, a measurement uncertainty of 1% seems more realistic [29, 30]. Therefore, both RELAP simulations (C&T and G&G) were run again with the experimental error of the data assumed to be that 95% of the probability was covered by $\pm 1\%$ around the mean data value. Besides the experimental error, everything else was kept the same. Thus, only the results and implications of the experimental error as it pertains to GPR active learning will be discussed.

Cheng & Todreas RELAP Simulation

Table 6.5 shows the results with the experimental error scaling as 1% of the mean for the Cheng & Todreas RELAP simulation. As before, the product of the SE and Linear covariance functions had the lowest RMSE for the predictive mean of the emulator. When compared with Table 6.2, there is no large difference in performance for any of the covariance functions. There are minor differences in the RMSE, largely due to the fact that there is a larger experimental error, and that larger experimental error allows the emulator more freedom to explore different uncertain parameter combinations. However, the best performing covariance functions with the fixed experimental error are still, by and large, the best performing covariance functions when the experimental error scales as the mean. The product of the SE and Linear covariance functions was the best performing emulator with the larger error. Figure 6.11 shows the 30 training points the active learning methodology added to the initial training set.

Table 6.5: Cheng & Todreas emulator BIC and RMSE values with error as 1% of mean.

			Predictive Mean
Covariance Function		BIC	RMSE
Squared Exponential (SE)		**201.8**	163.0
Rational Quadratic (RQ)		**200.3**	155.2
Matérn($\nu = 3/2$)		$\underline{82.8}$	$\underline{286.5}$
Matérn($\nu = 5/2$)		186.4	$\underline{167.7}$
Product	{SE,Linear}	116.1	**111.6**
	{SE,Matérn($\nu = 5/2$)}	197.9	150.3
	{RQ,Linear}	$\underline{95.2}$	**129.7**
	{RQ,Matérn($\nu = 5/2$)}	**207.4**	143.4
	{Matérn($\nu = 5/2$),Linear}	$\underline{90.6}$	**135.7**
Sum	{SE,Linear}	179.8	155.2
	{SE,SE}	73.85	$\underline{217.3}$
	{SE,Matérn($\nu = 5/2$)}	177.1	158.3
	{Matérn($\nu = 5/2$),Linear}	175.5	158.5

Figure 6.12 shows the posterior predictions for the same emulator. Visually, it is indistinguishable from the fixed error posterior predictions in Figure 6.4. They both predict the experimental data very well, but the fixed experimental error emulator performs slightly better with a RMSE of 74.1 compared to the scaled experimental error RMSE of 111.

The posterior distributions for the uncertain parameters show the greatest difference between the treatment of the experimental error. As anticipated, the larger experimental error causes the variance of the posteriors in Figure 6.13 to be greater than in Figure 6.5. The mean of the B and C posteriors are nearly the same, confirming the accuracy of the posteriors; it would be very troubling if the posteriors from Figures 6.5 and 6.13 didn't look similar at all. The Φ_S parameter, however, doesn't have quite the same mean when the experimental error increases. This could be a result of the B and C parameters being highly correlated and the Φ_S parameter not. Figure 6.14 shows that the correlation of the B and C parameters are 1.00, while there is only a correlation of approximately 0.60 between B,C

and Φ_S. This would allow for the similar B and C posteriors distributions to have a different corresponding Φ_S posterior distributions.

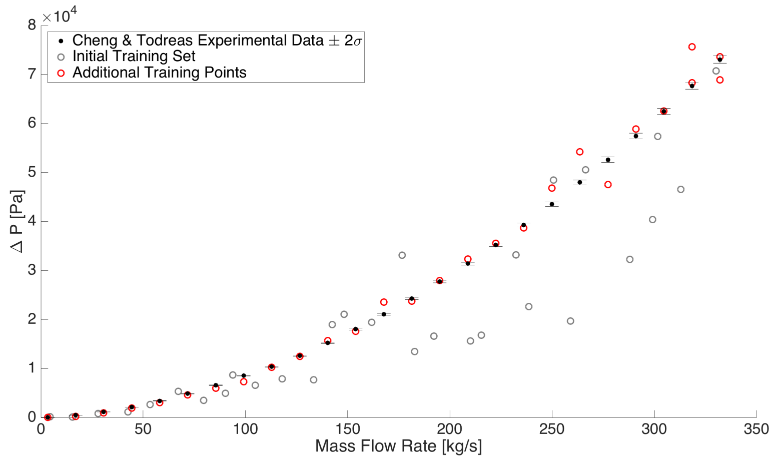

Figure 6.11: C&T additional training points with error as 1% of the mean from SE and Linear covariance functions.

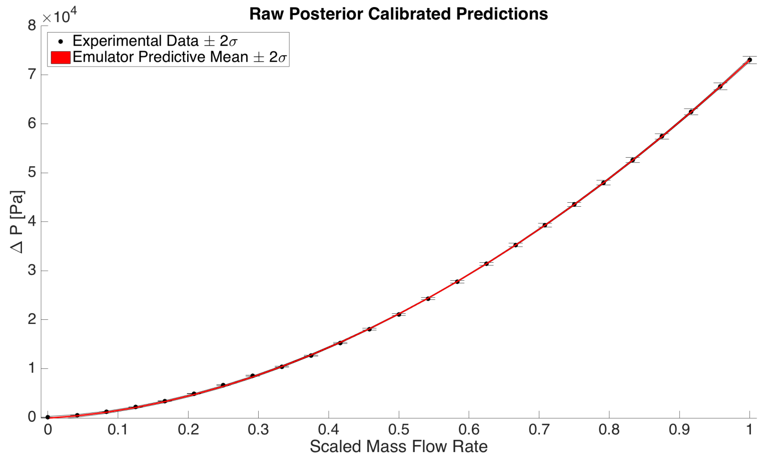

Figure 6.12: C&T emulator posterior predictions of the emulator with error as 1% of the mean from SE and Linear covariance functions.

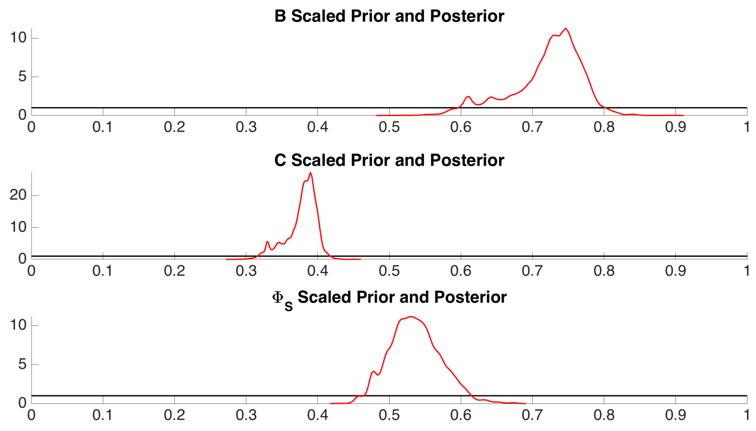

Figure 6.13: Posterior distributions of the B, C, and Φ_S uncertain parameters with error as 1% of the mean.

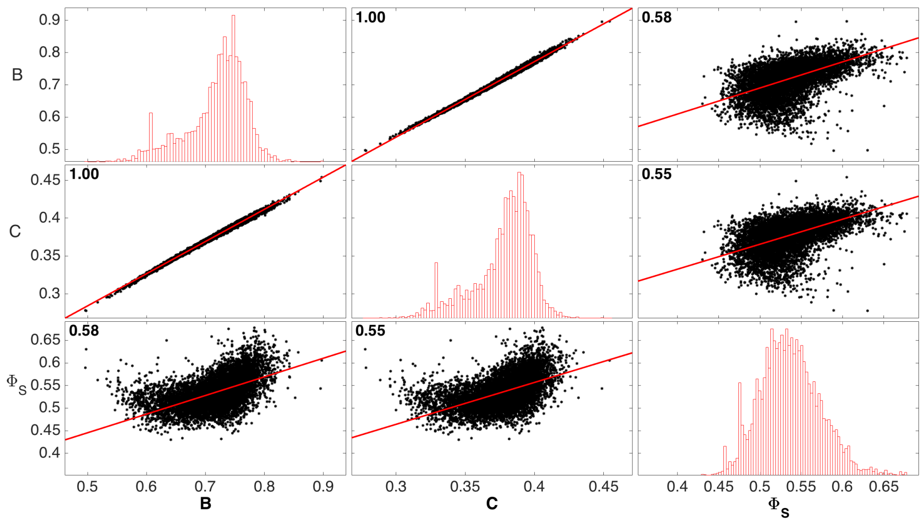

Figure 6.14: C&T correlation of B, C, and Φ_S uncertain parameters.

Gopalakrishnan & Gillette (G&G) RELAP Simulation

The results for the G&G RELAP simulation with the error scaling as the mean is much the same as the C&T simulation. From Table 6.6 it is clear that the covariance functions which performed well with the constant error continued to perform well with the experimental error scaling as the mean. It is, however, interesting that with the increased experimental error the predictive mean RMSE for the top three covariance functions in Table 6.6 is actually lower than the top three predictive mean RMSE values in Table 6.4. This is because the larger experimental error provides the MCMC sampling algorithm the opportunity to better explore the posterior uncertain parameter samples space. The smaller and constant experimental error limits the combinations of uncertain parameter values that the MCMC algorithm will sample, causing the MCMC sampling to not be adequately mixed as discussed before in Section 2.4. Figure 6.15 shows the 70 training points the active learning methodology added to the initial training set, with the improved posterior predictions shown in Figure 6.16.

Table 6.6: G&G emulator BIC and RMSE values with error as 1% of mean.

Covariance Function		BIC	Predictive Mean RMSE
Squared Exponential (SE)		426.4	4898
Rational Quadratic (RQ)		400.8	**2713**
Matérn($\nu = 3/2$)		355.3	3781
Matérn($\nu = 5/2$)		483.8	3306
Product	{SE,Linear}	407.6	**2971**
	{SE,Matérn($\nu = 5/2$)}	498.2	4872
	{RQ,Linear}	407.5	**3280**
	{RQ,Matérn($\nu = 5/2$)}	465.6	4571
	{Matérn($\nu = 5/2$),Linear}	413.2	3979
Sum	{SE,Linear}	**545.6**	5836
	{SE,Matérn($\nu = 5/2$)}	**508.2**	4703
	{Matérn($\nu = 5/2$),Linear}	**535.8**	3861

The uncertain parameter posterior distributions in Figure 6.17 show that there is little confidence in the ability for the data to adequately determine many of the uncertain parameters. The nozzle loss coefficient is the only uncertain parameters that is well explained by the experimental data in this case. Therefore, the GPR demonstrates that the G&G experimental data is not adequate to predict the value of any of the LB/UB and DF uncertain parameters. The best solution would be to do a simultaneous calibration of C&T DF parameters with all of the G&G uncertain parameters so the C&T data can help inform the G&G emulator. Ultimately, the G&G data does not provide enough information to properly determine the uncertain parameter posterior distributions and other experimental data should be used to determine the LB/UB uncertain parameters.

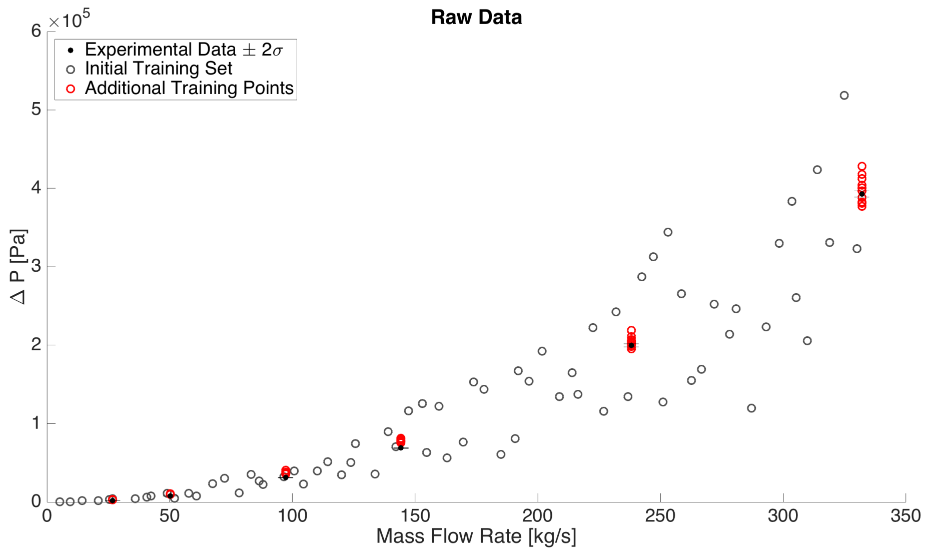

Figure 6.15: G&G additional training points with 1% error from SE and Linear covariance functions.

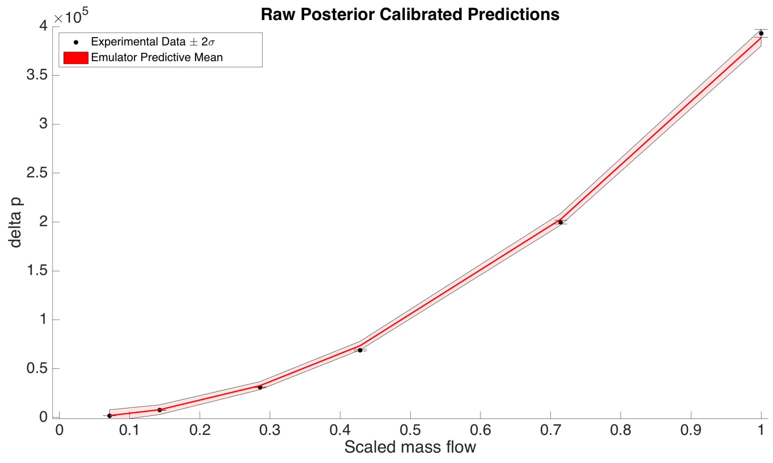

Figure 6.16: G&G emulator posterior predictions with 1% error from SE and Linear covariance functions.

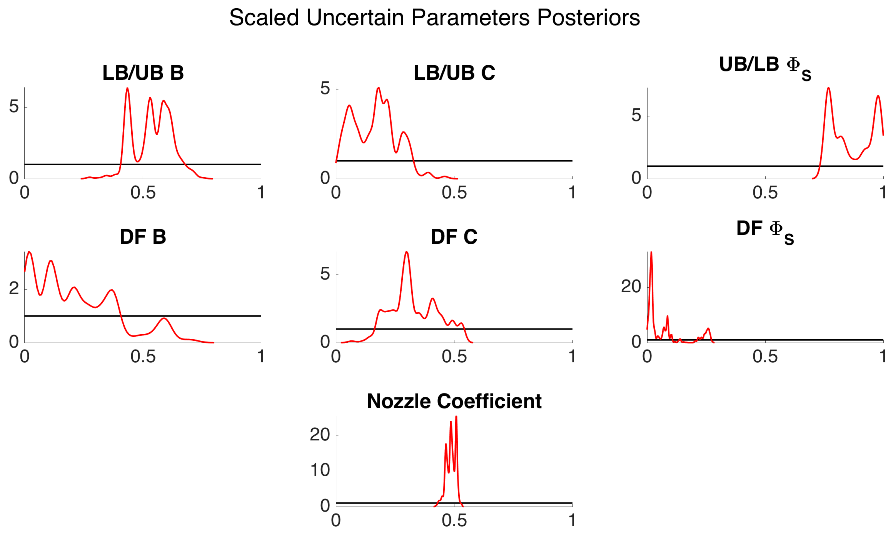

Figure 6.17: G&G emulator posterior distributions of the seven uncertain parameters with 1% error from SE and Linear covariance functions.

Chapter 7

Summary, Conclusions, and Future Work

7.1 Summary

Bayesian inference allows a framework for solving inverse problems that would otherwise be analytically intractable. Observational data can be used for uncertainty quantification (UQ) of computer model predictions and for inferring the distribution of the input parameters; this properly performs reverse UQ. The Gaussian Process framework allows an emulator to be constructed in place of the long running computer code in order to make Markov Chain Monte Carlo (MCMC) sampling of the posterior distributions feasible and computationally economical. Even a "fast" computer simulation taking 3 seconds per iteration would require over 83 hours of run time for 10^5 MCMC samples. An emulator that is 1000x faster would allow the same 10^5 MCMC samples to be completed in only 300 seconds. As the run time for a computer code increases, the value added by using an emulator increases as well since direct MCMC sampling would become impractical to even attempt.

The work reported here improves the performance of Gaussian Process Regression (GPR) emulators through active learning to choose which additional training points should be added to the current emulator training set. Further investigation into the role and implications of different covariance functions within the GPR framework are also investigated. The active learning methodology is demonstrated using a simple friction factor demonstration problem with a function that could easily be directly MCMC sampled from, in order to provide a benchmark for the emulator performance. The active learning methodology is then applied to two RELAP simulations to demonstrate the ability for the methodology to work with a long running computer code and higher complexity. For each of the three GPR active learning demonstrations the influence of the covariance function is also tested with several important observations and recommendations made.

7.2 Conclusions

The GPR active learning methodology was implemented and shown to produce nearly identical uncertain parameter distributions to the directly sampled MCMC case. Of the several covariance functions tested with the friction factor demonstration, the squared exponential (SE) covariance function and the product of the SE and Linear covariance functions were shown to perform best. This comparison was only possible because the friction factor example was simple enough for a direct MCMC sampling. For the more complex RELAP simulations, a direct MCMC sampling of the RELAP model was not feasible. Several recommendations for future applications of GPR are given with regards to the covariance function, including:

- The SE covariance function as the primary covariance function used in the literature for its simplicity also holds merit as a simple, yet effective covariance function for most applications. However, much forethought should be given to the covariance function of choice based upon the experimental data that is being modeled.
- In GPR applications where the experimental error is much smaller than the mean of the experimental data, a product of the SE and Linear covariance provides better results than the simple SE covariance function.
- In some instances, the experimental data may not be adequate to properly determine uncertain parameter posterior distributions, and simultaneous calibration from several different sets of experimental data for different uncertain parameters will produce better results.

Ultimately, the active learning methodology was shown to work well and produce better emulator performance with fewer training points. Insight was provided on which covariance functions should be chosen based on the experimental data that is being modeled.

7.3 Recommendations for Future Work

Optimization of Training Set

Usual GPR applications utilize data sets that containing thousands, if not tens of thousands of data points. In those situations, the goal is to reduce the number of training points by finding the best set that models the entire data because of the cost to invert the covariance matrix. The application of GPR in uncertainty quantification poses a similar, yet vastly different problem: find the minimal number of training points that still model the entire data without having an inventory of thousands of points to choose from. This becomes more important when each training point requires running a computer code which could take hours. While the naive approach of using 10 trainings point per uncertain parameter for the initial training set worked, the optimal minimal number of training points would be a great area of further research. This could provide a lower limit on the required number of training points and further reduce the number of times the computer code has to be run to determine the uncertain parameters, which is the ultimate goal anyway.

MCMC Mixing Rate and Optimization

For the GPR applications that have a small experimental error, often times the MCMC mixing was not the optimal 23% from [6]. This was a result of the experimental error limiting the combinations of uncertain parameter proposals that would be accepted by the MCMC sampling scheme. As a result, the MCMC scheme often sat at certain values for long periods of time, skewing the posterior distributions. The mixing rate for all GPR applications, regardless of experimental error or even the covariance function, is a problem for the entire MCMC community and not just for GPR applications.

MCMC Stopping Criterion

As mentioned before, MCMC sampling is more of an art-form than a science for many applications. Most of the time, expert knowledge and past experience with MCMC sampling allows a user to make informed decisions about how many iterations a simulation should use. However, this can lead to either reduced efficiency by computing too many iterations, or poor results by computing too few sampling iterations. Several stopping criterion, or convergence diagnostics, have been proposed in the literature. Convergence control techniques can be based upon one single output (single chain) or on outputs from several independent replications of the chain started from a preassigned initial distribution (parallel chains) [31].

Bibliography

[1] L. Biegler *et al.*, *Largescale Inverse Problems and Quantification of Uncertainty.* John Wiley & Sons, 2011.

[2] D. S. Sivia, *Data Analysis: A Bayesian Tutorial.* Clarendon: Oxford University Press, second ed., 2006.

[3] K. P. Murphy, *Machine Learning : A Probabilistic Perspective.* MIT Press, 2012.

[4] J. P. Yurko, *Uncertainty Quantification in Safety Codes Using a Bayesian Approach with Data from Separate and Integral Effect Tests.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2014.

[5] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, *Markov Chain Monte Carlo in Practice.* Chapman and Hall, 1996.

[6] G. O. Roberts and J. S. Rosenthal, "Optimal Scaling for Various Metropolis-Hastings Algorithms," *Statistical Science*, vol. 16, no. 4, pp. 351–67, 2011.

[7] H. Haario, E. Saksman, and J. Tamminen, "An Adaptive Metropolis Algorithm," *Bernoulli*, vol. 7, pp. 223–42, 1998.

[8] W. H. McAdams, *Heat Transmission.* New York: McGraw-Hill, third ed., 1954.

[9] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments.* Springer, 2003.

[10] G. E. P. Box and G. Jenkins, *Time Series Analysis: Forecasting and Control.* Holden-Day, 1976.

[11] *NIST/SEMATECH e-Handbook of Statistical Methods.* http://www.itl.nist.gov, March 2016.

[12] MATLAB, *version R2015b.* Natick, Massachusetts: The MathWorks Inc., 2016.

[13] C. E. Rasmussen and C. K. Williams, *Gaussian Process for Maching Learining.* MIT Press, 2005.

[14] R. M. Neal, *Bayesian Learning for Neural Networks.* Springer, 1996.

[15] M. D. Mckay, R. J. Beckman, and W. J. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code," *Technometrics*, vol. 21, no. 2, pp. 239–45, 1979.

[16] D. K. Duvenaud, *Automatic Model Construction with Gaussian Processes.* PhD thesis, University of Cambridge, 2014.

[17] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, tenth ed., 1972.

[18] M. L. Stein, *Interpolation of Spatial Data.* Springer, 1999.

[19] C. E. Rasmussen, "Gaussian Processes Covariance Functions and Classification," tech. rep., Max Planck Institute for Biological Cybernetics, 2006.

[20] A. Kapoor *et al.*, "Active Learning with Gaussian Processes for Object Categorization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[21] N. Lawrence, M. Seeger, and R. Herbrich, "Fast Sparse Gaussian Process Methods: The Informative Vector Machine," in *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, pp. 609–16, 2003.

[22] K. Kowalska and L. Peel, "Maritime Anomaly Detection Using Gaussian Process Active Learning," in *2012 15th International Conference on Informaiton Fusion (FUSION)*, (Singapore), 2012.

[23] A. Gorodetsky and Y. Marzouk, "Mercer Kernels and Integrated Variance Experimental Design: Connections Between Gaussian Process Regression and Polynomial Approximation." arXiv preprint arXiv:1503.00021, 2015.

[24] T. H. Cormen *et al.*, *Introduction to Algorithms.* MIT Press, third ed., 2009.

[25] C. D. Rasmussen and Z. Ghahramani, "Occam's Razor," *Advances in Neural Information Processing Systems*, pp. 294–300, 2001.

[26] G. Schwarz, "Estimating the Dimension of A Model," *The Annals of Statisticcs*, vol. 6, no. 2, pp. 461–64, 1978.

[27] N. Todreas and M. Kazimi, *Nuclear Systems Volume I: Thermal Hydraulic Fundamentals, Second Edition*, vol. 1. CRC Press, 2011.

[28] A. Gopalakrishnan and J. Gillette, “EBRFLOW — A Computer Program for Predicting the Coolant Flow Distribution in the Experimental Breeder Reactor-II,” *Nuclear Technology*, vol. 17, March 1973.

[29] S. Sadana *et al.*, “A Computer Controlled Precision High Pressure Measuring System,” *Measurement Science Review*, vol. 11, no. 6, pp. 198–202, 2011.

[30] Swagelok Company, *Industrial and Process Pressure Gauges*, March 2016.

[31] D. Chauveau and J. Diebolt, “An Automated Stopping Rule for MCMC Convergence Assessment.” 1998.