A METAMODELING APPROACH FOR APPROXIMATION OF MULTIVARIATE, STOCHASTIC AND DYNAMIC SIMULATIONS

A Thesis
Presented to
The Academic Faculty

by

Andres Felipe Hernandez Moreno

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Chemical & Biomolecular Engineering

Georgia Institute of Technology
May 2012

A METAMODELING APPROACH FOR APPROXIMATION OF MULTIVARIATE, STOCHASTIC AND DYNAMIC SIMULATIONS

Approved by:

Professor Martha A. Grover, Advisor
School of Chemical & Biomolecular Engineering
Georgia Institute of Technology

Professor Roshan J. Vengazhiyil
School of Industrial & Systems Engineering
Georgia Institute of Technology

Professor Matthew J. Realff
School of Chemical & Biomolecular Engineering
Georgia Institute of Technology

Professor Jeff S. Shamma
School of Electrical & Computer Engineering
Georgia Institute of Technology

Professor Clifford L. Henderson
School of Chemical & Biomolecular Engineering
Georgia Institute of Technology

Date Approved: 22 March 2012

"Porque un soñador no muere,

hasta que la ultima esperanza sea abatida"

Andres F. Hernandez

"Only in death will the true dreamer,

be separated from his imagination"

Translation by Dr. Karen Head. CETL, Georgia Tech.

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to God, for giving me the opportunity to walk this very long story of getting a Ph.D, despite being so far away from home, and finally accomplishing my dream. Then... I do not have enough words or ideas to express my gratitude to Martha. I could not have a better advisor; her patience, her charisma and thoughtful ideas, gave me strength during the darkest hours of this journey. I will always appreciate all the work you did, and I will follow the way you thought me about how to do science.

I would like to acknowledge all my committee members, Dr. Matthew Realff, Dr. Clifford Henderson, Dr. Roshan Vengazhiyil and Dr. Jeff Shamma, for all those significant comments and suggestions that make this work possible. Also, I would like to express my gratitude to Dr. Jye-Chyi Lu, Dr. Nicholas Hud and Dr. David Lynn from who I learned the value of a good research collaboration, and the value of thinking outside of the engineering box. On a really special note, I want to acknowledge the Center for the Enhancement of Teaching and Learning at Georgia Tech, and in particular its graduate communication coordinator, Dr. Karen Head, for making me improve every day in all aspect of my communication skills.

I would like to thanks all the members of the Grover research group, from the ones who came first (Cihan, Rentian, Paul, Aparna and Jonathan), to the ones who are right now (Huayu, Michael, Yuzhen, Dan, Christine, Ming-Chien and Xun). I will appreciate their friendship. I want to express all my gratitude and appreciation to Florencia Carrillo and her family for making me feel at home during all these years, to Pedro P. Franco and his family who have been witnesses of this triumph, and Alonso Herrera and his family for their unmeasurable friendship.

Last, I want to say thanks to my mom Mercedes, my aunt Yolanda and all my family back in Colombia who gave me their support during these years. I feel so proud for been in the family I am, and this is achievement for all of us.

This thesis work was made possible by the financial support of the Air Force Office of Scientific Research, the National Science Foundation and the NSF/NASA Center for Chemical Evolution.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis describes the implementation of metamodeling approaches as a solution to approximate multivariate, stochastic and dynamic simulations. In the area of statistics, metamodeling (or "model of a model") refers to the scenario where an empirical model is build based on simulated data. In this thesis, this idea is exploited by using pre-recorded dynamic simulations as a source of simulated dynamic data. Based on this simulated dynamic data, an empirical model is trained to map the dynamic evolution of the system from the current discrete time step $\mathbf{x}(s)$, to the next discrete time step $\mathbf{x}(s+1)$. Therefore, it is possible to approximate the dynamics of the complex dynamic simulation, by iteratively applying the trained empirical model. The rationale in creating such approximate dynamic representation is that the empirical models / metamodels are much more affordable to compute than the original dynamic simulation, while having an acceptable prediction error.

The successful implementation of metamodeling approaches, as approximations of complex dynamic simulations, requires understanding of the propagation of error during the iterative process. Prediction errors made by the empirical model at earlier times of the iterative process propagate into future predictions of the model. The propagation of error means that the trained empirical model will deviate from the expensive dynamic simulation because of its own errors. Based on this idea, Gaussian process model is chosen as the metamodeling approach for the approximation of expensive dynamic simulations in this thesis. This empirical model was selected not only for its flexibility and error estimation properties, but also because it can illustrate relevant issues to be considered if other metamodeling approaches were used for this purpose.

The implementation of Gaussian process models for approximating expensive dynamic simulations begins by understanding and exploring the effects of stochastic observations in this empirical model. It was found that Gaussian process models have a noise limit at which the model is still capable of identifying the local correlation in the residuals, from the stochastics observations. This noise level limit is defined as a signal-to-noise ratio of 1×10^1. If the signal-to-noise ratio in the observations is below this number, the identification of local correlation decreases. Also, a methodology is proposed and developed to characterize the error estimation properties of a Gaussian process model. This methodology is later used to quantify the propagation of error in dynamics and then expanded to include multivariate systems. Then, the Gaussian process model is implemented as an iterative mapping function to describe the dynamics of a one-dimensional state with stochastic observations. The results shows how it is possible to track the propagated error during the dynamic prediction by accounting for the input uncertainty in the traditional GPM prediction distribution.

Then, the dynamic implementation of Gaussian process models moves towards multidimensional, stochastic and dynamic systems. Using a well-studied dynamic system in chemical engineering, it was possible to use a multivariate Gaussian process model (cokriging) as an iterative mapping function for the prediction of multivariate dynamic systems. Also, it was shown that Gaussian process models are a good choice to describe the dynamics of systems with stable steady states, thanks to the data density around the steady states. Finally, Gaussian process modeling is implemented as an approximate model to describe the growth of platinum nanoparticles on a carbon nanotube surface. With this elaborate and complex dynamic system, other metamodeling approaches were explored for use can be used as approximate models. In conclusion, the implementation of metamodeling approaches for approximating expensive stochastic dynamic simulations is possible, but it may required of

specific statistical tools tailored for the identification of dynamic characteristics in the simulated dynamic data.

CHAPTER I

INTRODUCTION

1.1 Approximate models of expensive dynamic simulations

Mathematical descriptions of dynamic systems range in their complexity, from simple linear models to nonlinear expressions with a large number of parameters and assumptions. Fields in which expensive dynamic simulations are present include combustion, turbulent flow, and nanoscale phenomena. In many nanoscale phenomena, expensive dynamic simulations are used to characterized nanostructures at the atomic level and to understand the mechanical and chemical properties of these materials. For example, *ab initio* calculations can capture accurately the behavior of a complex chemical system at microscopic scales, but their information is often high-dimensional, stochastic, and sometimes impractical to be used at macroscopic length and time scales of physical interest [80]. As a result, most of the information generated by these expensive simulations is not suitable for tasks like process control or material structure design due to the long computational time. Simulating nanoscale phenomena for practical purposes requires the reduction in the computational effort, accounting for the errors in a less accurate simulation than the original detailed one.

To illustrate this modeling problem, consider the mathematical model of the deposition of platinum nanoparticles on carbon nanotubes in supercritical carbon dioxide [57], as an expensive dynamic simulation (a complete description of this nanoscale process can be found in Section 6.1). Although the nanoparticle dynamics model is fairly simple, it provides enough context to explain the necessity of approximate models for expensive dynamic simulations. The majority of the computational cost in

the nanoparticle dynamics model is due to the stochastic nature of the kinetic Monte Carlo (kMC) simulation used to simulate the growth of platinum nanoparticles. The kMC simulation uses the Gillespie algorithm [39] to simulate chemical reactions of individual atoms and/or molecules as probabilistic events that occur at known rates. The reaction events are associated with the mechanistic descriptions and hypotheses about how the growth of platinun nanoparticles is occurring, making the kMC simulation a valuable tool for simulating chemical systems at a microscale level. However, the computational cost of a kMC simulation increases when the number of reaction events and the number of atoms/molecules used in the simulation increases.

To explain the computational cost of the kMC simulation, let us define $\mathbf{z}(s) \in \mathbb{R}^m$ as a variable that contains all the information from the expensive dynamic simulation. For different chemical systems, $\mathbf{z}(s)$ could contain state variables, control variables and parameters from a high-order simulation. For the kMC simulation in the nanoparticle dynamic model, $\mathbf{z}(s)$ contains the concentrations of platinum nanoparticles at different sizes, and the concentrations of other chemical species on the carbon nanotube surface. Figure 1 is a graphical representation of the expensive kMC simulation. The figure shows how the kMC simulation can be understood as a discrete-time model using the Markov property. As part of the discrete representation of the kMC simulation, a constant sampling rate, Δt, is used to map from one discrete time to the next one.

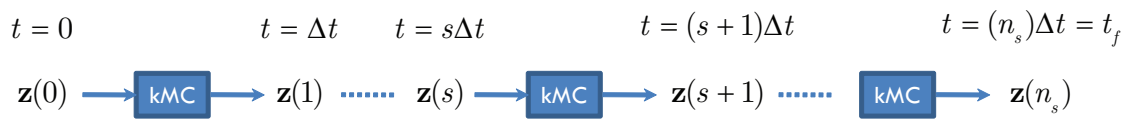


Figure 1: Representation of the nanoparticle dynamic model as an expensive dynamic simulation

Usually the computational cost in many dynamic simulations is associated with

the number of variables that the model has, and/or the short sampling rate Δt necessary to capture the dynamic features of a fast timescale process. In the case of the nanoparticle dynamics model, it is possible to decrease the computational cost of the simulation by decreasing the dimensionality of the $\mathbf{z}(s)$ variables from the kMC simulation. Figure 2 shows a simple scheme of the process of reduction and reconstruction of the information from the nanoparticle dynamic model. Let define the variable $\mathbf{x}(s) \in \mathbb{R}^d$ as a low-order version of the high-order variables $\mathbf{z}(s)$. It is clear from the discussion in this paragraph that $d \leq m$, representing the dimensional reduction from the high-order, expensive variables $\mathbf{z}(s)$ to the low-order, affordable variables $\mathbf{x}(s)$.

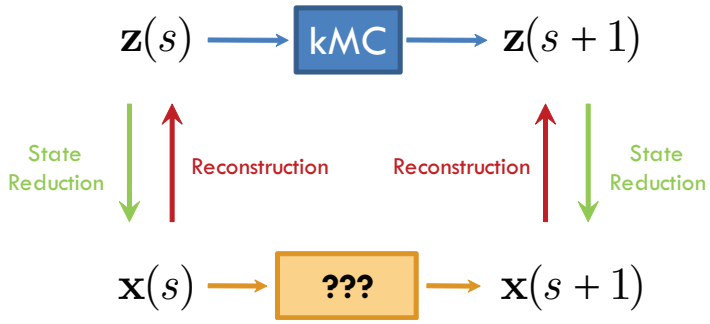


Figure 2: Approximate modeling for the nanoparticle dynamic model as an expensive dynamic simulation

The rationale in creating an approximate dynamic model is the possibility to represent the dynamics of an expensive simulation, by constructing mathematical models that are affordable to compute, and with an acceptable prediction error. In the case of the nanoparticle dynamics model, the improvement in the computational cost is conceived by combining the dimensional reduction of $\mathbf{z}(s)$ and, the construction of a mathematical model that maps the dynamic information of $\mathbf{x}(s)$ faster than the kMC simulation. However, a process systems engineer will still be interested in the nanoparticle size distribution, making necessary to recover the information in $\mathbf{x}(s)$, back to the original variables $\mathbf{z}(s)$. This is the reason to include a reconstruction step as part of the modeling scheme. For other cases of expensive dynamic simulations,

the reduction and reconstruction steps may not be necessary to decrease the computational cost. Perhaps the dimension of the expensive variables $\mathbf{z}(s)$ is small enough, such that an approximate dynamic model can be build directly with these variables, approximating the system dynamics faster than the original simulation.

The research focus of this thesis is in the construction of mathematical models for the low-order dynamics of the $\mathbf{x}(s)$ variables, using the nanoparticle dynamic model as the main case study. This thesis does not analyze the reduction / reconstruction process as part of the framework to reduce the computational cost of the nanoparticle dynamic model. There is an extensive body of literature regarding this subject including linear and nonlinear dimensionality reduction techniques, and some good reviews about the implementation of these techniques in the approximate modeling of expensive dynamic simulations [80, 106].

The approximate dynamic model of the low-order variables $\mathbf{x}(s)$ is built using the expensive dynamic simulation as a data generator, and then estimating the parameters of a mathematical model that will approximate the system dynamics. This approach for approximating expensive simulations has been explored previously in the area of statistics under the name of *metamodeling* [131] (the word comes from the idea of building a "model of a model"). The research developed in this thesis is different from other metamodeling approaches. The nature of the simulated data, is different since the nanoparticle dynamic model is a stochastic and dynamic simulation. In the case of the nanoparticle dynamic model, the approximate dynamic model is based on the dynamic data generated from the full kMC dynamic simulation after the reduction step. Although the results in this thesis are obtained using simulated dynamic data, they could be extended to cases where experimental dynamic data are available and where closed-form expressions of physical experiments may be needed.

1.2 Data-driven dynamic models

Empirical modeling refers to the creation of models that capture trends in measured data without explaining the mathematical or physical relationships that govern a system. Empirical models are employed to accomplish several process operation tasks like system identification, inferential modeling, process optimization and control. The main advantage of an empirical model is its simplified mathematical structure, connecting inputs and outputs of a system. This simplified structure is helpful to represent complex systems in many research fields. However, the drawback of these type of models is a required expertise from the modeler to choose the appropriate mathematical structure that represents the system accurately. In practice, most empirical models are formulated in an adaptive form to fit the information from the system. A few examples of empirical models are polynomial functions, neural networks, genetic algorithms and fuzzy models.

Iterative mapping is a particular application for empirical models for the dynamic description of a system. The methodology uses the formulation of a dynamic model written in a discrete-time formulation

$$\mathbf{x}(s+1) = \mathbf{f}\left(\mathbf{x}(s), \mathbf{u}(s)\right), \quad s = 0, 1, \ldots \tag{1}$$

with time $t = s\Delta t$. The user can build an empirical model based on the dynamic data collected from the system, and use the model it as an approximation of the true underlying model $\mathbf{f}$. The approximated dynamic model, $\hat{\mathbf{f}}$, represents the transition mapping from one discrete time to the next one, and its recursive iteration predicts dynamic trajectories of the state variables.

Traditional implementations of iterative mapping are based on the Markov assumption, which means that the dynamics of the state variables $\mathbf{x}(s)$ can be described using only information from the previous discrete time. However, iterative mapping can encompass other mathematical descriptions such as

- autoregressive moving average models, including several past inputs and outputs, e.g. $\mathbf{x}(s+1) = \mathbf{f}\left(\mathbf{x}(s), \mathbf{x}(s-1), \mathbf{u}(s), \mathbf{u}(s-1)\right)$
- time-dependent empirical models, e.g. $\mathbf{x}(s+1) = \mathbf{f}\left(\mathbf{x}(s), \mathbf{u}(s), s\right)$

An approximate dynamic model for expensive dynamic simulations can be built using the principles of iterative mapping. The expensive dynamic simulation acts as the "true" underlying model $\mathbf{f}$, generating simulated data of the system. Information from previous simulations is collected at different state variables $\mathbf{x}$ and control inputs $\mathbf{u}$. The stored information is used to generate an empirical model $\hat{\mathbf{f}}$ that predicts the outcome of a one-step-ahead prediction. Finally, the empirical model $\hat{\mathbf{f}}$ is implemented in an iterative mapping to approximate the dynamics of the expensive dynamic simulation. This is the reason why these approximate models are called *data-driven dynamic models*. This implementation of approximated models for expensive simulations has been studied previously in our group [106, 107, 108] for nanoscale phenomena in thin film deposition.

One example of iterative mapping is *in situ* adaptive tabulation (ISAT) proposed by Pope [114] for use in dynamic simulations of combustion, based on the concepts of *storage and retrieval*. The idea is to store function evaluations of the expensive combustion simulation in a database. The prediction of the mathematical model is made by searching in the database for the closest values at the required input state. Additional calculations in the methodology include the estimation of the gradient of the function and a region of validity, to estimate the local error in the prediction. Pope recognized the importance in the selection of the time step Δt to capture the changes in the state variables (which in his case were concentrations of several species in the combustion process) due to the differences in the timescales of the combustion reactions. The main challenge with ISAT is the intensive search in the table to find the appropriate values to make the prediction, and the large number of expensive function evaluations necessary to capture the noise in stochastic simulations. Some recent

publications related with ISAT involves studies in the error of the methodology [88], improvements in the searching on the tables [146], sensitivity analysis [2], dynamic optimization [145] and control [55].

A different approach to apply iterative mapping is to divide the state-space in "cells", and associate a constant value of $\mathbf{f}$ to generate the transition mapping. This approach, known as cell mapping [59], has been used in the analysis of dynamic systems [36] and optimal control [23, 166]. The main difficulty with this approach is how to split the state-space to represent most accurately the iterative mapping, particularly for high-dimensional systems. To address this problem, adaptive refinement techniques were proposed to improve the approximation of the cell mapping in regions where the long-term dynamics of the system take place [48]. Inspired by the idea of cell mapping, other variations of iterative mapping had been formulated, mixing techniques to subdivide the state-space with independent models on each subdomain. Two examples of this idea are a clustering algorithm like self-organizing map to subdivide the state-space for a simple cell mapping [14], and the usage of response surfaces for each subdomain to represent the iterative mapping [139].

In past years, more elaborate empirical models have been implemented to describe the dynamics of a complex system. Qi and coworkers [116] build a data-based spatiotemporal modeling approach based on the Karhunen-Loeve decomposition, combined with a least-squares support vector machine, one of the most recent and effective empirical models developed in the machine learning community. Another recent work implements a data-based multimodel approach that combines several partial least squares regression models, built locally in the state space, with a weighting function constructed from fuzzy c-means clustering [9]. Fuzzy models have been used recently on other settings such as a novel implementation of fuzzy cognitive maps and the Hebbian learning algorithm [89], and as an iterative clustering algorithm combined with a dynamic Takagi-Sugeno fuzzy model for the simulation of a heat exchanger

[52].

1.3 Error estimation in data-driven dynamic models

Using a data-driven dynamic model $\hat{\mathbf{f}}$ implies an approximation of the expensive dynamic simulation $\mathbf{f}$. Because there is a finite amount of information or data used in $\hat{\mathbf{f}}$, there will be a prediction error in the model relative to $\mathbf{f}$. In many of the previous examples of data-driven dynamic models, it is not clear how to estimate the error in the dynamic prediction. Most research on data-driven dynamic models focuses on implementing a large sequence of data analysis tools to capture the dynamics of a complex system. But few examples explore the relationship between the errors in each of the data analysis tools and the overall prediction error of the methodology. In fact, researchers have avoided the problem of estimating the prediction error in their models by using large amounts of data collected from their examples that guarantee a "good" representation of the complex system, or by using a validation set of data that evaluates the methodology performance. For most of the complex systems that are analyzed in chemical engineering, neither of these scenarios may be possible, due to the potential costs of carrying out a physical experiment on a lab bench, or in the case of this thesis, the high computational cost of the expensive dynamic simulation. Therefore, before implementing an approximate model for expensive dynamic simulations, it is necessary first to address the issue of error estimation in data-driven dynamic models.

The problem of error estimation in an empirical model is extremely important for iterative mapping. Prediction errors made by the empirical model in earlier times of the iterative mapping process propagate into future predictions of the data-driven dynamic model. The propagation of error means that the data-driven dynamic model will deviate from the expensive dynamic simulation because of its own errors. The

closest ideas regarding this subject are found in the mathematical analysis of numerical methods. While the major source of propagated error in a numerical method like Runge-Kutta is associated with the sampling rate Δt, in the case of iterative mapping the empirical model is the source of uncertainty. These research discussions are rarely found in the current body of literature on data-driven dynamic models.

Because of these questions regarding the estimation of error in empirical models, a probabilistic approach based on statistical models could be a better alternative to the construction of data-driven dynamic models [80]. Using an empirical model with a statistical framework also allows for a natural implementation of stochastic dynamic observations in the data-driven dynamic model. Among the different statistical models, Gaussian process models have been one of the most attractive empirical models for engineering applications. The reasons for the increasing interest in this particular model are:

1. Good prediction performance

2. High degree of customization and flexibility, due to the multiple correlation functions that can be used in the model

3. The model contains features of local and global estimation across the input space, which is a recurrent discussion in the implementation of data-driven models

4. It has convenient mathematical properties, thanks to the assumption of a Gaussian process, that allows this empirical model to be used either in a Bayesian or frequentist statistical framework

5. Based on the theory used to build these empirical models, Gaussian process models provide a theoretical quantification of their own prediction error: the prediction variance

Based on the problem of uncertainty propagation in data-driven dynamic models and the statistical properties of Gaussian process models, the research in this thesis builds on the following research question:

> *"Can a Gaussian process model estimate the propagated error generated during an iterative mapping process, based on its error estimation properties?"*

This research question is the driving force for the investigation presented in this thesis. The estimation of predicted errors are not only used to quantify the performance of an empirical model, but also as a metric of the uncertainty and/or lack of knowledge of the empirical model, relative to the true behavior of the system. Similarly as the error estimation properties of Gaussian process models are used to improve the performance of the model, the estimation of propagated error could be used to improve the prediction of a data-driven dynamic model. Furthermore, estimating the propagated error could be used to improve the understanding and exploration of the dynamics in a complex system, coupling it with systematic procedures found in sequential design of experiments. Finally, if the research question is demonstrated, it will represent that Gaussian process models have such statistical behavior, regardless of the system being modeled and/or approximated, a significant improvement that goes beyond the field of chemical engineering.

1.4 Research summary and scientific contributions

This thesis is a study about the implementation of Gaussian process models as data-driven dynamic models with the purpose of approximating an expensive stochastic dynamic simulation, such as the nanoparticle dynamic model. The previous sections in this introduction served to present the motivation about this study, to explain the reasons for selecting Gaussian process models and, to illustrate the potential impact of this work. The research question described in the Section 1.3 is not the goal of this

thesis, rather is a question that has emerged from it. In order to answer that question, this thesis starts from understanding the error estimation properties of Gaussian process models, and building up to the implementation of those properties in a multidimensional data-driven dynamic model for stochastic dynamic simulations. Each chapter has a scientific contribution that could stand separately from the main topic of the thesis, but also all chapters in the thesis contain practical recommendations to other users about how Gaussian process models should be implemented in iterative mapping.

Figure 3 shows the outline and organization between the different chapters in the thesis. The research in this thesis is divided in three concepts: using stochastic data to build a Gaussian process model, understanding Gaussian process models in dynamics and, implementing multidimensional Gaussian process models as a data-driven dynamic model from stochastic information. Chapter 3 presents a framework to analyze the error estimation of Gaussian process models, using the predicted error distribution across the input space. This chapter also evaluates the effect of using stochastic data to estimate the parameters of a Gaussian process model. Chapter 4 shows how to use Gaussian process models as data-driven dynamic models. Using a very simple chemical system, this chapter illustrates relevant issues of this implementation that are necessary to consider in more complex systems. Chapter 4 expands the error estimation framework developed in Chapter 3 for its application in dynamics, with the purpose of quantifying the propagated error by the Gaussian process models in an iterative mapping.

Chapter 5 shows a realistic scenario for the construction of data-driven dynamic models. This chapter uses the description and lessons learned in Chapter 4, to construct a multidimensional Gaussian process model for dynamic systems. Chapter 5 also expands the implementation of the error estimation framework for Gaussian process model to a stochastic, multidimensional and dynamic model. Finally, Chapter 6

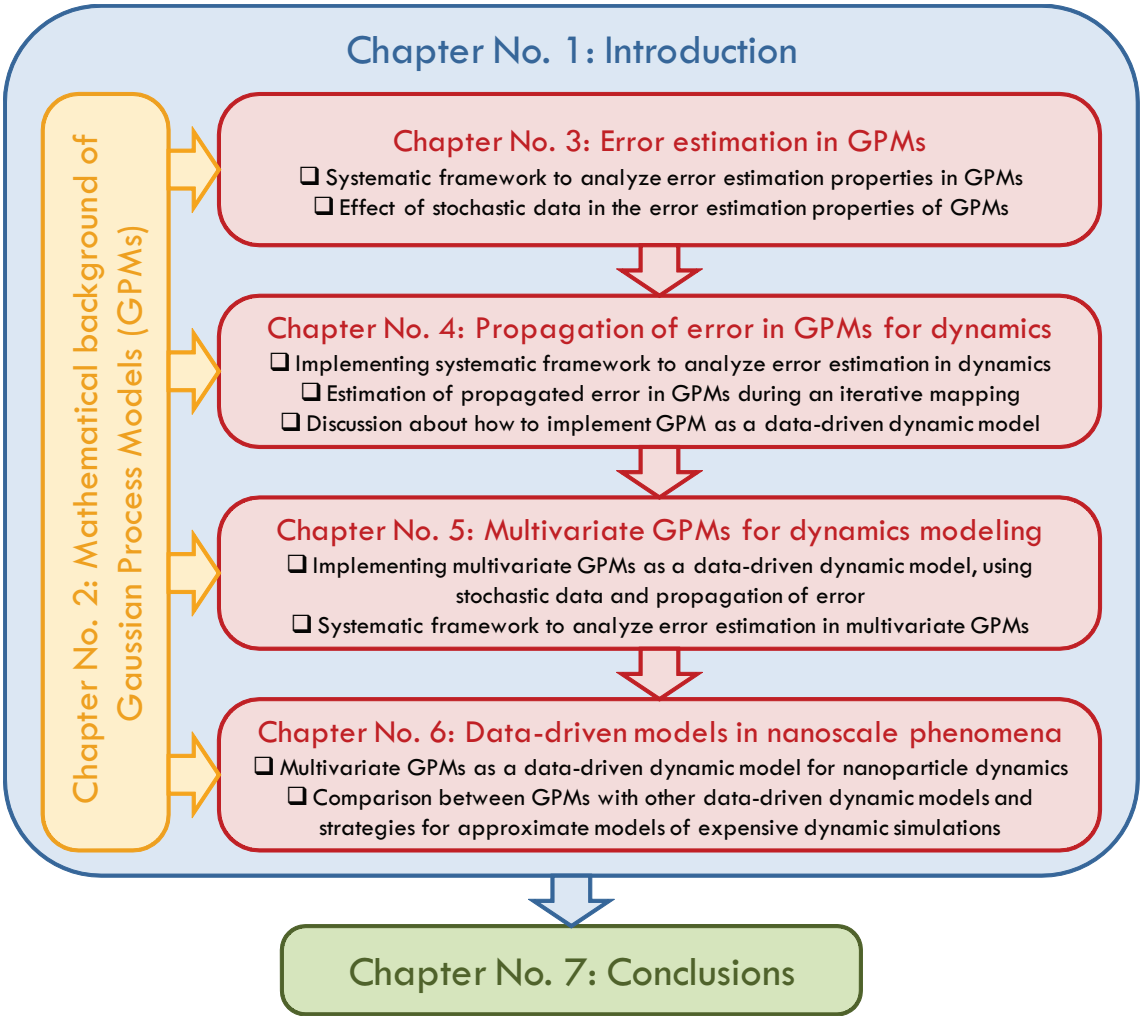


Figure 3: Summary of research work and scientific contributions in this thesis.

is an application of all the research concepts developed in the previous chapters, to the case of approximate models of expensive dynamic simulations. This application offers a careful comparison between several data-driven dynamic models, including Gaussian process models, support vector regression, regression-based inverse distance weighting, radial basis functions and equation-free approximations.

CHAPTER II

MATHEMATICAL BACKGROUND OF GAUSSIAN PROCESS MODELS

According to a definition in the book of Carl Edward Rasmussen [120], "A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution". Gaussian processes have been used extensively because of the mathematical properties that comes with the Gaussian distribution. The applications of Gaussian processes cover problems in classification [69], reinforcement learning [28], latent variable models [154] and state reduction [152]. This chapter will be focused on the mathematical description of Gaussian processes for a common problem in engineering: regression. This type of implementation of Gaussian processes is usually referred as Gaussian process regression, or just Gaussian process model (from now on referred to as GPM). This chapter begins by explaining the statistical and mathematical relationships in a GPM. Then, the chapter explores several methodologies for the estimation of the GPM parameters, focusing on the quantification of parameter uncertainty in the model. After completing the basic description of GPM, the chapter introduces several implementations of GPM in spatio-temporal processes, and the implementation of GPM as part of an iterative mapping procedure. Finally, the chapter concludes with a brief description of a multivariate GPM and its potential implementation for dynamic predictions.

Gaussian process model is a type of empirical model with a long history in the statistics community, beginning with the seminal work by Daniel Gerhardus Krige [81] and then expanded by French mathematician Georges Matheron in the 1960s [100]. These ideas gave birth to the research area of spatial statistics where the GPM

is known as *Kriging*. Krige's idea was that the difference between the prediction of a linear model and the true mean value of a function at some point can be related with the distance between the point of interest and a set of sampled points in the input space.

Figure 4 is a visualization of Krige's idea for deterministic observations. Figure 4a shows the regression made by a linear regression model, such as an ordinary least-squares error. In these regression models, a residual value remains between the prediction and the true value of the function at each point. This figure shows that it is more likely that nearby points in the input space will have similar residual values. If two points in the input space are far apart from each other, it is likely that their residual values will be different. This local behavior suggests that the magnitude of the residual at an unknown point could be related to the magnitude of the residuals at different reference points in the domain and, the distances between them.

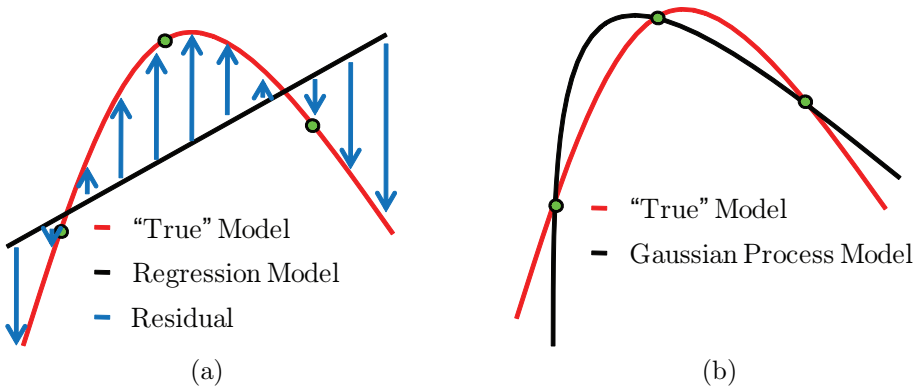


Figure 4: Graphical representation of Krige's idea for deterministic observations: Gaussian process model. (a) Linear regression model. (b) Gaussian process model.

In order to model this distance-based behavior, a stochastic Gaussian process can be added to the linear regression model, using the location of the sample points and their corresponding outcomes. The resulting model containg these two elements is known as a Gaussian process model. A GPM uses explicitly the location of the experimental points to construct the distance-based local correlation. This concept is

not usually employed in regression, in which the location of the experimental points is not explicitly considered once the model had been identified.

2.1 Deriving a Gaussian process model

Consider a set $\mathcal{D}$ of n input/output pairs $\{\mathbf{x}_i, y(\mathbf{x}_i)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$, $y(\mathbf{x}_i) \in \mathbb{R}$, $i = 1, \ldots, n$. This set of input/output pairs will be referred as sample points or experimental points. Consider also that the observed data $y(\mathbf{x}_i)$ contains an additive measurement noise $\eta \sim \mathcal{N}(0, \sigma_u^2)$ to the true response of the simulation $y_{tr}(\mathbf{x}_i) \in \mathbb{R}$

$$y(\mathbf{x}_i) = y_{tr}(\mathbf{x}_i) + \eta \tag{2}$$

Despite the presence of noise in the observed data, the objective is to predict the true response of the simulation $y_{tr}(\mathbf{x})$ at some other point $\mathbf{x}$.

In a GPM, the set of unknown true responses of the simulations are treated as random variables that are drawn from a joint Gaussian distribution [120]. For the elements in the set $\mathcal{D}$, the mean function $m(\mathbf{x})$ and the covariance matrix $K \in \mathbb{R}^{n \times n}, K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ of the true response values $y_{tr}(\mathbf{x}_i)$ are defined as

$$\mathbf{y}_{tr} \sim \mathcal{GP}(m, K) \tag{3}$$

$$\mathbb{E}[\mathbf{y}_{tr}] = m(\mathbf{x}) = H\boldsymbol{\beta} \tag{4}$$

$$\mathbb{E}\left[(\mathbf{y}_{tr} - H\boldsymbol{\beta})(\mathbf{y}_{tr} - H\boldsymbol{\beta})^T\right] = K \tag{5}$$

where $\mathbf{y}_{tr} = [y_{tr}(\mathbf{x}_1)\ y_{tr}(\mathbf{x}_2)\ \ldots\ y_{tr}(\mathbf{x}_n)]^T, \mathbf{y}_{tr} \in \mathbb{R}^n$, $H \in \mathbb{R}^{n \times p}$ represents a set of p regression or trend functions evaluated at the $\mathbf{x}_i$ inputs in $\mathcal{D}$ and, $\boldsymbol{\beta} \in \mathbb{R}^p$ are the p regression coefficients. The set of regression functions represent the linear model mentioned on Figure 4. There is no limitation in the selection of these regression functions as long as they preserve the linearity of the $\boldsymbol{\beta}$ coefficients.

Based on the description of the observed data in Equation (2), and the multivariate distribution of the true response, the output/observed information $y(\mathbf{x}_i)$ in the set

$\mathcal{D}$ can also be drawn from a multivariate Gaussian distribution as

$$\mathbf{y} \sim \mathcal{GP}\left(m, K + \sigma_u^2\, I\right) \tag{6}$$

$$\mathbb{E}\left[\mathbf{y}\right] = m\left(\mathbf{x}\right) = H\boldsymbol{\beta} \tag{7}$$

$$\mathbb{E}\left[\left(\mathbf{y} - H\boldsymbol{\beta}\right)\left(\mathbf{y} - H\boldsymbol{\beta}\right)^T\right] = K + \sigma_u^2\, I \tag{8}$$

where $\mathbf{y} = \left[y(\mathbf{x}_1)\ y(\mathbf{x}_2)\ \ldots\ y(\mathbf{x}_n)\right]^T, \mathbf{y} \in \mathbb{R}^n$ and $I \in \mathbb{R}^{n \times n}$ is the identity matrix.

The covariance matrix K can be constructed only after defining the function k that models the correlation between the true response values. Frequently, the correlation is described using the distance between sample points, with a monotonically decaying function. A common distance-based correlation function employed in the GPM literature is

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sigma_c^2\, \exp\left[-\frac{1}{2}\sum_{a=1}^{d}\frac{\left(x_{i,a} - x_{j,a}\right)^2}{\ell_a^2}\right] \tag{9}$$

where $\boldsymbol{\theta} = (\ell_1^2 \ldots \ell_d^2, \sigma_c^2, \sigma_u^2)$ are the parameters that control the features of the correlation in the GPM between samples in the set $\mathcal{D}$. Based on this correlation function, the GPM can also be understood as an empirical model that uses a positive-definite function to model the correlation between the residuals of a linear model in the $\boldsymbol{\beta}$ parameters and the true responses [24, 79].

The mathematical description of GPM as a regression model can be obtained in three different ways. Goldberger [43] obtained the GPM expressions by solving a constrained optimization problem for the best linear unbiased predictor and its mean square error. Other authors [70, 120] use a much simpler approach, taking advantage of the mathematical properties of the Gaussian process to derive the GPM expressions as conditional distributions to the joint Gaussian prior distribution in Equations (6)–(8). Lastly, a full Bayesian interpretation of GPM can be formulated, assuming that the coefficients of the linear model $\boldsymbol{\beta}$ are treated as random variables with some prior distribution π [26, 79]. This last Bayesian approach for GPMs yields identical GPM

expressions as the first two approaches when the prior distribution π is assumed to be non-informative. Since the first two approaches can be explained using the same mathematical variables, only these approaches will be described in more detail.

2.1.1 Best linear unbiased estimator

In the interest of estimating the true response at a new point $\mathbf{x} \in \mathbb{R}^d$, consider a linear predictor based on the stochastic observations of the set $\mathcal{D}$

$$\hat{y}(\mathbf{x}) = \boldsymbol{\lambda}^T(\mathbf{x})\mathbf{y} \tag{10}$$

where $\boldsymbol{\lambda}(\mathbf{x}) \in \mathbb{R}^n$ represents a vector of weights to be assigned to each output value in $\mathbf{y}$. To obtain the weight vector, one must minimize the mean square prediction error of the linear predictor

$$MSE\left[\hat{y}(\mathbf{x})\right] = \mathbb{E}\left[\left(y_{tr}(\mathbf{x}) - \boldsymbol{\lambda}^T(\mathbf{x})\mathbf{y}\right)^2\right] \tag{11}$$

subject to the unbiased prediction constraint

$$\mathbb{E}\left[y_{tr}(\mathbf{x}) - \boldsymbol{\lambda}^T(\mathbf{x})\mathbf{y}\right] = 0 \tag{12}$$

This constrained optimization problem is used to force the linear predictor, on average, to be close to the true function that it is estimating, eliminating any bias that the predictor could have. This constrained optimization problem is solve using the method of Lagrange multipliers, and its solution is known as the *best linear unbiased predictor (BLUP)* [43], which corresponds to:

$$\hat{y}\left(\mathbf{x}, \mathcal{D}\right) = \mathbf{h}^T\left(\mathbf{x}\right)\hat{\boldsymbol{\beta}} + \mathbf{k}^T\left(\mathbf{x}, \mathcal{D}\right)\left(K + \sigma_u^2\, I\right)^{-1}\left[\mathbf{y} - H\hat{\boldsymbol{\beta}}\right] \tag{13}$$

where $\mathbf{k}\left(\mathbf{x}, \mathcal{D}\right) \in \mathbb{R}^n$ is the correlation vector between $\mathbf{x}$ and each of the $\mathbf{x}_i$ samples in $\mathcal{D}$ using the correlation function in Equation (9); $\mathbf{h} \in \mathbb{R}^p$ represents a set of p regression functions evaluated at $\mathbf{x}$ and $\hat{\boldsymbol{\beta}}\left(\boldsymbol{\theta}\right) = \left(H^T\left(K + \sigma_u^2\, I\right)^{-1} H\right)^{-1} H^T\left(K + \sigma_u^2\, I\right)^{-1}\mathbf{y}$

is the generalized least-squares estimator of the regression coefficients. Similarly, the GPM prediction variance of the linear predictor is calculated as

$$\sigma_y^2(\mathbf{x},\mathcal{D}) = k(\mathbf{x},\mathbf{x}) - \begin{bmatrix}\mathbf{h}^T(\mathbf{x}) & \mathbf{k}^T(\mathbf{x},\mathcal{D})\end{bmatrix} \begin{bmatrix} 0 & H^T \\ H & K+\sigma_u^2 I \end{bmatrix}^{-1} \begin{bmatrix}\mathbf{h}(\mathbf{x}) \\ \mathbf{k}(\mathbf{x},\mathcal{D})\end{bmatrix} \tag{14}$$

where $k(\mathbf{x},\mathbf{x})$ is the evaluation of the correlation function between the unknown point and itself. It is also important to notice that in the original description of this constrained optimization [43], Goldberger did not specify a correlation function to describe the regression covariance matrix. Goldberger does not describe the nature of the observations to be either deterministic or stochastic. This means that either $\mathbf{y}$ or $\mathbf{y}_{tr}$ can be used in the GPM equations. A GPM can be written as a predictive distribution that depends on the observed data in the set $\mathcal{D} = \{\mathbf{x}_i, y(\mathbf{x}_i)\}$,

$$(y_{tr}|\mathbf{x},\mathcal{D}) \sim \mathcal{N}\left(\hat{y}(\mathbf{x},\mathcal{D}), \sigma_y^2(\mathbf{x},\mathcal{D})\right) \tag{15}$$

where the mean and variance are Equations (13) and (14) respectively. In summary, a GPM provides a predictive distribution of possible true response values, but in most of the GPM implementations, $\hat{y}(\mathbf{x},\mathcal{D})$ is used as the GPM mean prediction.

Gaussian process models have been used as empirical models of expensive deterministic simulations for optimization or fast function evaluation [98]. In those noise-free cases, there is no reason to estimate σ_u^2 as part of the GPM parameters and, as a result, GPM becomes an interpolator of the outcomes $y(\mathbf{x}_i)$ when GPM is evaluated at the sample points $\mathbf{x}_i$. In his seminal book [24], Cressie describes σ_u^2 a parameter that represents measurement error in noisy data, calling it the *nugget effect*. Recently, Ankenman et al. [6] thoroughly explain the role of σ_u^2 with their stochastic kriging metamodel and point the attention of the readers to the relationship between σ_u^2 and σ_c^2 in the GPM prediction.

2.1.2 From Gaussian process priors to conditional predictive distributions

This section presents the second approach to derive the GPM expressions in Equations (13) and (14). In the beginning of this chapter, the definition of a Gaussian process was presented. This definition considers all true response values of the simulation at any point in the input space $y_{tr}(\mathbf{x})$ to be in the same collection of random variables, which is modeled by Equations (3)–(5). Consider then the subset of the elements in the collection of random variables formed by all true response values at the sample points $\mathbf{y}_{tr}$ in the set $\mathcal{D}$, and the true response value at any other point in the input space $y_{tr}(\mathbf{x})$. By the marginalization property of the multivariate Gaussian distribution, the joint distribution of a subset of elements from the collection of random variables, modeled by Equations (3)–(5), will also be a multivariate Gaussian distribution with similar modeling representation. This result allows one to write the joint Gaussian distribution as

$$\begin{bmatrix} \mathbf{y}_{tr} \\ y_{tr}(\mathbf{x}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} H\boldsymbol{\beta} \\ \mathbf{h}^T(\mathbf{x})\,\boldsymbol{\beta} \end{bmatrix}, \begin{bmatrix} K & \mathbf{k}(\mathbf{x},\mathcal{D}) \\ \mathbf{k}^T(\mathbf{x},\mathcal{D}) & k(\mathbf{x},\mathbf{x}) \end{bmatrix} \right) \tag{16}$$

If the noise in the observed data is included as part of the joint Gaussian distribution, the multivariate Gaussian distribution becomes

$$\begin{bmatrix} \mathbf{y} \\ y_{tr}(\mathbf{x}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} H\boldsymbol{\beta} \\ \mathbf{h}^T(\mathbf{x})\,\boldsymbol{\beta} \end{bmatrix}, \begin{bmatrix} K + \sigma_u^2\, I & \mathbf{k}(\mathbf{x},\mathcal{D}) \\ \mathbf{k}^T(\mathbf{x},\mathcal{D}) & k(\mathbf{x},\mathbf{x}) \end{bmatrix} \right) \tag{17}$$

Equation (17) can be used to derive a posterior predictive distribution of $y_{tr}(\mathbf{x})$, by conditioning it to the joint Gaussian prior distribution of the observed data in the set $\mathcal{D}$ in Equation (15) [120]. By doing this, the predictive distribution is restricted to be in agreement with the stochastic observations. By creating the conditional distribution of y_{tr} given $\mathbf{y}$, the equations that describe the GPM predictive distribution, match the GPM expressions obtained by the constraint optimization procedure in Section 2.1.1.

2.2 Building a Gaussian process model

The implementation of a Gaussian process models requires three elements:

1. The selection of the sample points $\mathbf{x}_i$ in the set $\mathcal{D}$, along with its noisy observed data $\mathbf{y}$

2. The selection of a basis set of regression functions H

3. A correlation function k that models the local correlation in the Gaussian process

In statistics, the selection of sample points from simulated data to construct empirical models has been developed extensively under the name of design and analysis computer experiments (DACE) [21]. Over the years, the DACE community has developed a significant number of techniques and methodologies to improve the selection of input points for deterministic simulations [19, 25, 77, 103]. Since the early 1990s, Gaussian process modeling has been used in the DACE community [122] because its singular behavior as an interpolator when the measurement noise η is not included in the GPM predictive distribution. Most of the DACE methodologies can be classified as model-based or space-filling. The first class takes advantage of the structure of an empirical model, and builds a experimental design that satisfy a optimal criteria derived from the model. The space-filling class is focused on spreading the sample points in the input space to guarantee a good exploration of the region, regardless of the empirical model to be estimated. Perhaps the most popular and most used of the DACE methodologies is the Latin hypercube sampling (LHS)[101], because its simple implementation, good space-filling and good statistical properties.

The selection of a basis set of regression functions in GPM plays an important role, especially in the cases where the GPM is evaluated at points far away from the sample points in the set $\mathcal{D}$. In such cases, the GPM predictive distribution depends

only on the statistical characteristics of the selected regression functions. The simplest implementation of GPM assumes that the regression function is a constant and known value —usually this value is zero. This type of GPM implementation is known in statistics as *simple kriging*, where the predictive distribution relies only in the local correlation of the observed data. A more common GPM implementation assumes a constant, but unknown, regression function and it is referred with the name *ordinary kriging*. Last, a more complex and detailed implementation of a GPM involves a a basis set of regression functions, like polynomial basis functions, and it is known as *universal kriging*. There are no particular limitations in the selection of the basis functions that can be used in the GPM, since the GPM requires only the evaluation of the basis functions and not the particular mathematical structure of them. Joseph et al. [64] consider the selection of regression functions using a Bayesian variable selection technique for universal kriging.

The most unique element of a Gaussian process model is the correlation function k. The correlation function, also known as the *kernel* function, represents the local correlation features in the residual. Therefore, the performance and the mathematical properties of the Gaussian process model, like mean square continuity and differentiability, depend on the selection of this covariance function. In general, an arbitrary function of the input points $\mathbf{x}_i$ and $\mathbf{x}_j$ can be a valid covariance function, if the function is positive semidefinite, since it will guarantee the inversion of the correlation matrix K. The intention in this section is to illustrate some of the most common choices of covariance functions for practical applications, but the seminal reference that describes the theory behind the construction of valid covariance functions is the book written by Noel Cressie [24].

Equation (9) is known as the Gaussian or squared exponential covariance function, and it uses the distance between the input points $\mathbf{x}_i$ and $\mathbf{x}_j$ to describe the local correlation of the residuals. A *stationary* correlation function is a function that describes

the local correlation based only on the difference between $\mathbf{x}_i$ and $\mathbf{x}_j$. The stationary condition describes a covariance function that is independent of the location of $\mathbf{x}_i$ and $\mathbf{x}_j$, which means that, no matter where the points are located in the input or design space, the correlation between their residuals will remain the same, as long as the magnitude of the lag vector $\mathbf{x}_i - \mathbf{x}_j$ is the same. Other type of non-stationary covariance functions for GPM [110, 155] could include *dot product* correlation functions in the form of $k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_c^2 + \mathbf{x}_i \cdot \mathbf{x}_j$ or some linear terms that emphasize the local correlation of the GPM at specific locations in the input space. However, this adds additional parameters to the model.

A brief list of common stationary correlation functions are [24, 91, 120, 153]:

- Exponential correlation function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_c^2 \exp\left[-\sum_{a=1}^{d} \frac{|x_{i,a} - x_{j,a}|}{\ell_a}\right] \tag{18}$$

  $\boldsymbol{\theta} = (\sigma_c^2, \ell_a)$ where $\ell_a > 0$ and $a = 1, \ldots, d$

- Stable correlation function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_c^2 \exp\left[-\sum_{a=1}^{d} \left(\frac{|x_{i,a} - x_{j,a}|}{\ell_a}\right)^{\gamma_a}\right] \tag{19}$$

  $\boldsymbol{\theta} = (\sigma_c^2, \ell_a, \gamma_a)$ where $\ell_a > 0$, $0 < \gamma_a \leq 2$ and $a = 1, \ldots, d$

- Rational quadratic correlation function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_c^2 \sum_{a=1}^{d} \left(1 + \frac{(x_{i,a} - x_{j,a})^2}{2\alpha_a \ell_a^2}\right)^{\alpha_a} \tag{20}$$

  $\boldsymbol{\theta} = (\sigma_c^2, \ell_a, \alpha_a)$ where $\ell_a, \alpha_a > 0$ and $a = 1, \ldots, d$

The covariance function parameters ℓ_a are known as *characteristic length-scale* or *range parameters*, since they define the length scales over which residuals are correlated.

2.3 Parameter estimation methodologies for Gaussian process models

After collecting the information in the set $\mathcal{D}$, selecting a basis set of regression functions, and selecting a local correlation function k, the next step is the estimation of the parameters that define the GPM. The most common GPM parameter estimation approach is the maximum likelihood estimator (MLE). This methodology is an empirical Bayes approach to determine which are the parameter values that are most consistent with the observed data in the set $\mathcal{D}$, assuming that this parameter distribution behaves as a multivariate Gaussian distribution. According to this approach, the negative log-likelihood function of the MLE, which relates the GPM parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ given the observed data in the set $\mathcal{D}$, is

$$-\ln \mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{\beta} | \mathcal{D}\right) = \frac{n}{2}\ln\left(2\pi\right) + \frac{1}{2}\ln\left(\left|V\left(\boldsymbol{\theta}\right)\right|\right) + \frac{1}{2}\left(\mathbf{y} - H\boldsymbol{\beta}\right)^{T} V^{-1}\left(\boldsymbol{\theta}\right)\left(\mathbf{y} - H\boldsymbol{\beta}\right) \quad (21)$$

where $V \in \mathbb{R}^{n \times n}$ and $V = K + \sigma_u^2\, I$ is the regression covariance matrix of the Gaussian process model. To simplify the parameter estimation procedure in MLE, Equation (21) can be written as a function that depends only on $\boldsymbol{\theta}$. The estimated regression coefficients $\hat{\boldsymbol{\beta}}$ can be expressed as a function of $\boldsymbol{\theta}$ by making the first derivative of $-\ln \mathcal{L}$ respect to $\boldsymbol{\beta}$, be equal to zero. The result of this mathematical operation is known as the generalized least-squares estimator of the regression coefficients [79].

$$\begin{aligned} \frac{\partial\left(-\ln \mathcal{L}\right)}{\partial \boldsymbol{\beta}} &= -\left(\mathbf{y} - H\boldsymbol{\beta}\right)^{T} V^{-1} H = 0 \\ \hat{\boldsymbol{\beta}}\left(\boldsymbol{\theta}\right) &= \left(H^{T} V^{-1} H\right)^{-1} H^{T} V^{-1} \mathbf{y} \end{aligned} \quad (22)$$

Most of the time, the estimation of the GPM parameters is made with a nonlinear optimization that minimizes the expression in Equation (21), due to the mathematical complexity of obtaining analytical solutions for the $\boldsymbol{\theta}$ in the correlation function. The nonlinear nature of this optimization problem is found in the nonlinear Gaussian correlation function that is uses the GPM parameters $\boldsymbol{\theta}$ in the regression covariance

matrix V. As a result, the estimated GPM parameters $\hat{\boldsymbol{\theta}}$ by the MLE methodology are

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left[-\ln \mathcal{L}\left(\boldsymbol{\theta}\right) \right] \tag{23}$$

An alternative approach to the estimation of the GPM parameters is based on maximizing the marginal distribution of the correlation function parameters in the likelihood function, after averaging over all possible values of the $\boldsymbol{\beta}$ coefficients [72]. This method is called restricted maximum likelihood estimator (RMLE), it was first presented by Patterson and Thompson in the context of analysis of variance [113] and later Kitanidis [73] implemented it for the estimation of GPM. Several authors have compared multiple parameter estimation methodologies for GPM [111, 125, 165], including MLE and RMLE. The conclusion is that RMLE provides GPM estimated parameters with less bias compared with the estimated parameters from MLE, but at the cost of increasing their parameter uncertainty or variance. The major criticism regarding MLE or RMLE is the necessity to assume a particular distribution, in this case a multivariate Gaussian distribution. An alternative approach for the estimation of the GPM parameters is cross-validation [26, 98], which does not assume a specific distribution that relates the GPM parameters with the observed data. Nonetheless, MLE is the most used GPM parameter estimator for its straightforward implementation in any of the available nonlinear optimization tools.

2.3.1 Parameter uncertainty for Gaussian process models

The parameter uncertainty in an empirical model occurs because the finite amount of information collected in its training set. When the number of sample points in the training set increases, the identification of the parameters is more robust, decreasing the uncertainty around the estimated parameters. In statistics, the uncertainty in the estimated parameters is modeled by considering the estimated parameters as a random vector, created as a realization of the true parameter vector that characterize

the system. For this point of view, different sets of training data generate a realization of the true parameter vector that corresponds to their estimated parameters. Some authors have investigated the effect of parameter uncertainty in GPM [71, 74, 94, 96, 112], exploring the bias in the GPM estimated parameters and computational suggestions for GPM calculations. Some results in this area are summarized here.

Consider a Gaussian process model that uses the Gaussian correlation function in Equation (9) to describe its local correlation features. Assume that the distribution of random GPM parameter vectors $\boldsymbol{\theta}$ associated with this correlation function is a normal distribution defined as:

$$\boldsymbol{\theta} \sim \mathcal{N}\left(\mathbb{E}(\boldsymbol{\theta}), P\right) \tag{24}$$

where $\mathbb{E}(\boldsymbol{\theta}) \in \mathbb{R}^{d+2}$ is the expected value of this distribution and $P \in \mathbb{R}^{(d+2)\times(d+2)}$ represents the parameter covariance matrix of the GPM parameters. This GPM parameter distribution could be used to modify Equations (13) and (14), and account for the parameter uncertainty using a Taylor-series approximation centered at the input mean distribution $\mathbb{E}(\boldsymbol{\theta})$. For simplicity in the notation and in the following derivation, the expressions of the Gaussian process model are rearranged using the blockwise matrix inversion as follows. Define:

$$G \in \mathbb{R}^{(n+p)\times(n+p)} = \begin{bmatrix} 0 & H^T \\ H & V \end{bmatrix} \tag{25}$$

$$\mathbf{g} \in \mathbb{R}^{n+p} = \begin{bmatrix} \mathbf{h}\left(\mathbf{x}\right) \\ \mathbf{k}\left(\mathbf{x}, \mathcal{D}\right) \end{bmatrix} \tag{26}$$

$$\mathbf{y}_s \in \mathbb{R}^{n+p} = \begin{bmatrix} \mathbf{0} \\ \mathbf{y} \end{bmatrix} \tag{27}$$

Using G, $\mathbf{g}$ and $\mathbf{y}_s$, Equations (13) and (14) can be rewritten in a simplified version [138] of the GPM, emphasizing on the terms that depends on the GPM parameter

vector $\boldsymbol{\theta}$.

$$\hat{y}(\boldsymbol{\theta}) = \mathbf{y}_s^T G^{-1}(\boldsymbol{\theta})\,\mathbf{g}(\boldsymbol{\theta}) \tag{28}$$

$$\sigma_y^2(\boldsymbol{\theta}) = k(\mathbf{x},\boldsymbol{\theta}) - \mathbf{g}^T(\boldsymbol{\theta})\,G^{-1}(\boldsymbol{\theta})\,\mathbf{g}(\boldsymbol{\theta}) \tag{29}$$

Using a second order Taylor-series expansion of Equation (28), the GPM mean prediction can be approximated around the expected parameter vector $\mathbb{E}(\boldsymbol{\theta})$. This approximation is written as:

$$\begin{aligned}\hat{y}(\boldsymbol{\theta}) \approx{}& \hat{y}(\mathbb{E}(\boldsymbol{\theta})) + (\boldsymbol{\theta} - \mathbb{E}(\boldsymbol{\theta}))^T \left[\frac{\partial[\hat{y}(\mathbb{E}(\boldsymbol{\theta}))]}{\partial\boldsymbol{\theta}}\right] \\ &+ \frac{1}{2}\,(\boldsymbol{\theta} - \mathbb{E}(\boldsymbol{\theta}))^T \left[\frac{\partial^2[\hat{y}(\mathbb{E}(\boldsymbol{\theta}))]}{\partial\boldsymbol{\theta}^2}\right](\boldsymbol{\theta} - \mathbb{E}(\boldsymbol{\theta}))\end{aligned} \tag{30}$$

Applying the expectation operator in both sides of Equation (30), the expected GPM mean prediction $\mathbb{E}[\hat{y}(\boldsymbol{\theta})]$ due to parameter uncertainty is [138]:

$$\begin{aligned}\mathbb{E}[\hat{y}(\boldsymbol{\theta})] \approx{}& \mathbb{E}[\hat{y}(\mathbb{E}(\boldsymbol{\theta}))] + \cancelto{0}{(\mathbb{E}(\boldsymbol{\theta}) - \mathbb{E}(\boldsymbol{\theta}))^T} \left[\frac{\partial[\hat{y}(\mathbb{E}(\boldsymbol{\theta}))]}{\partial\boldsymbol{\theta}}\right] \\ &+ \frac{1}{2}\,\mathbb{E}\left[(\boldsymbol{\theta} - \mathbb{E}(\boldsymbol{\theta}))^T \left[\frac{\partial^2[\hat{y}(\mathbb{E}(\boldsymbol{\theta}))]}{\partial\boldsymbol{\theta}^2}\right](\boldsymbol{\theta} - \mathbb{E}(\boldsymbol{\theta}))\right] \\ \approx{}& \hat{y}(\mathbb{E}(\boldsymbol{\theta})) + \frac{1}{2}\,\mathrm{Tr}\left[P\,\frac{\partial^2[\hat{y}(\mathbb{E}(\boldsymbol{\theta}))]}{\partial\boldsymbol{\theta}^2}\right] \\ &+ \frac{1}{2}\,\cancelto{0}{(\mathbb{E}(\boldsymbol{\theta}) - \mathbb{E}(\boldsymbol{\theta}))^T} \left[\frac{\partial^2[\hat{y}(\mathbb{E}(\boldsymbol{\theta}))]}{\partial\boldsymbol{\theta}^2}\right] \cancelto{0}{(\mathbb{E}(\boldsymbol{\theta}) - \mathbb{E}(\boldsymbol{\theta}))} \\ \approx{}& \hat{y}(\mathbb{E}(\boldsymbol{\theta})) + \frac{1}{2}\,\mathrm{Tr}\left[P\,\frac{\partial^2[\hat{y}(\mathbb{E}(\boldsymbol{\theta}))]}{\partial\boldsymbol{\theta}^2}\right]\end{aligned} \tag{31}$$

Following a similar procedure as in Equation (31), the expected GPM prediction variance $\mathbb{E}\left[\sigma_y^2(\boldsymbol{\theta})\right]$ due to parameter uncertainty is [138]:

$$\mathbb{E}\left[\sigma_y^2(\boldsymbol{\theta})\right] \approx \sigma_y^2(\mathbb{E}(\boldsymbol{\theta})) + \frac{1}{2}\,\mathrm{Tr}\left[P\,\frac{\partial^2\left[\sigma_y^2(\mathbb{E}(\boldsymbol{\theta}))\right]}{\partial\boldsymbol{\theta}^2}\right] \tag{32}$$

It is important to notice that the mean parameter vector $\mathbb{E}(\boldsymbol{\theta})$ of the distribution is unknown. Therefore, this quantity is approximated by the estimated mean parameter vector $\hat{\boldsymbol{\theta}}$ calculated in Equation (23). When $\hat{\boldsymbol{\theta}}$ is used in Equations (31) and (32), the GPM parameter covariance matrix P will then be used to represent the uncertainty

in the maximum likelihood estimates. Usually the parameter covariance matrix P is approximated using the Cramer-Rao lower bound of unbiased estimators and the Fisher information matrix. The Fisher information matrix $\mathcal{I}(\boldsymbol{\theta}) \in \mathbb{R}^{(d+2)\times(d+2)}$ of the parameter set $\boldsymbol{\theta}$ is defined as

$$\mathcal{I}_{i,j}(\boldsymbol{\theta}) = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta_i \partial\theta_j} \ln \mathcal{L}(\boldsymbol{\theta}) \middle| \boldsymbol{\theta}\right], \quad i,j = 1,\ldots,d+2 \tag{33}$$

Applying the definition of $\mathcal{I}(\boldsymbol{\theta})$ to the likelihood function in Equation (21), the Fisher information matrix of $\hat{\boldsymbol{\theta}}$ [74] is:

$$\mathcal{I}_{i,j}(\hat{\boldsymbol{\theta}}) = \left[\frac{\partial \boldsymbol{\beta}}{\partial\theta_i}\right]^T H^T V^{-1} H \left[\frac{\partial \boldsymbol{\beta}}{\partial\theta_j}\right] + \frac{1}{2}\,\mathrm{Tr}\left[V^{-1}\frac{\partial V}{\partial\theta_i}V^{-1}\frac{\partial V}{\partial\theta_j}\right], \quad i,j = 1,\ldots,d+2 \tag{34}$$

where V is the regression covariance matrix of the GPM, which depends on $\hat{\boldsymbol{\theta}}$, and the derivative of the $\boldsymbol{\beta}$ regression functions with respect to each of the elements in the GPM parameter vector, evaluated at $\hat{\boldsymbol{\theta}}$,

$$\frac{\partial \boldsymbol{\beta}}{\partial\theta_i}(\hat{\boldsymbol{\theta}}) = -\left(H^T V^{-1} H\right)^{-1}\left[H^T V^{-1}\frac{\partial V}{\partial\theta_i}V^{-1}\left(\mathbf{y} - H\hat{\boldsymbol{\beta}}\right)\right], \quad i = 1,\ldots,d+2 \tag{35}$$

Using the Cramer-Rao inequality, the lower bound of the parameter covariance matrix P for an unbiased estimator is the inverse of the Fisher information matrix, therefore

$$P \geq \mathcal{I}^{-1}(\hat{\boldsymbol{\theta}}) \tag{36}$$

The implementation of Equation (36) to estimated the parameter uncertainty in GPM requires additional comments regarding its usage. The Cramer-Rao lower bound is based on the asymptotic behavior of the MLE as an unbiased estimator of its parameters. This asymptotic behavior means that when n, the number of sample points used in MLE, increases to infinity, the estimated GPM parameter $\hat{\boldsymbol{\theta}}$ is equal to the true GPM parameter vector $\mathbb{E}(\boldsymbol{\theta})$. Originally, the definition in Equation (36) is achieved asymptotically for MLE when independent and identically distributed observations are used [112]. Because of the local correlation features of Gaussian

process models, the asymptotic behavior of MLE for GPM parameters does not only depend on the number of sample points n, but also its distribution and location in the input space. There are two asymptotic behaviors of MLE for GPM: Increasing domain asymptotics refers to increase the number of sample points and the domain of the input space, in order to preserve the data density. Infill asymptotics refers to increase the number of sample points, while fixing the domain of the input space, therefore increasing the data density.

Mardia and Marshall [96] demonstrate the application of Equation (36) for GPM under increasing domain asymptotics. However, the majority of the GPM applications are in infill asymptotics where the input domain is a fixed area. Some results have been presented for GPM parameter uncertainty under fixed-area asymptotics [3, 159, 164] concluding that part of the GPM parameters have an asymptotic efficient behavior in their estimation, but this result is subject to the structure of the covariance function. Moreover, achieving asymptotic efficient behavior in the parameter estimates could be cumbersome because the growth in the GPM computational complexity when the number of input points in the model increases. Under these conditions, it could be more appropriate to use the biased version of the Cramer-Rao lower bound [33] to estimate P. Recent theory about biased Cramer-Rao lower bound is limited to linear regression models and has not yet been applied to nonparametric models like GPM. Despite these considerations, Equation (36) still been used as a rough estimate of the GPM parameter uncertainty. Although the research of this thesis does not involve any asymptotic behavior of the GPM parameters, it is necessary to describe most of these aspects for a complete picture of the GPM implementation.

The description of Equations (31) and (32) requires the second derivatives of the GPM mean prediction and GPM prediction variance with respect to the GPM parameter vector. The evaluation of Equations (31) and (32) employs the estimated mean parameter vector $\hat{\boldsymbol{\theta}}$ from the maximum likelihood estimator in Equation (23)

and its parameter covariance matrix P. Using matrix differential calculus Equations (28) and (29), the Jacobian vector and Hessian matrix of the GPM mean prediction $\hat{y}(\boldsymbol{\theta})$ and GPM prediction variance $\sigma_y^2(\boldsymbol{\theta})$ respect to each of the GPM parameters are:

$$\frac{\partial\left[\hat{y}\left(\boldsymbol{\theta}\right)\right]}{\partial\boldsymbol{\theta}} \in \mathbb{R}^{d+2}, \quad \frac{\partial^2\left[\hat{y}\left(\boldsymbol{\theta}\right)\right]}{\partial\boldsymbol{\theta}^2} \in \mathbb{R}^{(d+2)\times(d+2)}, \quad i,j = 1,\ldots,d+2$$

$$\frac{\partial\left[\hat{y}\left(\boldsymbol{\theta}\right)\right]}{\partial\theta_i} = -\mathbf{y}_s^T G^{-1}\frac{\partial G}{\partial\theta_i}G^{-1}\mathbf{g} + \mathbf{y}_s^T G^{-1}\frac{\partial\mathbf{g}}{\partial\theta_i} \tag{37}$$

$$\begin{aligned}\frac{\partial^2\left[\hat{y}\left(\boldsymbol{\theta}\right)\right]}{\partial\theta_i\partial\theta_j} &= 2\ \mathbf{y}_s^T G^{-1}\frac{\partial G}{\partial\theta_i}G^{-1}\frac{\partial G}{\partial\theta_j}G^{-1}\mathbf{g} - \mathbf{y}_s^T G^{-1}\frac{\partial^2 G}{\partial\theta_i\partial\theta_j}G^{-1}\mathbf{g} \\ &\quad - \mathbf{y}_s^T G^{-1}\frac{\partial G}{\partial\theta_j}G^{-1}\frac{\partial\mathbf{g}}{\partial\theta_i} - \mathbf{y}_s^T G^{-1}\frac{\partial G}{\partial\theta_i}G^{-1}\frac{\partial\mathbf{g}}{\partial\theta_j} + \mathbf{y}_s^T G^{-1}\frac{\partial^2\mathbf{g}}{\partial\theta_i\partial\theta_j}\end{aligned} \tag{38}$$

$$\frac{\partial\left[\sigma_y^2\left(\boldsymbol{\theta}\right)\right]}{\partial\boldsymbol{\theta}} \in \mathbb{R}^{d+2}, \quad \frac{\partial^2\left[\sigma_y^2\left(\boldsymbol{\theta}\right)\right]}{\partial\boldsymbol{\theta}^2} \in \mathbb{R}^{(d+2)\times(d+2)}, \quad i,j = 1,\ldots,d+2$$

$$\frac{\partial\left[\sigma_y^2\left(\boldsymbol{\theta}\right)\right]}{\partial\theta_i} = \frac{\partial\left[k\left(\mathbf{x},\mathbf{x}\right)\right]}{\partial\theta_i} - 2\ \mathbf{g}^T G^{-1}\frac{\partial\mathbf{g}}{\partial\theta_i} + \mathbf{g}^T G^{-1}\frac{\partial G}{\partial\theta_i}G^{-1}\mathbf{g} \tag{39}$$

$$\begin{aligned}\frac{\partial^2\left[\sigma_y^2\left(\boldsymbol{\theta}\right)\right]}{\partial\theta_i\partial\theta_j} &= \frac{\partial^2\left[k\left(\mathbf{x},\mathbf{x}\right)\right]}{\partial\theta_i\partial\theta_j} - 2\ \left[\frac{\partial\mathbf{g}}{\partial\theta_i}\right]^T G^{-1}\frac{\partial\mathbf{g}}{\partial\theta_j} - 2\ \mathbf{g}^T G^{-1}\frac{\partial^2\mathbf{g}}{\partial\theta_i\partial\theta_j} \\ &\quad + 2\ \mathbf{g}^T G^{-1}\frac{\partial G}{\partial\theta_i}G^{-1}\frac{\partial\mathbf{g}}{\partial\theta_j} + 2\ \mathbf{g}^T G^{-1}\frac{\partial G}{\partial\theta_j}G^{-1}\frac{\partial\mathbf{g}}{\partial\theta_i} \\ &\quad - 2\ \mathbf{g}^T G^{-1}\frac{\partial G}{\partial\theta_i}G^{-1}\frac{\partial G}{\partial\theta_j}G^{-1}\mathbf{g} + \mathbf{g}^T G^{-1}\frac{\partial^2 G}{\partial\theta_i\partial\theta_j}G^{-1}\mathbf{g}\end{aligned} \tag{40}$$

where, G, $\mathbf{g}$ and $\mathbf{y}_s$ are defined in Equations (26) and (27) and, for the case of the Gaussian correlation function

$$\frac{\partial\left[k\left(\mathbf{x},\mathbf{x}\right)\right]}{\partial\theta_i} = \begin{cases} 0 & \text{if } \theta_i = \ell_a, \quad a = 1,\ldots d \\ 1 & \text{if } \theta_i = \sigma_c^2 \\ 0 & \text{if } \theta_i = \sigma_u^2 \end{cases} \tag{41}$$

$$\frac{\partial^2\left[k\left(\mathbf{x},\mathbf{x}\right)\right]}{\partial\theta_i\partial\theta_j} = 0, \quad i,j = 1,\ldots,d+2 \tag{42}$$

To close the formulation of the parameter uncertainty in GPM using a Taylor-series approximation, it is necessary to obtain the first and second derivatives of the regression covariance matrix V with respect to each of the $\boldsymbol{\theta}$ GPM parameters in

the correlation function. These derivatives are necessary to evaluate the parameter covariance matrix P via the Fisher information matrix in Equation (34), and the derivatives of G and $\mathbf{g}$ with respect to the GPM parameters. In the end, the implementation of the parameter uncertainty for GPM comes down to how easy and mathematically convenient it is to calculate the derivatives of the parameters in the selected correlation function. For the case of the Gaussian correlation function in Equation (9), the first and second derivatives of the regression covariance matrix V are:

$$\frac{\partial V}{\partial \theta_a}, \frac{\partial^2 V}{\partial \theta_a \partial \theta_b} \in \mathbb{R}^{n \times n}, \quad \boldsymbol{\theta} = \left(\ell_1^2 \ldots \ell_d^2, \sigma_c^2, \sigma_u^2\right)$$

$$\frac{\partial V_{ij}}{\partial \ell_a} = k\left(\mathbf{x}_i, \mathbf{x}_j\right) \left[\frac{\left(x_{i,a} - x_{j,a}\right)^2}{\ell_a^3}\right] \tag{43}$$

$$\frac{\partial V_{ij}}{\partial \sigma_c^2} = \frac{1}{\sigma_c^2} k\left(\mathbf{x}_i, \mathbf{x}_j\right) \tag{44}$$

$$\frac{\partial V}{\partial \sigma_u^2} = I \tag{45}$$

$$\frac{\partial^2 V_{ij}}{\partial \ell_a \partial \ell_b} = \begin{cases} k\left(\mathbf{x}_i, \mathbf{x}_j\right) \left[\frac{\left(x_{i,a} - x_{j,a}\right)^2}{\ell_a^3}\right]^2 - 3\ k\left(\mathbf{x}_i, \mathbf{x}_j\right) \left[\frac{\left(x_{i,a} - x_{j,a}\right)^2}{\ell_a^4}\right] & \text{if } a = a \\ k\left(\mathbf{x}_i, \mathbf{x}_j\right) \left[\frac{\left(x_{i,a} - x_{j,a}\right)^2}{\ell_a^3}\right] \left[\frac{\left(x_{i,b} - x_{j,b}\right)^2}{\ell_b^3}\right] & \text{if } a \neq b \end{cases} \tag{46}$$

$$\frac{\partial^2 V_{ij}}{\partial \sigma_c^2 \partial \ell_a} = \frac{\partial V_{ij}}{\partial \sigma_c^2} \left[\frac{\left(x_{i,a} - x_{j,a}\right)^2}{\ell_a^3}\right] \tag{47}$$

$$\frac{\partial^2 V_{ij}}{\partial \sigma_u^2 \partial \ell_a} = \frac{\partial^2 V_{ij}}{\partial \left(\sigma_c^2\right)^2} = \frac{\partial^2 V_{ij}}{\partial \sigma_u^2 \partial \sigma_c^2} = \frac{\partial^2 V_{ij}}{\partial \left(\sigma_u^2\right)^2} = 0 \tag{48}$$

where $a, b = 1, \ldots, d$ and $i, j = 1, \ldots, n$.

2.4 Computational issues for Gaussian process models

Some of the benefits of using Gaussian process models include the ability to measure its prediction error based on the GPM prediction variance σ_y^2, the relatively small number of parameters, and its robustness when only a small number of sample points are used in the model [47]. All these benefits come at the cost of a large computational

load and potential problems of numerical stability, especially during the parameter estimation step. The evaluation of the negative log-likelihood function, Equation (21), requires the inversion of the regression covariance matrix V, which requires time $\mathcal{O}(n^3)$, and the evaluation of the derivatives of the function with respect to each of the GPM parameters in $\boldsymbol{\theta}$, which requires time $\mathcal{O}(n^2)$ per parameter [99, 120]. However, these computational issues are only relevant during the parameter estimation step and not during the GPM implementation itself. The inverse of the regression covariance matrix V needs to calculated once, and then it can be reused during the evaluation of the GPM mean prediction and GPM prediction variance. In addition to this aspect of the nonlinear optimization, it is also known that the likelihood function could have multiple local optimal values in its surface. However, in practice, this multiple local optima situation is not a problem that severely affects the prediction made by the GPM. These issues are handle by implementing a Cholesky decomposition of the regression covariance matrix V, which improves the computation load of the inverse calculation, and by incorporating a probabilistic global optimization algorithm like simulated annealing or by a stochastic search procedure that improves the selection of an initial guess in the deterministic optimization.

The problems of numerical stability in the regression covariance matrix V are more significant during the evaluation of the GPM expressions. A GPM considers the local correlation between the sample points in the set $\mathcal{D}$ by modeling the off-diagonal elements of the K matrix. This modeling procedure induces potential ill-conditioning problems in the GPM calculations, that are more severe when the number of sample points n increases. Several authors have described and analyzed the potential factors that affect the condition number of V [27, 105] in a GPM, which are:

- The structure of the sample points in the set $\mathcal{D}$ (i.e. number, location, distribution and density of the sample points in the input space)
- The correlation function k itself

The ill-conditioning problems in GPM due to the structure of the sample points are very significant in the area of design of computer experiments, since applications in this area targets modification in the data structure in order to satisfy certain prediction property of the GPM. Ababou et al. [1] showed that the Gaussian correlation function Equation (9), generates more numerical instabilities than the exponential correlation function, Equation (18), in an ordinary kriging model. Thus, the most commonly used correlation function in Gaussian process modeling is the one with the most numerical instability. Perhaps the reason why the research community still using the Gaussian correlation function is because its mathematical properties that make possible detailed mathematical analysis like the one just presented in Section 2.3.1. Despite all these problems, the presence in the GPM of the parameter σ_u^2 helps in the numerical stability of the GPM [97], since this parameter is added to the diagonal of the K correlation matrix.

2.5 Dynamic systems modeling and Gaussian process models

Let us summarize what it has been described so far in this chapter, in order to connect all the information with the creation of approximate models for expensive dynamic simulations. This chapter started with a mathematical description and derivation of a Gaussian process model, explaining each of the important aspects of this empirical model. This mathematical description has also explained why GPM is widely used as an approximate model for expensive deterministic simulations. Based on this mathematical description, it is clear that all the properties in the model are related to the correlation function k. Because of the significance of the correlation function, this chapter spent several sections presenting the methodologies to estimate the parameters in the correlation function, the quantification of parameter uncertainty to improve the model, and the computational issues regarding the correlation function.

All the previous sections are a summary of the implementation of GPM in metamodeling. Now, this chapter moves all this information to incorporate the concept of dynamics. The next sections of this chapter includes a brief description of traditional GPM implementations for spatio-temporal process in geostatistics, then it shows the implementation of GPM in the framework of iterative mapping, and concludes with a more complex mathematical formulation for multivariate Gaussian process models and dynamics.

Gaussian process modeling is presented as an empirical model that describes a function based on the spatial distribution of the information across the input space. In many practical cases, these functions have a temporal component, to describe the evolution over time. A few examples of spatio-temporal processes are motion capture [22], climate change and meteorology [53], environmental monitoring [95] and financial economics [38].

In geostatistics, spatio-temporal processes are modeled using two different approaches [83, 84]. In the reduction approach, the first step is an initial interpolation over the time dimension at each of the spatial locations, so all the information is available at the same time interval. Then, these approximated values are used as inputs of the GPM to make a spatial prediction. By repeating this procedure at different time intervals, several GPMs are created as layers or snapshots of the process. This approach is appropriate when the time period between measurements is large or when the recording process of the information is not synchronized. Problems with this approach include the amount of information to be stored at each time interval, and the extrapolation of the dynamics beyond the time frame of the recorded trajectories.

On the other hand, the extended approach considers time as an additional dimension in the input space, taking advantage of the multidimensional input formulation of the GPM. This approach implies that time is included in the covariance function k, considering time differences between sample points. The time difference

makes the mathematical modeling at different times easier for the model and also reduces the amount of information to be stored. The controversy that exists around this approach is due to the formulation of the covariance function that represents the local correlation of this variable. Several references that address this topic are [61, 63, 93, 115, 134].

Several authors have been using GPM in spatio-temporal processes using either of the two previous approaches [11, 40, 92]. Given that the extended approach includes the time dimension in the covariance function, time-series theory provides useful indications for a correlation function to handle this variable. A dynamic application of kriging can be seen as a multivariate time series where the information is correlated in time by an autocorrelation function [135]. To mention a few dynamic applications of kriging in geostatistics, Lindengergh et al. [87] proposed a kriging model based on the reduced approach to estimate concentrations of water vapor, and Jost et al. [66] formulate an extended approach for kriging to analyze the distribution of soil water storage and nitrate concentration, respectively. Both of these implementations consider time as a continuous variable that is incorporated in the correlation function k.

In the engineering and machine learning communities, Gaussian process models have been implemented to describe the dynamics of a system in a different way as in geostatistics. In these fields, a Gaussian process model is used as a black-box model that maps the states of the system, from a discrete time to the next discrete time, describing its dynamic behavior. This recursive usage of an empirical model is known as iterative mapping. Some of the recent applications where GPM is used in iterative mapping involves the identification and dynamic modeling of a bioreactor using GPM [10], dynamics of a hydraulic positioning system [47], dynamics for nanoparticle deposition [56], and more recently combinations with model predictive control [46], dynamic programming [28] and the temperature control of a building using multiple

past states [161]. All the previous dynamic implementations using GPM are limited to predictions of one state variable, except the case of the dynamics for nanoparticle deposition. This section contains a brief description of how a Gaussian process model can be implemented for dynamic systems using an iterative mapping.

2.5.1 Dynamic system identification using Gaussian process models

The GPM structure presented in the previous sections could be used to build an empirical model to describe the dynamics of a system, using either experimental or simulated data. A GPM can be used recursively to predict dynamic states, in an autoregressive mode, using current dynamic state information [10, 28, 78]. Assume that a dynamic system $\mathbf{f}$ can be represented as a discrete-time model, with $t = s\Delta t$, $s \in \mathbb{Z}, s \geq 0$ and

$$\mathbf{x}(s+1) = \mathbf{f}\left(\mathbf{x}(s)\right) \tag{49}$$

where $\mathbf{x}$ is a dynamic state and s is a discrete time index. An approximated version of this discrete-time model $\hat{\mathbf{f}}$ using GPM can be written as [56, 57]

$$\hat{\mathbf{x}}(s+1) = \hat{\mathbf{f}}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) \qquad x = 0, 1, ... \tag{50}$$

$$t = s\Delta t \tag{51}$$

$$\mathcal{D} = \left\{\mathbf{x}_i, y\left(\mathbf{x}_i\right)\right\}, y_i = \mathbf{x}(s+1) = \mathbf{f}\left(\mathbf{x}_i(s)\right) \tag{52}$$

where $\hat{\mathbf{f}}$ is the approximation of $\mathbf{f}$, and $\hat{\mathbf{x}}$ is the approximated dynamic state of $\mathbf{x}$ using Equation (13) as the state predictor in Equation (50). As an alternative to the usage of the classic GPM mean prediction in Equation (13), the GPM implementation using iterative mapping can also use Equation (31) to incorporate the parameter uncertainty correction in the predictions. Figure 5 represents a flowchart of this GPM implementation.

In the approximated dynamic representation using GPM, pre-computed evaluations of the function to be approximated, $\mathbf{f}\left(\mathbf{x}_i(s)\right)$, at different input values $\mathbf{x}_i(s)$ are

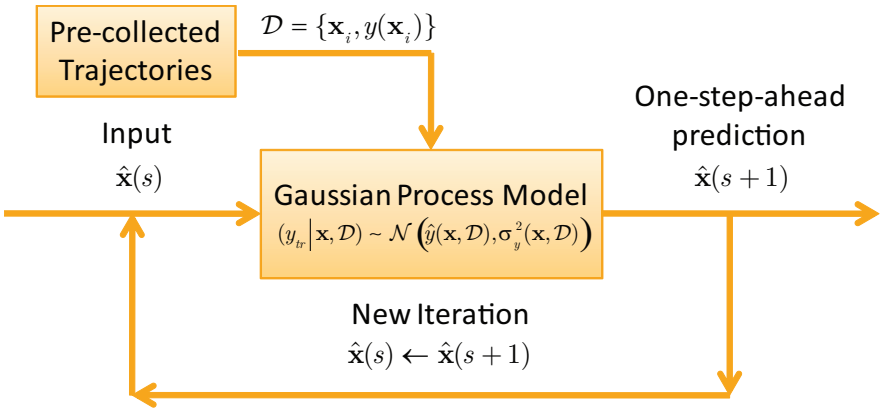


Figure 5: Flowchart of the recursive one-step-ahead prediction scheme using a GPM for dynamic systems modeling [57].

stored in $\mathcal{D}$ as a database for the system dynamics. The database is used as reference information for state prediction at untried locations in the state space by interpolating between the sampled points using GPM. In practice, a subset of information from the database may be extracted from the set $\mathcal{D}$ to approximate the dynamics at a new value of $\mathbf{x}$ with reduced computational demands. The dynamic representation could be extended to include control inputs $\mathbf{u}(s)$, as additional dimensions in the input space of the GPM.

The dynamic implementation of GPM has some resemblance to the equation-free prediction of system dynamics [67], in which evaluations of the original function $\mathbf{f}(\mathbf{x}_i(s))$ are made in the vicinity of $\mathbf{x}_i(s)$ and with small Δt, to approximate the dynamic evolution of the system. The difference between the two approaches is in the function evaluations from the expensive dynamic simulations. In the equation-free approach, the function evaluations are made "on line" while making the predictions, compared to the presented methodology where the evaluations are made off-line and used to map the states in time. A similar off-line implementation for expensive dynamic simulations has been used previously for dynamics of thin film deposition [108]. Compared to the previous example using expensive dynamic simulations, this thesis uses a nonlinear mathematical model to generate dynamic predictions compared to

the single cell mapping approach in Reference [108], in addition to the error estimation properties of the GPM.

To implement the iterative mapping in Equation (50) for multivariate systems, a GPM is constructed for each of the d state variables. Using Equations (13), (14) and (15), the distribution of the next state $\mathbf{x}(s+1)$ is assumed to be a multivariate Gaussian distribution with a state mean vector $\hat{\mathbf{x}}(s+1) \in \mathbb{R}^d$ and a state covariance matrix $S \in \mathbb{R}^{d \times d}$ as

$$(\mathbf{x}(s+1)|\mathbf{x}(s), \mathcal{D}) \sim \mathcal{N}\left(\hat{\mathbf{x}}(s+1), S\right) \tag{53}$$

$$\hat{x}_i(s+1) = \hat{y}_i(\hat{\mathbf{x}}(s), \mathcal{D}) \qquad i = 1, \ldots, d \tag{54}$$

$$S_{ij} = \begin{cases} \sigma^2_{y,i}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) & \text{if } i = j, \quad i, j = 1, \ldots d \\ \\ 0 & \text{if } i \neq j \end{cases} \tag{55}$$

where $\hat{x}_i(\hat{\mathbf{x}}(s), \mathcal{D})$ and $\sigma^2_{y,i}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right)$ are the GPM mean prediction and GPM prediction variance for the i^{th} state variable. Notice in Equation (55) that the off-diagonal terms of the state covariance matrix S are equal to zero. The reason for this consideration is because each of the d GPMs are estimated independently from the other GPMs used in the the prediction. Later on, in Section 2.6, this consideration will be relaxed by the implementation of a multivariate Gaussian process model, where several GPMs are estimated simultaneously. The main assumption in this GPM dynamic framework is that the system dynamics can be completely characterized using only the information from the previous time step (Markov property). This assumption implies that the dynamic behavior of the system dynamics is well characterized by the multiple GPMs, without the necessity of using states from previous time steps. Other implementations of this GPM dynamic framework do not satisfy the Markov property, relying the dynamic prediction in using information from several time steps in the past [161], but only predicting a single state variable.

2.5.2 Uncertainty in state prediction for Gaussian process models

Due to the recursive nature of the GPM dynamic model, error in the prediction will propagate from one step to the next. However, it is possible to calculate and propagate the uncertainty in the GPMs, and also to use it as a correction term in the state prediction. There are two different approaches to consider the state uncertainty in the recursive dynamic framework. The first approach is using a Taylor-series approximation, similar to the parameter uncertainty expressions in Section 2.3.1. The second approach is an analytical analysis of the state uncertainty in GPM presented by Girard and Murray-Smith [40]. The authors called this last approach as a *Gaussian* approximation to the propagation of uncertainty in the state prediction.

2.5.2.1 Taylor-series approximation

This section explain how to calculate the state propagation in the GPM using a Taylor-series approximation. By incorporating uncertainty propagation in the GPM dynamic framework, the GPM mean prediction *and* the GPM prediction variance from the previous time step are used to improve the prediction. The predicted state distribution in Equations (53)–(55) is:

$$\mathbf{x} \sim \mathcal{N}\left(\hat{\mathbf{x}}, S\right) \tag{56}$$

Similarly to the parameter uncertainty in Section 2.3.1, a Taylor-series expansion centered at the input mean distribution, $\hat{\mathbf{x}}$, is used to modify the GPM mean and prediction variance. Since changes in the training set $\mathcal{D}$ are not considered here, this symbol will no longer be explicitly stated. Using Equations (26) and (27), Equations (13) and (14) are rewritten, indicating the elements in the GPM that depend on the input point

$$\hat{y}\left(\mathbf{x}\right) = \mathbf{y}_s^T G^{-1} \mathbf{g}\left(\mathbf{x}\right) \tag{57}$$

$$\sigma_y^2\left(\mathbf{x}\right) = k\left(\mathbf{x}, \mathbf{x}\right) - \mathbf{g}^T\left(\mathbf{x}\right) G^{-1} \mathbf{g}\left(\mathbf{x}\right) \tag{58}$$

Following a similar procedure as in Equation (31), a second-order Taylor-series approximation on Equations (57) and (58) leads to the following expressions for the expected GPM mean prediction $\mathbb{E}\left[\hat{y}\left(\mathbf{x}\right)\right]$, and expected GPM variance $\mathbb{E}\left[\sigma_y^2\left(\mathbf{x}\right)\right]$

$$\mathbb{E}\left[\hat{y}\left(\mathbf{x}\right)\right] \approx \hat{y}\left(\hat{\mathbf{x}}\right) + \frac{1}{2}\ \mathrm{Tr}\left[S\left(\mathbf{x}\right)\ \frac{\partial^2\left[\hat{y}\left(\hat{\mathbf{x}}\right)\right]}{\partial \mathbf{x}^2}\right] \tag{59}$$

$$\mathbb{E}\left[\sigma_y^2\left(\mathbf{x}\right)\right] \approx \sigma_y^2\left(\hat{\mathbf{x}}\right) + \frac{1}{2}\ \mathrm{Tr}\left[S\left(\mathbf{x}\right)\ \frac{\partial^2\left[\sigma_y^2\left(\hat{\mathbf{x}}\right)\right]}{\partial \mathbf{x}^2}\right] \tag{60}$$

For the evaluation of Equations (59) and (60), the Jacobian vectors and Hessian matrices of the GPM equations with respect to each of the state variables are required.

$$\frac{\partial\left[\hat{y}\left(\mathbf{x}\right)\right]}{\partial \mathbf{x}} \in \mathbb{R}^d, \quad \frac{\partial^2\left[\hat{y}\left(\mathbf{x}\right)\right]}{\partial \mathbf{x}^2} \in \mathbb{R}^{d\times d}, \quad i,j = 1,\ldots,d$$

$$\frac{\partial\left[\hat{y}\left(\mathbf{x}\right)\right]}{\partial x_i} = \mathbf{y}_s^T G^{-1} \frac{\partial \mathbf{g}}{\partial x_i} \tag{61}$$

$$\frac{\partial^2\left[\hat{y}\left(\mathbf{x}\right)\right]}{\partial x_i \partial x_j} = \mathbf{y}_s^T G^{-1} \frac{\partial^2 \mathbf{g}}{\partial x_i \partial x_j} \tag{62}$$

$$\frac{\partial\left[\sigma_y^2\left(\mathbf{x}\right)\right]}{\partial \mathbf{x}} \in \mathbb{R}^d, \quad \frac{\partial^2\left[\sigma_y^2\left(\mathbf{x}\right)\right]}{\partial \mathbf{x}^2} \in \mathbb{R}^{d\times d}, \quad i,j = 1,\ldots,d$$

$$\frac{\partial\left[\sigma_y^2\left(\mathbf{x}\right)\right]}{\partial x_i} = \frac{\partial\left[k\left(\mathbf{x},\mathbf{x}\right)\right]}{\partial \mathbf{x}_i} - 2\ \mathbf{g}^T G^{-1} \frac{\partial \mathbf{g}}{\partial x_i} \tag{63}$$

$$\frac{\partial^2\left[\sigma_y^2\left(\mathbf{x}\right)\right]}{\partial x_i \partial x_j} = \frac{\partial^2\left[k\left(\mathbf{x},\mathbf{x}\right)\right]}{\partial \mathbf{x}_i \partial \mathbf{x}_j} - 2\ \left[\frac{\partial \mathbf{g}}{\partial x_i}\right]^T G^{-1} \frac{\partial \mathbf{g}}{\partial x_j} - 2\ \mathbf{g}^T G^{-1}\left[\frac{\partial^2 \mathbf{g}}{\partial x_i \partial x_j}\right] \tag{64}$$

The state uncertainty propagation in GPM depends on the structure in the vector $\mathbf{g}\left(\mathbf{x}\right)$. Since $\mathbf{g}(\mathbf{x})$ contains information about the correlation function and the regression functions in the GPM, an explicit evaluation of the first and second derivatives in each of the regression functions $\mathbf{h}(\mathbf{x})$ might be needed. Notice that when a constant regression function is used in the GPM, the state uncertainty propagation in the GPM dynamic implementation depends only on structure of the correlation function.

To close the description of the state uncertainty, the first and second derivatives of the vector $\mathbf{k}\left(\mathbf{x},\mathcal{D}\right)$ and $k\left(\mathbf{x},\mathbf{x}\right)$ with respect to the uncertain input state $\mathbf{x}$ are derived for the Gaussian correlation function in Equation (9).

$$\frac{\partial\left[\mathbf{k}\left(\mathbf{x},\mathcal{D}\right)\right]}{\partial x_i}, \frac{\partial^2\left[\mathbf{k}\left(\mathbf{x},\mathcal{D}\right)\right]}{\partial x_i \partial x_j} \in \mathbb{R}^n, \quad a = 1,\ldots,n \quad i,j = 1,\ldots,d$$

$$\frac{\partial\left[k_a\left(\mathbf{x},\mathbf{x}_a\right)\right]}{\partial x_i} = k\left(\mathbf{x},\mathbf{x}_a\right)\left[-\frac{\left(x_i - x_{a,i}\right)}{\ell_i^2}\right] \tag{65}$$

$$\frac{\partial^2\left[k_a\left(\mathbf{x},\mathbf{x}_a\right)\right]}{\partial x_i \partial x_j} = \begin{cases} k\left(\mathbf{x},\mathbf{x}_a\right)\left[-\frac{\left(x_i - x_{a,i}\right)}{\ell_i^2}\right]^2 + k\left(\mathbf{x},\mathbf{x}_a\right)\left[-\frac{1}{\ell_i^2}\right] & \text{if } i = j \\ k\left(\mathbf{x},\mathbf{x}_a\right)\left[-\frac{\left(x_i - x_{a,i}\right)}{\ell_i^2}\right]\left[-\frac{\left(x_j - x_{a,j}\right)}{\ell_j^2}\right] & \text{if } i \neq j \end{cases} \tag{66}$$

$$\frac{\partial\left[k(\mathbf{x},\mathbf{x})\right]}{\partial x_i} = \frac{\partial^2\left[k(\mathbf{x},\mathbf{x})\right]}{\partial x_i \partial x_j} = 0 \tag{67}$$

where $k_a\left(\mathbf{x},\mathbf{x}_a\right)$ represents the a^{th} element in the $\mathbf{k}\left(\mathbf{x},\mathcal{D}\right)$ vector and $\mathbf{x}_a$ is the a^{th} sample point in the set $\mathcal{D}$. Notice that for the Gaussian correlation function in Equation (9), $k\left(\mathbf{x},\mathbf{x}\right) = \sigma_c^2$, which explains the result in Equation (67).

2.5.2.2 Gaussian approximation

Girard and Murray-Smith [40] presented a careful analysis about how an uncertain input distribution like Equation (56) will alter the GPM predictive distribution in Equation (15). Theoretically, the GPM predictive distribution at an uncertain input distribution is equal to

$$\left(y_{tr}|\mathcal{D},\hat{\mathbf{x}},S\right) = \int_{-\infty}^{\infty}\left(y_{tr}|\mathcal{D},\mathbf{x}\right)\left(\mathbf{x}|\hat{\mathbf{x}},S\right)d\mathbf{x} \tag{68}$$

Due to the nonlinear relationship of the GPM predictive distribution on $\mathbf{x}$, Equation (68) cannot be solved analytically. The authors then approximate the solution of Equation (68) by assuming that the resulting GPM predictive distribution will remain as a Gaussian distribution. This is why the authors call this approach a *Gaussian approximation*. With this assumption in place, the authors approximate the solution of Equation (68) as

$$\left(y_{tr}|\mathcal{D},\hat{\mathbf{x}},S\right) \approx \mathcal{N}\left(\mu\left(\hat{\mathbf{x}},S\right),\nu\left(\hat{\mathbf{x}},S\right)\right) \tag{69}$$

where $\mu\left(\hat{\mathbf{x}},S\right)$ and $\nu\left(\hat{\mathbf{x}},S\right)$ represent the mean and variance of the approximated GPM predictive distribution of an uncertain input distribution. In their work, Girard and Murray-Smith derived μ and ν using the expected value of the GPM mean

prediction $\mathbb{E}\left[\hat{y}(\mathbf{x})\right]$, the expected value of the GPM prediction variance $\mathbb{E}\left[\sigma_y^2(\mathbf{x})\right]$, and the variance of the GPM mean prediction $\mathrm{Var}\left[\hat{y}(\mathbf{x})\right]$, respect to the uncertain input distribution in Equation (56). Except by the Gaussian approximation in the resulting GPM predictive distribution on Equation (68), every term in this approach was computed analytically.

Using the GPM description in Equations (57) to (58) and the law of iterated expectations and conditional variances, the mean $\mu\left(\hat{\mathbf{x}}, S\right)$ and variance $\nu\left(\hat{\mathbf{x}}, S\right)$ of the approximate GPM predictive distribution in Equation (69) are [40]:

$$\mu\left(\hat{\mathbf{x}}, S\right) = \boldsymbol{\alpha}^T \mathbb{E}[\mathbf{g}(\mathbf{x})] \tag{70}$$

$$\nu\left(\hat{\mathbf{x}}, S\right) = \mathbb{E}\left[k(\mathbf{x}, \mathbf{x})\right] - \mathrm{Tr}\left(\left(G^{-1} - \boldsymbol{\alpha}\boldsymbol{\alpha}^T\right)\mathbb{E}\left[\mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x})\right]\right) - \left(\mu\left(\hat{\mathbf{x}}, S\right)\right)^2 \tag{71}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{n+p}$, $\boldsymbol{\alpha} = G^{-1}\mathbf{y}_s$, $\mathbb{E}\left[\mathbf{g}(\mathbf{x})\right] \in \mathbb{R}^{n+p}$ is the vector that contains the expected value of each g_i entry in $\mathbf{g}$ and, $\mathbb{E}\left[\mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x})\right] \in \mathbb{R}^{(n+p)\times(n+p)}$ is the matrix that contains the expected value of each $g_i g_j$ product entry in $\mathbf{g}$.

Like in the case of the Taylor-series approximations, the final expressions in the Gaussian approximation depend on the mathematical structure of the elements in the vector $\mathbf{g}$, since they require the analytical solution of integrals of the form:

$$\int_{-\infty}^{\infty} k\left(\mathbf{x}, \mathbf{x}\right) p(\mathbf{x}) d\mathbf{x}$$

$$\int_{-\infty}^{\infty} g\left(\mathbf{x}, \mathbf{x}_i\right) p(\mathbf{x}) d\mathbf{x}$$

$$\int_{-\infty}^{\infty} g\left(\mathbf{x}, \mathbf{x}_i\right) g\left(\mathbf{x}, \mathbf{x}_j\right) p(\mathbf{x}) d\mathbf{x}$$

where $p(\mathbf{x})$ is the probability distribution of the uncertain input state, Equation (56). In the original paper [40], the authors assumed a GPM that uses $\mathbf{h}(\mathbf{x}) = 0$ as a regression function. This assumption allows them to not evaluate integrals in the form of

$$\int_{-\infty}^{\infty} h_i\left(\mathbf{x}\right) p(\mathbf{x}) d\mathbf{x} \quad i = 1, \ldots, p$$

$$\int_{-\infty}^{\infty} h_i(\mathbf{x})\, h_j(\mathbf{x})\, p(\mathbf{x}) d\mathbf{x} \quad i,j = 1,\ldots,p$$

$$\int_{-\infty}^{\infty} h_i(\mathbf{x})\, k(\mathbf{x},\mathbf{x}_j)\, p(\mathbf{x}) d\mathbf{x} \quad i = 1,\ldots,p \quad j = 1,\ldots,n$$

which could be difficult to solve, depending on the selected regression function. Fortunately, the expressions derived in this paper [40] can be easily implemented when a constant regression function $\mathbf{h}(\mathbf{x}) = 1$ is used in a GPM.

For a GPM that uses a Gaussian correlation function, Equation (9), and a constant regression function $\mathbf{h}(\mathbf{x}) = 1$, the expressions for $\mathbb{E}[k(\mathbf{x},\mathbf{x})]$, $\mathbb{E}[\mathbf{g}(\mathbf{x})]$ and $\mathbb{E}\left[\mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x})\right]$, with respect to probability density function $p(\mathbf{x})$ of the uncertain input distribution in Equation (56), are:

$$\mathbb{E}[k(\mathbf{x},\mathbf{x})] = \int_{-\infty}^{\infty} k(\mathbf{x},\mathbf{x})\, p(\mathbf{x}) d\mathbf{x} = \int_{-\infty}^{\infty} \sigma_c^2\, p(\mathbf{x}) d\mathbf{x} = \sigma_c^2 \tag{72}$$

$$\mathbb{E}[\mathbf{g}(\mathbf{x})] \in \mathbb{R}^{n+1}, \quad \mathbb{E}[\mathbf{g}(\mathbf{x})] = \begin{bmatrix} \mathbb{E}[\mathbf{h}(\mathbf{x})] \\ \mathbb{E}[\mathbf{k}(\mathbf{x},\mathcal{D})] \end{bmatrix}$$

$$\mathbb{E}[\mathbf{h}(\mathbf{x})] = \int_{-\infty}^{\infty} \mathbf{h}(\mathbf{x})\, p(\mathbf{x}) d\mathbf{x} = \int_{-\infty}^{\infty} p(\mathbf{x}) d\mathbf{x} = 1 \tag{73}$$

$$\mathbb{E}[\mathbf{k}(\mathbf{x},\mathcal{D})] \in \mathbb{R}^n, \quad i = 1,\ldots,n$$

$$\mathbb{E}[k(\mathbf{x},\mathbf{x}_i)] = \int_{-\infty}^{\infty} k(\mathbf{x},\mathbf{x}_i)\, p(\mathbf{x}) d\mathbf{x}$$

$$\mathbb{E}[k(\mathbf{x},\mathbf{x}_i)] = \sigma_c^2 \sqrt{\frac{|W|}{|W+S|}} \exp\left[-\frac{1}{2}(\hat{\mathbf{x}}-\mathbf{x}_i)^T (W+S)^{-1} (\hat{\mathbf{x}}-\mathbf{x}_i)\right] \tag{74}$$

$$\mathbb{E}\left[\mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x})\right] \in \mathbb{R}^{(n+1)\times(n+1)}, \quad \mathbb{E}\left[\mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x})\right] = \begin{bmatrix} \mathbb{E}\left[\mathbf{h}(\mathbf{x})\mathbf{h}^T(\mathbf{x})\right] & \mathbb{E}\left[\mathbf{h}(\mathbf{x})\mathbf{k}^T(\mathbf{x})\right] \\ \mathbb{E}\left[\mathbf{k}(\mathbf{x})\mathbf{h}^T(\mathbf{x})\right] & \mathbb{E}\left[\mathbf{k}(\mathbf{x})\mathbf{k}^T(\mathbf{x})\right] \end{bmatrix}$$

$$\mathbb{E}\left[\mathbf{h}(\mathbf{x})\mathbf{h}^T(\mathbf{x})\right] = \int_{-\infty}^{\infty} \mathbf{h}^2(\mathbf{x})\, p(\mathbf{x}) d\mathbf{x} = \int_{-\infty}^{\infty} p(\mathbf{x}) d\mathbf{x} = 1 \tag{75}$$

$$\mathbb{E}\left[\mathbf{h}(\mathbf{x})\mathbf{k}^T(\mathbf{x}, \mathcal{D})\right] = \mathbb{E}\left[\mathbf{k}^T(\mathbf{x}, \mathcal{D})\right] \tag{76}$$

$$\mathbb{E}\left[\mathbf{k}(\mathbf{x}, \mathcal{D})\mathbf{h}^T(\mathbf{x})\right] = \mathbb{E}\left[\mathbf{k}(\mathbf{x}, \mathcal{D})\right] \tag{77}$$

$$\mathbb{E}\left[\mathbf{k}(\mathbf{x}, \mathcal{D})\mathbf{k}^T(\mathbf{x}, \mathcal{D})\right] \in \mathbb{R}^{n \times n}, \quad i, j = 1, \ldots, n$$

$$\mathbb{E}\left[k(\mathbf{x}, \mathbf{x}_i)\, k(\mathbf{x}, \mathbf{x}_j)\right] = \int_{-\infty}^{\infty} k(\mathbf{x}, \mathbf{x}_i)\, k(\mathbf{x}, \mathbf{x}_j)\, p(\mathbf{x}) d\mathbf{x}$$

$$\mathbb{E}\left[k(\mathbf{x}, \mathbf{x}_i)\, k(\mathbf{x}, \mathbf{x}_j)\right] = \left(\sigma_c^2\right)^2 \sqrt{\frac{|W|}{|W + 2S|}} \exp\left[-\frac{1}{4}(\mathbf{x}_i - \mathbf{x}_j)^T W^{-1} (\mathbf{x}_i - \mathbf{x}_j)\right]$$
$$\exp\left[-\frac{1}{4}(2\hat{\mathbf{x}} - \mathbf{x}_i - \mathbf{x}_j)^T (W + 2S)^{-1} (2\hat{\mathbf{x}} - \mathbf{x}_i - \mathbf{x}_j)\right] \tag{78}$$

where $k(\mathbf{x}, \mathbf{x}_i)$ and $k(\mathbf{x}, \mathbf{x}_j)$ represent the i^{th} and j^{th} element in the $\mathbf{k}(\mathbf{x}, \mathcal{D})$, $\mathbf{x}_i$ and $\mathbf{x}_j$ are the i^{th} and j^{th} sample point in the set $\mathcal{D}$ and,

$$W \in \mathbb{R}^{d \times d}, \quad i, j = 1, \ldots, d \quad W_{ij} = \begin{cases} \ell_i^2 & \text{if } i = j \\ \\ 0 & \text{if } i \neq j \end{cases} \tag{79}$$

Finally, a Gaussian approximation approach could also be used to describe the effects of the parameter uncertainty in the GPM, using the parameter distribution in Equation (24). To the best of my knowledge, no evidence exists of this type of implementation to consider GPM parameter uncertainty.

2.5.3 Combining uncertainty sources for Gaussian process models

An advantage of using the Taylor-series approximation over the Gaussian approximation is the possibility to use simultaneously the results for the state and parameter uncertainty on the GPM prediction. After presenting the state and parameter uncertainty separately, the main results of these propagations can be combined in a single expression for the GPM mean prediction and GPM prediction variance. Given the normal distributions in Equations (24) and (56) and assuming that

$$\mathrm{Cov}(x_i, \theta_j) = 0, \quad i = 1, \ldots, d \quad j = 1, \ldots, d + 2 \tag{80}$$

the expected GPM prediction $\mathbb{E}\left[\hat{y}\left(\mathbf{x}, \boldsymbol{\theta}\right)\right]$, and expected GPM prediction variance $\mathbb{E}\left[\sigma_y^2\left(\mathbf{x}, \boldsymbol{\theta}\right)\right]$ are:

$$\mathbb{E}[\hat{y}(\mathbf{x}, \boldsymbol{\theta})] \approx \hat{y}(\hat{\mathbf{x}}, \hat{\boldsymbol{\theta}}) + \frac{1}{2}\ \mathrm{Tr}\left[S\ \frac{\partial^2[\hat{y}(\hat{\mathbf{x}}, \hat{\boldsymbol{\theta}})]}{\partial \mathbf{x}^2}\right] + \frac{1}{2}\ \mathrm{Tr}\left[P\ \frac{\partial^2[\hat{y}(\hat{\mathbf{x}}, \hat{\boldsymbol{\theta}})]}{\partial \boldsymbol{\theta}^2}\right] \tag{81}$$

$$\mathbb{E}[\sigma_y^2(\mathbf{x}, \boldsymbol{\theta})] \approx \sigma_y^2(\hat{\mathbf{x}}, \hat{\boldsymbol{\theta}}) + \frac{1}{2}\ \mathrm{Tr}\left[S\ \frac{\partial^2[\sigma_y^2(\hat{\mathbf{x}}, \hat{\boldsymbol{\theta}})]}{\partial \mathbf{x}^2}\right] + \frac{1}{2}\ \mathrm{Tr}\left[P\ \frac{\partial^2[\sigma_y^2(\hat{\mathbf{x}}, \hat{\boldsymbol{\theta}})]}{\partial \boldsymbol{\theta}^2}\right] \tag{82}$$

Equations (81) and (82) are the summary of the application of uncertainty propagation for Gaussian process model using Taylor-series approximations. The only GPM uncertainty aspect that has not been explored is the model uncertainty, which is a model selection problem on how to choose correlation and regression functions. The assumption in Equation (80) is that the GPM parameters $\boldsymbol{\theta}$ and the uncertain input point $\mathbf{x}$ are completely independent, therefore you can divide the application of the Taylor-series approximations in both random vectors without calculating the cross-second partial derivatives.

2.6 Multivariate Gaussian process models

The original definition of a Gaussian process models uses a multidimensional input point $\mathbf{x} \in \mathbb{R}^d$ to make predictions of a scalar value $y\left(\mathbf{x}\right) \in \mathbb{R}$. Because of this, the presented GPM dynamic implementation relies in the creation of d GPMs for each of the state variables to be predicted. This is an undesirable dynamic formulation for system identification, since the state covariance matrix S is not completely specified (remember the assumption in Equation (55) with the off-diagonal terms in S). Statisticians have expanded the mathematical theory of Gaussian process models for the prediction of multiple outputs simultaneously. Here is a brief description of this approach and some comments about its implementation for dynamic systems modeling.

Consider a set $\mathcal{D}$ of n input/output pairs $\{\mathbf{x}_i, \mathbf{y}\left(\mathbf{x}_i\right)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{y}\left(\mathbf{x}_i\right) \in \mathbb{R}^m$, $i = 1 \ldots n$. For simplicity in the description presented here, assume that $m = 2$.

According to the description of a Gaussian process in Section 2.1 and the GPM dynamic framework in Section 2.5, each of the two output dimensions in the set $\mathcal{D}$ can be considered independently as:

$$\mathbf{y}_1 \sim \mathcal{GP}\left(m_1, K_{11} + \sigma_{u,1}^2\, I\right) \tag{83}$$

$$\mathbb{E}\left[\mathbf{y}_1\right] = m_1\left(\mathbf{x}\right) = H_1\boldsymbol{\beta}_1 \tag{84}$$

$$\mathbb{E}\left[\left(\mathbf{y}_1 - H_1\boldsymbol{\beta}_1\right)\left(\mathbf{y}_1 - H_1\boldsymbol{\beta}_1\right)^T\right] = K_{11} + \sigma_{u,1}^2\, I \tag{85}$$

$$\mathbf{y}_2 \sim \mathcal{GP}\left(m_2, K_{22} + \sigma_{u,2}^2\, I\right) \tag{86}$$

$$\mathbb{E}\left[\mathbf{y}_2\right] = m_2\left(\mathbf{x}\right) = H_2\boldsymbol{\beta}_2 \tag{87}$$

$$\mathbb{E}\left[\left(\mathbf{y}_2 - H_2\boldsymbol{\beta}_2\right)\left(\mathbf{y}_2 - H_2\boldsymbol{\beta}_2\right)^T\right] = K_{22} + \sigma_{u,2}^2\, I \tag{88}$$

with their corresponding regression covariance matrices, $V_{11} = K_{11} + \sigma_{u,1}^2\, I$ and $V_{22} = K_{22} + \sigma_{u,2}^2\, I$.

A multivariate GPM enables the correlation of information among the multiple outputs in the model. This means that a multivariate GPM not only correlates the residuals of one variable based on its location, but also correlates the residuals of one predicted variable with the residuals of another predicted variable. In order to model the spatial correlation of residuals from two different predicted variables, statisticians used a *cross-covariance* function. The cross-covariance function $k_{ij}\left(\mathbf{x}_a, \mathbf{x}_b\right)$ describes the spatial correlation between the residuals of the predicted variables y_i and y_j, at the locations $\mathbf{x}_a$ and $\mathbf{x}_b$. By using the cross-correlation function, the cross-correlation matrix $K_{ij} \in \mathbb{R}^{n\times n}$ and the regression cross-covariance matrix $V_{ij} \in \mathbb{R}^{n\times n}$ are also defined. With these new variables, the multivariate GPM can be obtained using a similar constrained nonlinear optimization that was used to create the best linear unbiased predictor in Section 2.1.1 [150]. For the simplified case of only two predicted variables, the multivariate GPM predictive distribution is:

$$\left[\mathbf{y}_{tr}|\mathbf{x}, \mathcal{D}\right] \sim \mathcal{N}\left(\hat{\mathbf{y}}\left(\mathbf{x}, \mathcal{D}\right), \Sigma_y^2\left(\mathbf{x}, \mathcal{D}\right)\right) \tag{89}$$

$$\hat{\mathbf{y}}\left(\mathbf{x},\mathcal{D}\right)=\mathbf{h}^{T}\left(\mathbf{x}\right)\hat{\boldsymbol{\beta}}+\mathbf{k}^{T}\left(\mathbf{x},\mathcal{D}\right)V^{-1}\left[\mathbf{y}-H\hat{\boldsymbol{\beta}}\right] \tag{90}$$

$$\Sigma_{y}^{2}\left(\mathbf{x},\mathcal{D}\right)=K_{0}\left(\mathbf{x},\mathbf{x}\right)-\begin{bmatrix}\mathbf{h}^{T}\left(\mathbf{x}\right) & \mathbf{k}^{T}\left(\mathbf{x},\mathcal{D}\right)\end{bmatrix}\begin{bmatrix}0 & H^{T}\\ H & V\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{h}\left(\mathbf{x}\right)\\ \mathbf{k}\left(\mathbf{x},\mathcal{D}\right)\end{bmatrix} \tag{91}$$

where

$$\mathbf{y}\in\mathbb{R}^{2n}=\begin{bmatrix}\mathbf{y}_{1}\\ \mathbf{y}_{2}\end{bmatrix} \tag{92}$$

$$H\in\mathbb{R}^{2n\times 2p}=\begin{bmatrix}H_{1} & 0\\ 0 & H_{2}\end{bmatrix} \tag{93}$$

$$\boldsymbol{\beta}\in\mathbb{R}^{2p}=\begin{bmatrix}\boldsymbol{\beta}_{1}\\ \boldsymbol{\beta}_{2}\end{bmatrix} \tag{94}$$

$$\mathbf{h}(\mathbf{x})\in\mathbb{R}^{2p\times 2}=\begin{bmatrix}\mathbf{h}_{1}(\mathbf{x}) & 0\\ 0 & \mathbf{h}_{2}(\mathbf{x})\end{bmatrix} \tag{95}$$

$$\mathbf{k}\left(\mathbf{x},\mathcal{D}\right)\in\mathbb{R}^{2n\times 2}=\begin{bmatrix}\mathbf{k}_{11}\left(\mathbf{x},\mathcal{D}\right) & \mathbf{k}_{12}\left(\mathbf{x},\mathcal{D}\right)\\ \mathbf{k}_{21}\left(\mathbf{x},\mathcal{D}\right) & \mathbf{k}_{22}\left(\mathbf{x},\mathcal{D}\right)\end{bmatrix} \tag{96}$$

$$K_{0}\left(\mathbf{x},\mathbf{x}\right)\in\mathbb{R}^{2\times 2}=\begin{bmatrix}\mathbf{k}_{11}\left(\mathbf{x},\mathbf{x}\right) & \mathbf{k}_{12}\left(\mathbf{x},\mathbf{x}\right)\\ \mathbf{k}_{21}\left(\mathbf{x},\mathbf{x}\right) & \mathbf{k}_{22}\left(\mathbf{x},\mathbf{x}\right)\end{bmatrix} \tag{97}$$

$$V\in\mathbb{R}^{2n\times 2n}=\begin{bmatrix}V_{11} & V_{12}\\ V_{21} & V_{22}\end{bmatrix} \tag{98}$$

Using a multivariate GPM could be cumbersome for some unfamiliar in the way GPMs work. A multivariate GPM has a more challenging situation for parameter estimation, since the computational cost of calculating the inverse of the new regression covariance matrix V increases to $\mathcal{O}\left(mn^{3}\right)$, where m represents the number of multiple variables to be predicted. But perhaps the biggest challenge in the implementation of multivariate GPM is the appropriate selection of a cross-covariance function for the model that is consistent with the individual correlation function of the predicted

variables [149]. A wrong selection of this function will lead to a non-positive semidefinite V matrix, generating numerical instabilities in the GPM predictions. Many researchers have proposed new cross-covariance structures for the multivariate GPM [16, 41, 60, 82, 109], but the linear corregionalization model [45] is the most commonly used because its simple description to guarantee the positive semidefinite condition on V. To the best of my knowledge, no implementation of a multivariate Gaussian process model has been used in a dynamic framework as in the one presented here, despite the fact that the multivariate GPM provides a full state covariance matrix S with Equation (91).

CHAPTER III

ERROR ESTIMATION IN STOCHASTIC SIMULATIONS USING GAUSSIAN PROCESS MODELS

The research papers in the area of metamodeling approximate expensive deterministic simulations. As it will be shown in this chapter, Gaussian process models have distinctive mathematical characteristics that are attractive as approximate models for expensive deterministic simulations. However, in this research area, it is uncommon to build approximate models for *stochastic* simulations. Only in recent years, researchers in metamodeling started to propose and investigate implementations of Gaussian process models for stochastic simulations. The focus in this chapter is to elucidate the effects of using stochastic simulations in Gaussian process models. This chapter gives a detailed description of how stochastic simulations affect the parameter estimation and error estimation properties of Gaussian process models. Last, a framework for the analysis of error estimation in Gaussian process models is presented in this chapter. This framework is a key element in the overall research work in this thesis, since it will be used in the following chapters to analyze the error estimation in the context of dynamics and multidimensional Gaussian process models.

3.1 Background

Assume that the response of an expensive simulation $\mathbf{f}$ can be described as $y_{tr}(\mathbf{x}) = \mathbf{f}(\mathbf{x})$, where the subscript tr denote the mean response of the simulation at $\mathbf{x}$. The approximate model of the expensive simulation, $\hat{\mathbf{f}}$, predicts the mean response at $\mathbf{x}$ as: $y_{app}(\mathbf{x}) = \hat{\mathbf{f}}(\mathbf{x})$. Therefore, the mean square prediction error of the approximate

model at $\mathbf{x}$ is:

$$\delta^2(\mathbf{x}) = [y_{tr}(\mathbf{x}) - y_{app}(\mathbf{x})]^2 \tag{99}$$

To calculate $\delta^2(\mathbf{x})$ *exactly*, it is necessary to know the value of y_{tr}. In the scenario of expensive simulations, a user may not have the resources to make additional simulations for the validation of the approximate model. This validation problem scales up when the user is approximating *stochastic* simulations, since each observed response y from the simulation is corrupted by a measurement noise η around the true response $y(\mathbf{x}) = y_{tr}(\mathbf{x}) + \eta$. An idea to handle this validation problem is to implement estimators of the prediction error that do not require additional evaluations from the expensive simulation.

Gaussian process model (GPM) is one of the most popular approximate models, not only because its flexibility and good prediction results, but also because it has its own error estimation on the GPM prediction variance. According to the theory of GPM, when the GPM structure is completely known (that is, the true GPM parameter set, the true local correlation structure and the true regression functions in the model are known), the GPM prediction variance is an error estimator of the mean square prediction error. Many of the applications of GPM implement a "plug-in" version of the Gaussian process, using an estimated parameter set. Santner et al. [123] called this version of GPM the empirical best linear unbiased predictor. Several authors already mentioned this situation [24, 29], suggesting that this empirical version of GPM is an underestimator of the "true" GPM prediction variance. These types of issues raises questions about how to understand error estimation in Gaussian process models.

Error estimation measures are useful for the assessment of approximate models. The success in many applications of approximate models depends on the accuracy in the error estimation. Error estimators can be used to quantify the level of uncertainty or "trust" in the prediction of an approximate model, therefore indicating particular

regions in the input space where additional samples might be required. Some examples include improvement of design of experiments in the creation of approximate models, and optimization of time-consuming computer simulations via black-box models [29]. Early findings about error estimation of GPM were made by Meckesheimer et al. [102]. They evaluate leave-k-out cross-validation strategies as a procedure to assess the accuracy of low-order polynomial functions, radial basis functions and kriging models over the design space. Goel et al. [42] presented a detailed study on error estimation, evaluating response surfaces and kriging models for 6 classical benchmark examples in the statistics field. As a result of this study, Goel et al. concluded that local evaluations of GPM prediction variance can be used for global error estimation of Gaussian process models. Viana and Haftka [151] proposed to explain the correlation between the GPM prediction variance and the mean square prediction error using cross-validation, finding out that GPM prediction variance does not always correlate well with the actual prediction errors. As relevant as these results are in the error estimation of GPM, the studies were limited to deterministic simulations.

The presence of noise in the observations incorporates an additional element in the GPM prediction that has been discussed previously in the literature. Kleijnen and coworkers had worked extensively in the use of kriging models for random simulations [75, 76, 142, 143], using replicates at each sample point in the GPM to calculate sample means, and then treating those values as deterministic outcomes in the GPM. In the area of error estimation, the same group implemented a parametric bootstrapping approach to calculate the mean square prediction error of the GPM [29], but it was not used as an error estimator in the approximate model and it was only employed with deterministic simulations. A similar implementation of this bootstrapping approach was also used to evaluate the uncertainty of time-course experimental data in cell signaling pathways and network topology of time-series gene expression data [70]. Ankenman et al. [6] extended Kleijnen's GPM for random simulations with their

stochastic kriging model, which models the intrinsic uncertainty, or noise, in the simulations with an additional variance parameter for each sample point. Different from these papers, where the major interest was the GPM mean prediction, this chapter focuses on the GPM prediction variance and its role as an error estimator of the approximate model when stochastic observations are used in GPM.

3.2 Error estimation for Gaussian process models

In the context of metamodeling, there are two scenarios for error estimation. In one scenario, the user could be interested in a local (or pointwise) error estimation of the approximate model at some sample point $\mathbf{x}$ in the design space. In another scenario, the user could use a single error measure to quantify a global error estimation of the approximate model over the entire design space, as studied by Goel et al. [42]. Gaussian process models offer a direct evaluation of the local error estimation based on its prediction variance. According to Equations (99) and (11)

$$MSE\left[\hat{y}(\mathbf{x})\right] = \mathbb{E}\left[\left(y_{tr}(\mathbf{x}) - \boldsymbol{\lambda}^T(\mathbf{x})\mathbf{y}\right)^2\right] = \mathbb{E}\left[\delta^2(\mathbf{x})\right] \tag{100}$$

By the bias-variance decomposition of the mean square prediction error, and the unbiased constraint used in the derivation of GPM, the mean square prediction error can be estimated as

$$MSE\left[\hat{y}(\mathbf{x})\right] = \left(\mathbb{E}\left[y_{tr}(\mathbf{x}) - \boldsymbol{\lambda}^T(\mathbf{x})\mathbf{y}\right]\right)^2 + \sigma_y^2\left(\mathbf{x}, \mathcal{D}\right)$$

$$MSE\left[\hat{y}(\mathbf{x})\right] = \sigma_y^2\left(\mathbf{x}, \mathcal{D}\right) \tag{101}$$

where $\sigma_y^2\left(\mathbf{x}, \mathcal{D}\right)$ is described by Equation (14). Equation (101) allows the user to have an assessment of the GPM accuracy at $\mathbf{x}$ without additional function evaluations from the expensive simulation, based entirely on the statistical properties of the GPM.

Frequently, global error assessments over the entire design space are based on integrated local error estimates. The most common of those global error metrics is

the root integrated mean square error ($RIMSE$) defined as [21, 42]:

$$RIMSE = \sqrt{\frac{1}{\Omega}\int_{\mathbf{x}*} \delta^2\left(\mathbf{x}\right)\, \mathrm{d}\mathbf{x}}; \quad \Omega = \int_{\mathbf{x}*} \mathrm{d}\mathbf{x} \tag{102}$$

The exact evaluation of Equation (102) for a Gaussian process model is a difficult task. Equation (102) requires a closed-form expression of the expensive computer simulation, which may not be available, as well as the evaluation of the integral for the nonlinear GPM mean prediction.

Usually, a global error estimator of $RIMSE$ is calculated assuming that the prediction error $\delta^2(\mathbf{x})$ is equal to the GPM prediction variance $\sigma_y^2(\mathbf{x})$. The global error estimator is obtained by substituting Equation (14) in Equation (102). To further simplify the calculation of the global error estimator, a summation over all sample points is performed instead of evaluating the integral,

$$RIMSE = \sqrt{\frac{1}{n_t}\sum_{j=1}^{n_t} \sigma_y^2\left(\mathbf{x}_j\right)} \tag{103}$$

where n_t is the number of test points spread across the design space used in the evaluation. As it can be seen on Equation (103), explaining the relationship between $\sigma_y^2(\mathbf{x})$ and $\delta^2(\mathbf{x})$ at the local error level is vital for an accurate error estimation at the global error level.

3.3 Case studies and testing implementation

This section describes the different case studies that are used to evaluate the error estimation properties in Gaussian process models in the presence of stochastic observations. Error estimates are computed using the GPM prediction variance, Equation (14). The analysis of the error estimation is based on its assessment for local prediction error, as was explained in Section 3.2. The main interest with this analysis is to understand the role of the uncorrelated noise parameter σ_u^2 in the error estimation of GPM, and to provide recommendations for future GPM users.

3.3.1 Test problems and creation of noisy observations

The set of test problems was selected from a database of deterministic functions [30] that are frequently used in evaluating methods for global optimization.

Camelback function (CB)

$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{1}{3}x_1^4\right) x_1^2 + x_1 x_2 + \left(-4 + 4x_2^2\right) x_2^2$$
$$x_1 \in [-2, 2], \quad x_2 \in [-1, 1] \tag{104}$$

Branin-Hoo function (BH)

$$f(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$
$$x_1 \in [-5, 10], \quad x_2 \in [0, 15] \tag{105}$$

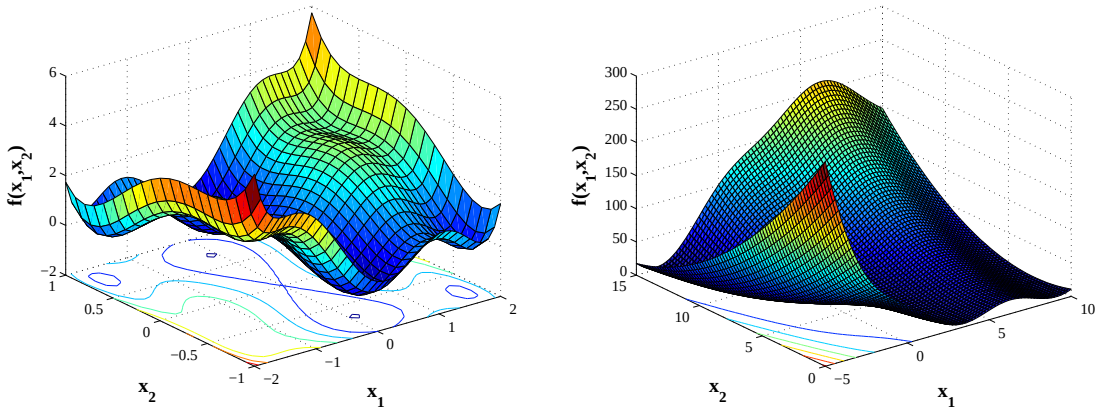
Hartman-3 function (HM3)

$$f(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right], \quad a_{ij}, c_i, p_{ij} \in \mathbb{R}$$
$$\mathbf{x} = [x_1, x_2, x_3], \quad x_i \in [0, 1] \tag{106}$$

Table 1: Parameters used in the Hartman-3 function with three variables

i	$\mathbf{a_{ij}}$			$\mathbf{c_i}$	$\mathbf{p_{ij}}$		
1	3.0	10	30	1.0	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3.0	10	30	3.0	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

The error estimation analysis for the test problems begins with the selection of an experimental design to create the GPM. The experimental design is created using the Latin hypercube algorithm built in MATLAB, incorporating a "maximin" criterion to ensure that the sample points are distributed over the design space. Table 2 shows the number of n sample points used to built the GPM for each test problem. The



(a) (b)

Figure 6: Test functions: (a) Camelback function (b) Branin-Hoo function.

values of n were chosen to match the values in Reference [42]. At each of the n sample points, the deterministic test problem is evaluated and its corresponding outcomes are stored. In order to make comparisons between the different test problems, the sample points and their corresponding observations are normalized between 0 and 1. Table 2 contains the values of the global minimum and maximum of the different test functions for normalization. Finally, after the normalization procedure, stochastic observations are created by adding a zero mean, Gaussian distributed and independent noise with a variance of σ_n^2. The magnitudes of σ_n^2 that are used in the analysis are

$$\begin{aligned} \sigma_n^2 &= \big[0, 1\times10^{-8}, 1\times10^{-6}, 1\times10^{-4}, 4\times10^{-4}, \\ &\quad 2.5\times10^{-3}, 1\times10^{-2}, 4\times10^{-2}, 2.5\times10^{-1}, 1\big] \end{aligned} \tag{107}$$

Table 2: Summary of Gaussian process model implementation for error estimation analysis

Test Problem	**Output Normalization**		**n**	$\mathbf{n_t}$
	Maximum	**Minimum**		
Camelback Function	5.7333	-1.0316	30	300
Branin-Hoo Function	308.1291	0.3979	20	300
Hartman-3 Function	3.7729×10^{-5}	-3.8628	70	900

3.3.2 Building Gaussian process models

The GPMs for the different test problems are constructed using the normalized input/output information. All GPMs are built using a constant regression function as a trend component. The local correlation between sample points in the normalized dataset is modeled using the Gaussian correlation function, shown in Equation (9). To obtain the GPM parameter set $\hat{\boldsymbol{\theta}}$ in the local correlation function, maximum likelihood estimation (MLE) is implemented as a constrained nonlinear optimization. The constraints in the parameter estimation are:

$$1 \times 10^{-3} \leq \ell_i \leq 1; \quad 1 \times 10^{-9} \leq \sigma_c^2, \sigma_u^2 \leq 10 \tag{108}$$

For the analysis of error estimation, two different GPM implementations are used. One of the GPM implementations does not include σ_u^2 as one of its estimated parameters. This GPM implementation is desirable for expensive deterministic simulations because it will interpolate the noise-free observations. Because of this characteristic, it will be called *interpolator GPM*. The second GPM implementation does include σ_u^2 in its estimated parameter set, which it is called from now on *regression GPM*. The main idea here is to see how an interpolator GPM will work with noise in the observations, and to contrast the different implementations for GPM.

3.3.3 Error estimation analysis

The objective in the error estimation analysis is to describe the relationship between a local error estimator and the true prediction error at some sample point $\mathbf{x}_t$ in the design space. For each test problem, n_t test sample points were selected in the design space according to Table 2. At each of the n_t sample points, the true prediction error $\delta(\mathbf{x}_t)$ and its corresponding GPM prediction variance $\sigma_y^2(\mathbf{x}_t)$ were computed and stored. These calculations are repeated using 2000 different experimental designs to account for the variability in the creation of a GPM. For example, in the case of the Branin-Hoo function, 6×10^5 pairs of $\left[\delta(\mathbf{x}_t), \sigma_y^2(\mathbf{x}_t)\right]$ are stored. To analyze this

large amount of data, the values of $\sigma_y^2(\mathbf{x}_t)$ were organized in ascending order. Then, the information is divided into bins of 1000 data points. For each of the bins, the sample mean of $\delta(\mathbf{x}_t)$, the sample variance of $\delta(\mathbf{x}_t)$ and, the sample mean of $\sigma_y^2(\mathbf{x}_t)$ were calculated. The goal is to relate the average $\sigma_y^2(\mathbf{x}_t)$ in a bin to the *distribution* of the $\delta(\mathbf{x}_t)$ having similar values of $\sigma_y^2(\mathbf{x}_t)$. Finally, the entire mentioned procedure is repeated for each of the σ_n^2 noise levels that are used in the stochastic simulations, to analyze the effect of noise in the error estimation of the Gaussian process models.

3.4 Results

3.4.1 Understanding stochastic simulations in a Gaussian process model

Before describing how to perform error estimation on a Gaussian process model, it is necessary to explain the effects of stochastic simulations in the model, more specifically on its parameters. Previously, Yin and coworkers [158] studied the influence of stochastic simulations in the estimation of the range parameters ℓ_i of a GPM, and their consequences on the mean square prediction error. However, these results were limited to a perfect estimation of the variance parameters σ_c^2 and σ_u^2, and to a single experimental design on the design space.

Traditional implementations of GPM for deterministic observations do not include σ_u^2 as one of the parameters to be estimated. The presence of σ_u^2 as an additional parameter in GPM transforms the interpolation model into a regression model, which is the more appropriate implementation for stochastic observations. Interpolation through each observation does not make sense when the samples are stochastic and could lead to erroneous predictions. However, the interpolator GPM is built using stochastic observations for comparison purposes, because it shows clearly how the GPM range parameters ℓ_i^2 and the correlated variance parameter σ_c^2 change when measurement noise is present in the observations.

Figure 7 shows how the different noise levels in the stochastic observations have

an effect on the interpolator GPM parameters. In a GPM, the range parameters ℓ_i describe the distance-based local correlation, and the correlated variance parameter σ_c^2 describes the corresponding data variability of the mean behavior of the function. When the noise level in the simulations is low, the data variability in the observations is due to the different mean values of the $\mathbf{f}$ simulation across the design space. As a result, the value of σ_c^2 remains fairly constant and ℓ_i is larger than zero (Figures 7a and 7b). When the noise level in the stochastic simulations increases, the estimated value of σ_c^2 in the interpolator GPM is similar to the noise added in the observations, and the range parameters ℓ_i decrease their values close to zero. As a result of this, the interpolator GPM assumes that all data variability in the observations is due to uncorrelated noise and is not capable of identifying a local correlation.

In an interpolator GPM, the effects of a local correlation in the data decrease when the range parameters ℓ_i in the Gaussian correlation function of Equation (9) also decrease. In more practical terms, when the range parameters ℓ_i decrease, the off-diagonals elements in the regression covariance matrix V and the entries in the correlation vector $\mathbf{k}\left(\mathbf{x}_t, \mathcal{D}\right)$ in Equations (13) and (14) also decrease. In the limit that all these entries become zero, the interpolator GPM mean prediction of a test point $\hat{y}\left(\mathbf{x}_t\right)$ is equal to the sample mean of the stochastic simulations and, the interpolator GPM prediction variance $\sigma_y^2\left(\mathbf{x}_t\right)$ is equal to the estimated value of σ_c^2. In other words, an interpolator Gaussian process model without local correlation predicts similarly to an ordinary linear least-squares model that uses the set of regression functions in the GPM.

Figure 7 illustrates an important issue in the implementation of the interpolator GPM for stochastic simulations. An interpolator Gaussian process model has a limit in the magnitude of the noise that it can handle. Figures 7c and 7d illustrate the behavior of the interpolator GPM around this noise limit behavior in the observations. Figure 7c shows the GPM mean prediction of 10 test points $\mathbf{x}_t$ from the

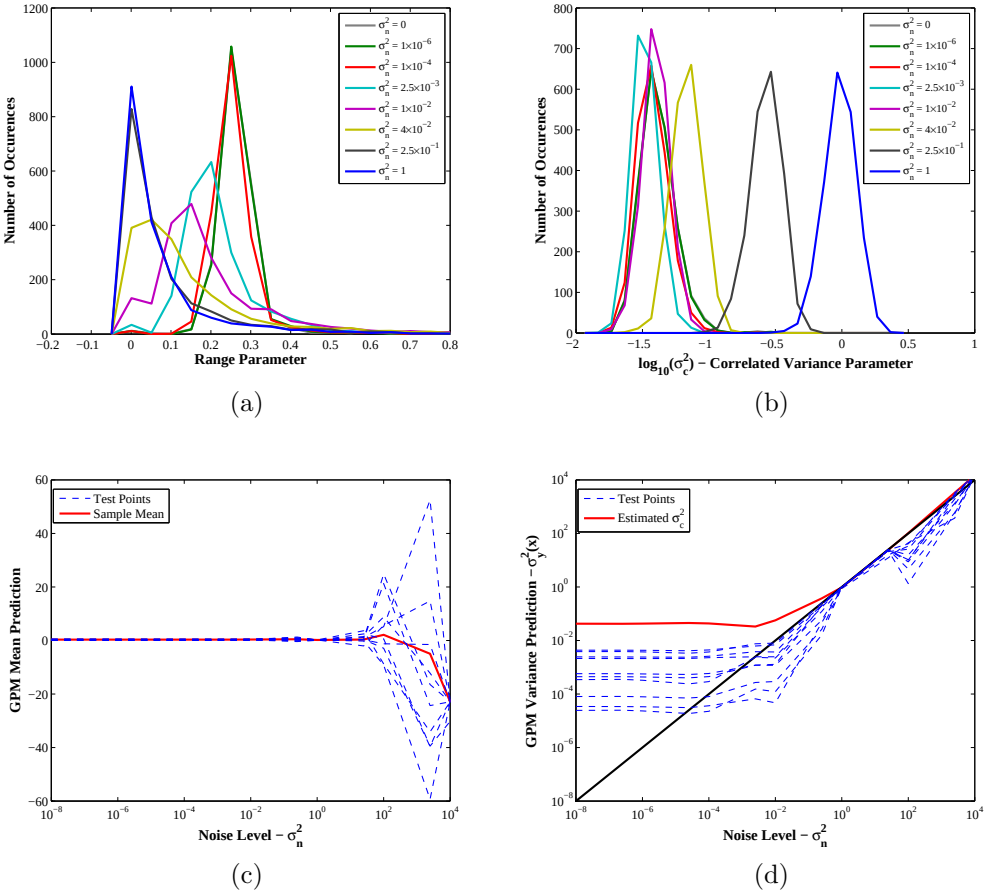


Figure 7: Effect of stochastic observations on interpolator Gaussian process models. Test problem: Camelback function. The distributions of estimated parameters in (a) and (b) were computed from 2000 different experimental designs, using the maximum likelihood estimator at different noise levels in the stochastic observations. (a) Estimated range parameters ℓ_i. (b) Estimated correlated variance parameter σ_c^2 on a 10-base logarithmic scale. (c) GPM mean prediction $\hat{y}(\mathbf{x})$ at 10 test points in the design space for a typical experimental design. (d) GPM predictive variance $\sigma_y^2(\mathbf{x})$ at 10 test points in the design space for a typical experimental design.

same experimental design at different noise levels in the observations. The figure also shows the sample mean of the stochastic observations at each of those noise levels. Figure 7d represents the GPM prediction variance for the same 10 test points, and the estimated σ_c^2 as a function of the noise level in the observations.

Based on the results in Figure 7, it is possible to identify the noise level when the identification of local correlation in the data decreases in the interpolator GPM.

According to this figure, an interpolator GPM starts to lose its local correlation features when the value of the noise is close to the value of σ_c^2, approximately at $\sigma_n^2 = 1 \times 10^{-2}$. It is more practical to describe this limit in terms of a signal-to-noise ratio of the observations. Since the interpolator GPM is using normalized output information $\mathcal{O}(1)$ in its experimental design, the noise limit for an interpolator GPM is a signal-to-noise ratio of:

$$SNR = \frac{\mu}{\sigma} = \frac{\mathcal{O}(1)}{\sigma_n} = \frac{1}{0.1} = 1 \times 10^1 \tag{109}$$

A regression GPM includes σ_u^2 as an extra parameter, which decomposes the data variability in the stochastic simulation between local correlation and an uncorrelated measurement noise. Figure 8 illustrates the balance between the local correlation and the noise, based on the ratio between σ_c^2 and σ_u^2. In Figure 8, a perfect identification of the variance parameters σ_c^2 and σ_u^2 is imposed in the regression GPM and the remaining GPM parameters (the range parameters ℓ_i) are estimated by the maximum likelihood estimator. This means that in Figure 8, σ_u^2 is equal to the noise level in the observations σ_n^2, and σ_c^2 is equal to the estimated σ_c^2 value when $\sigma_n^2 = 0$).

Figures 8a and 8b show how σ_u^2 changes the GPM mean prediction $\hat{y}(\mathbf{x})$ as its value gets closer to σ_c^2. Figures 8a illustrates the prediction of the regression GPM at one of the training points used in the model, and Figure 8b illustrates the regression GPM mean prediction of one of the testing points $\mathbf{x}_t$. At low σ_u^2 values, the mean prediction of the regression GPM at a sample point (Figures 8a) works almost as an interpolator GPM. In this scenario, the local correlation is more significant for the GPM prediction. As σ_u^2 increases, the prediction of the regression GPM begins to deviate from the original output sample, since both the σ_u^2 and σ_c^2 terms are relevant. For high-noise cases, where σ_u^2 is larger than σ_c^2, the identification of a local correlation in the data set becomes more difficult, and the regression GPM mean prediction at a training point approaches the sample mean over all the stochastic simulations in the sample set $\mathcal{D}$. A similar behavior occurs for the training point in Figure 8b.

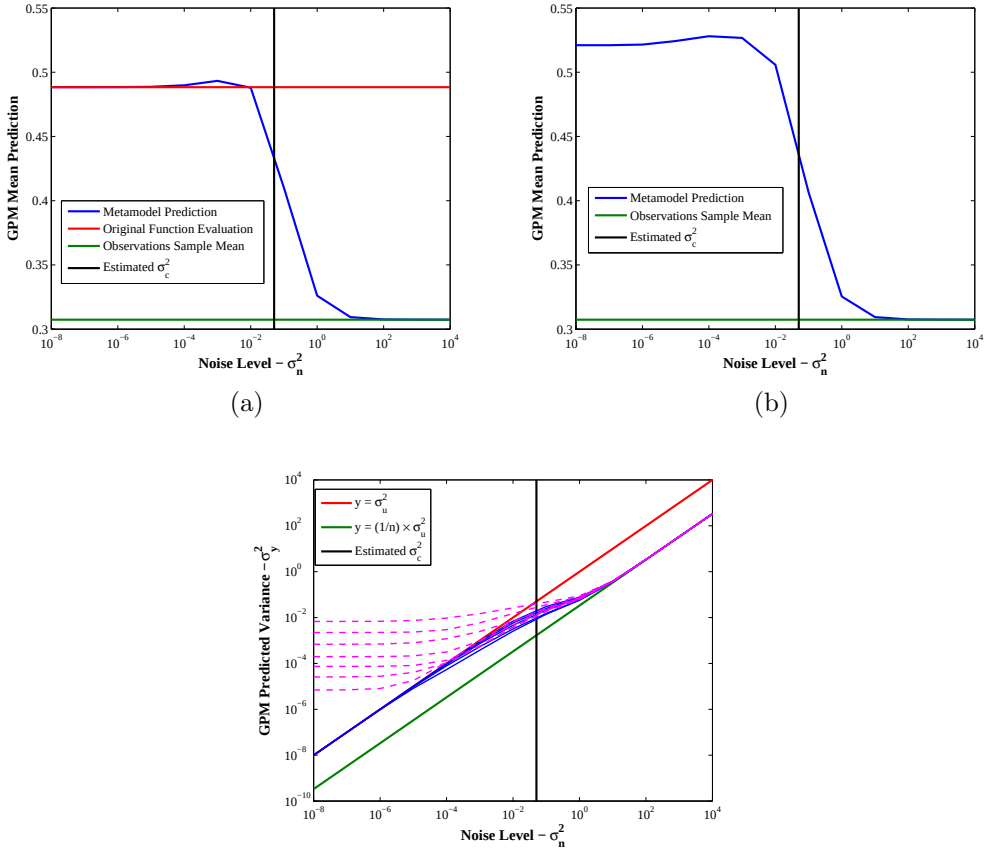


Figure 8: Effect of stochastic observation on a regression Gaussian process models. Test problem: Camelback function. $n = 30$. (a) GPM mean prediction $\hat{y}(\mathbf{x}_i)$ at a sample point $\mathbf{x}_i$ in a typical experimental design. (b) GPM mean prediction $\hat{y}(\mathbf{x}_t)$ at a test point $\mathbf{x}_t$ in the design space. Figure (c) shows the GPM predictive variance $\sigma_y^2(\mathbf{x})$ as a function of σ_u^2 using (c) $n = 30$ sample points. Blue solid lines represent GPM prediction variances at sample points in the set $\mathcal{D}$, magenta dotted lines represent GPM prediction variances at different test points in the design space and black vertical line represent the estimated σ_c^2 for $\sigma_n^2 = 0$

Figure 8c illustrates the qualitative characteristics of GPM prediction variance under various values of σ_u^2. It is easy to see the low-noise and high-noise sections in the GPM prediction variance. In the limit of high noise levels in the observations, the regression GPM prediction variance at a test point $\sigma_y^2(\mathbf{x}_t)$ is equal to $\frac{\sigma_u^2}{n}$, the variance of the sample mean of the stochastic simulations. This result indicates a significant difference in the error estimation between the interpolator and the regression GPM

with high noise. In Figure 8c, the GPM prediction variance at different sample points in the design space start to deviate from the 1:1 line as the noise level in the observations increases. As a result, the GPM prediction variance becomes lower than σ_u^2. This behavior in the prediction variance at the sample points seems to be correlated with the ratio between the estimated σ_u^2 and σ_c^2, more precisely, when $\frac{\sigma_c^2}{\sigma_u^2} \approx 10^2$. When the noise in the observations increases, the GPM prediction variance becomes similar to the prediction variance when a constant regression function is used for prediction, $\frac{\sigma_u^2}{n}$.

Finally, this section analyzes how the presence of the uncorrelated variance parameter σ_u^2 affects the local error estimation of the GPM prediction variance σ_y^2. Different from Figures 7 and 8, all parameters in the GPM in Figure 9 are estimated by a maximum likelihood estimator, emulating a usual implementation for GPM users. This change in the parameter estimation allows one to analyze the efficacy of the maximum likelihood estimator in the identification of measurement noise. Figure 9 shows the error estimation made by the interpolator and regression GPM in the presence of noise in the observations for two scenarios, $n = 30$ and $n = 100$, without replicates in the observations. In the case of deterministic observations $\sigma_n^2 = 0$ and with $n = 30$, both interpolator and regression GPM (Figures 9a and 9c respectively) behave similarly, because of the small value of the estimated σ_u^2 in the regression GPM. When the observations contain a noise level of $\sigma_n^2 = 1 \times 10^{-6}$, Figures 9b and 9d, there are not significant differences compared to the deterministic results, suggesting that the noise level added to the observations is too small to modify the GPM prediction results. It is also clear how the scatter plots line up around the $\sigma_y^2(\mathbf{x}) = \delta^2(\mathbf{x})$ line, supporting the proposed linear relationship between $\sigma_y^2(\mathbf{x})$ and the expected prediction error.

Figure 9d illustrates the difficulties in the accurate estimation of the measurement noise with σ_u^2 in the GPM, since the estimated σ_u^2 is not close to the noise level in the simulations. When the number of sample points in the model increases to $n = 100$,

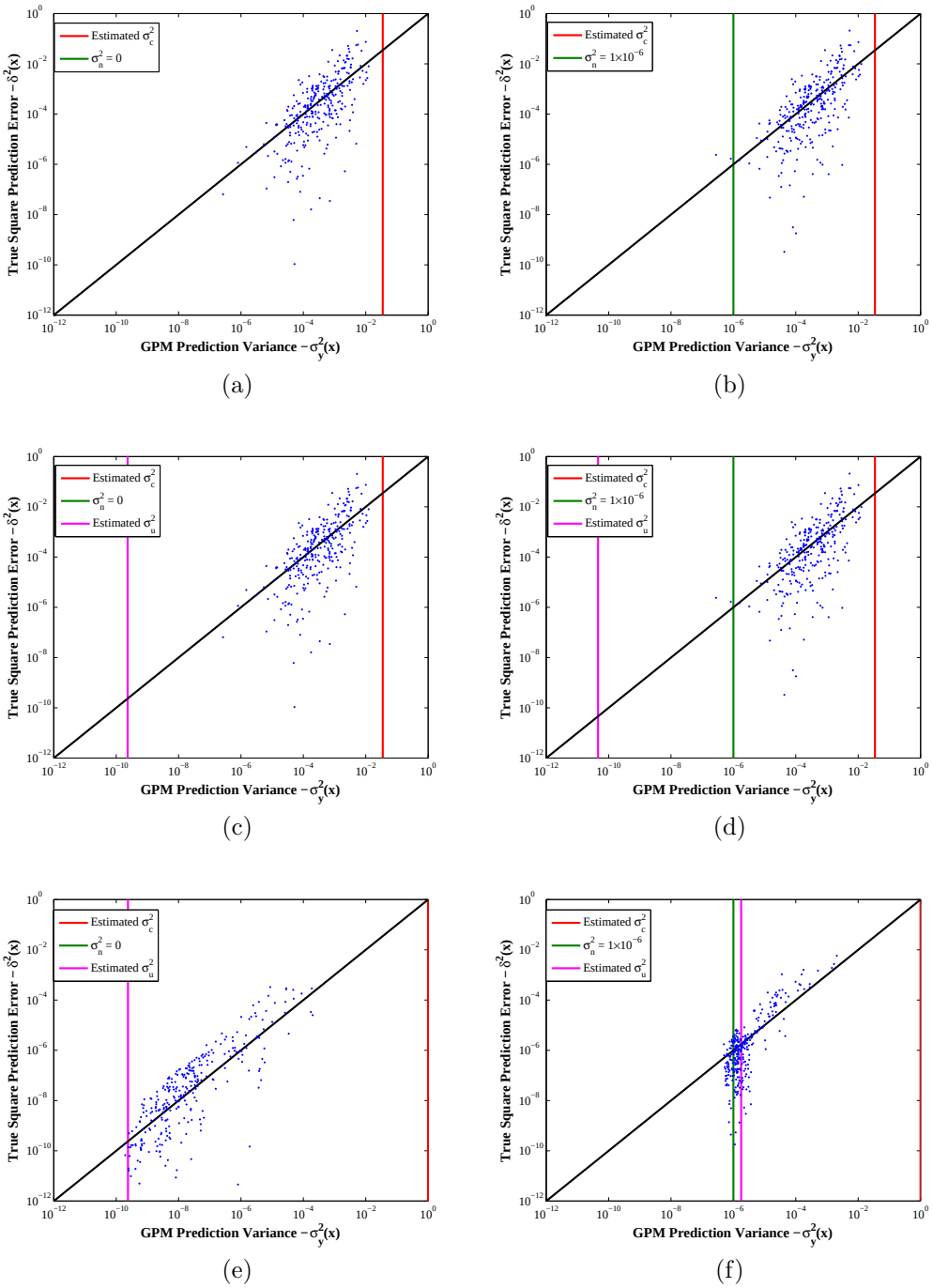


Figure 9: Local error estimation using GPM prediction variance $\sigma_y^2(\mathbf{x})$ using stochastic observations. Test problem: Camelback function. The noise level σ_n^2 is labeled in each of the figures. (a) and (b) do not include σ_u^2 in the GPM, $n = 30$. (c) and (d) σ_u^2 in the GPM, $n = 30$. (e) and (f) include σ_u^2 in the GPM, $n = 100$.

Figures 9e and 9f, the maximum likelihood estimator improves the identification of the measurement noise from the observations. Figures 9 shows the role of σ_u^2 in the regression GPM as a potential lower range for the GPM prediction variance. When the true square prediction error δ^2 is lower than the estimated σ_u^2, the GPM prediction variance σ_y^2 looks like a constant value around the estimated σ_u^2. When δ^2 is greater than the estimated σ_u^2, σ_y^2 exhibits a linear correlation with δ^2, similar to the results with the interpolator GPM.

In conclusion, the regression Gaussian process model also has a noise limit for the identification of the local correlation. Similar to the interpolator GPM, the noise limit for the regression GPM is a signal-to-noise ratio of 1×10^1. The main difference in this noise limit case between the two GPM implementations is in how the signal-to-noise ratio can be calculated. While in the interpolator GPM, the user may need additional information in the form of repetitions to estimate the signal-to-noise ratio in the simulations, in the regression GPM, the signal-to-noise ratio can be in terms of the ratio between the estimated σ_u^2 and σ_c^2. Therefore, a regression Gaussian process model identifies the local correlation between mean values of the simulations, as long as the estimated σ_u^2 is two or more orders of magnitude lower than the estimated σ_c^2. Thus, the user may diagnose a GPM by its relative values of σ_c^2 and σ_u^2. If $\sigma_u^2 \geq \sigma_c^2$, then the GPM should not be expected to yield accurate predictions. This conclusion is limited to Gaussian process models using a constant regression function and a small sample size.

3.4.2 Error estimation in Gaussian process models

Figure 10 shows the results of implementing the error estimation analysis described in Section 3.3.3. In this section, the value of the uncorrelated variance parameter σ_u^2 is imposed to be equal to the value of the noise level in the observations. Figures 10a and 10b provides a picture of the collected data in the error estimation, by performing

the same error estimation analysis described in Section 3.3.3, now with 20 different experimental designs and for the Branin-Hoo test problem. In Figure 10a, all collected δ values are centered around zero, confirming the unbiased prediction properties of the GPM across the design space. This figure also shows how the δ distribution changes as a function of σ_y^2. As was described previously, these results are consequences of the local correlation that is present in the regression GPM. On the other hand, in Figure 10b, the extremely high noise level in the observations causes the regression GPM to lose the identification of the local correlation. The vertical stripes correspond to individual realizations in which all points have the same value of $\sigma_y^2 = \frac{\sigma_u^2}{n}$.

Figures 10c and 10d shows the combined results of all the collected data points, as was described in Section 3.3.3. On the left, Figure 10c illustrates the transition in the δ distribution as the noise level in the observations increases. At low noise levels, the sample mean of δ for each bin of the regression GPM is close to zero, but the uncertainty in the nominal value of δ increases as the noise level increases. More interesting is the relationship between the sample variance of the δ distribution with σ_y^2, as is shown in Figure 10d. These variables have a strong linear correlation that covers from the low noise levels up to $\sigma_n^2 = 1 \times 10^{-2}$. This knowledge can be used to predict the error distribution at a particular point $\mathbf{x}$, regardless of its location in the design space. At higher noise levels, the sample variance of the δ distribution levels off such that it no longer corresponds to the value of σ_y^2.

As important as the values of the sample mean and sample variance of the δ distribution are, the shape of the distribution defines many of its properties. Figure 11 shows the histograms of some of the bins that were created in the analysis. The main idea is to describe what type of distribution can be used to interpret the δ distribution, and how the noise level in the observations might or might not alter the shape of the distribution. The presence of noise in the measurement incorporates a source of uncertainty in the predictions made by the regression GPM. That is why

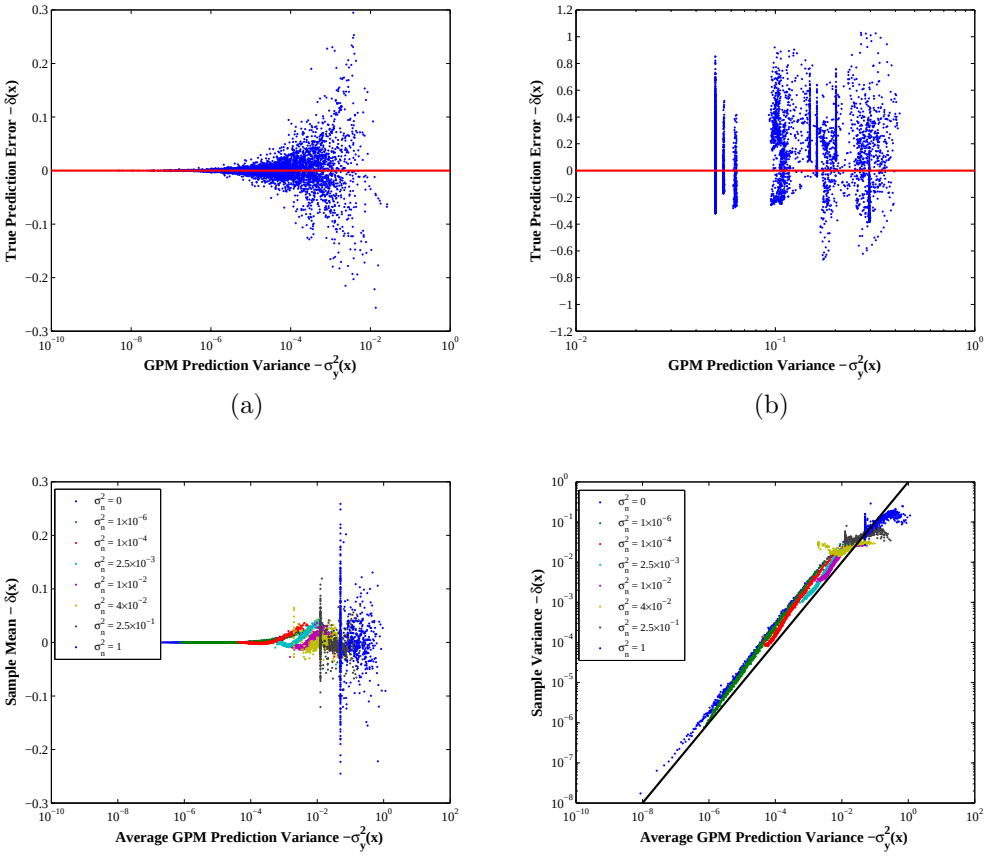


Figure 10: Description of true prediction error $\delta(\mathbf{x})$ distributions. (a) Scatter plot of $\delta(\mathbf{x})$ values for 20 different experimental designs, $\sigma_n^2 = 0$. (b) Scatter plot of $\delta(\mathbf{x})$ values for 20 different experimental designs, $\sigma_n^2 = 1$. Figures (c) and (d) shows sample mean and variances of the different bins created from scatter plots of 2000 different experimental designs at different σ_n^2 noise levels in the simulations. Test problem: Branin-Hoo function.

in Figure 11 there is an increase in the width of all distributions as the noise level increases.

In each of the subfigures in Figure 11, the average value of σ_y^2 in each of the corresponding bins is shown. Notice in the subfigures that the average value of σ_y^2 increases from left to right. While the $\delta(\mathbf{x})$ distributions on the left (Figures 11a, 11d and 11g) look like normal distributions, the $\delta(\mathbf{x})$ distributions on the right (Figures 11c, 11f and 11i) have a slightly bimodal character. This finding suggests that there

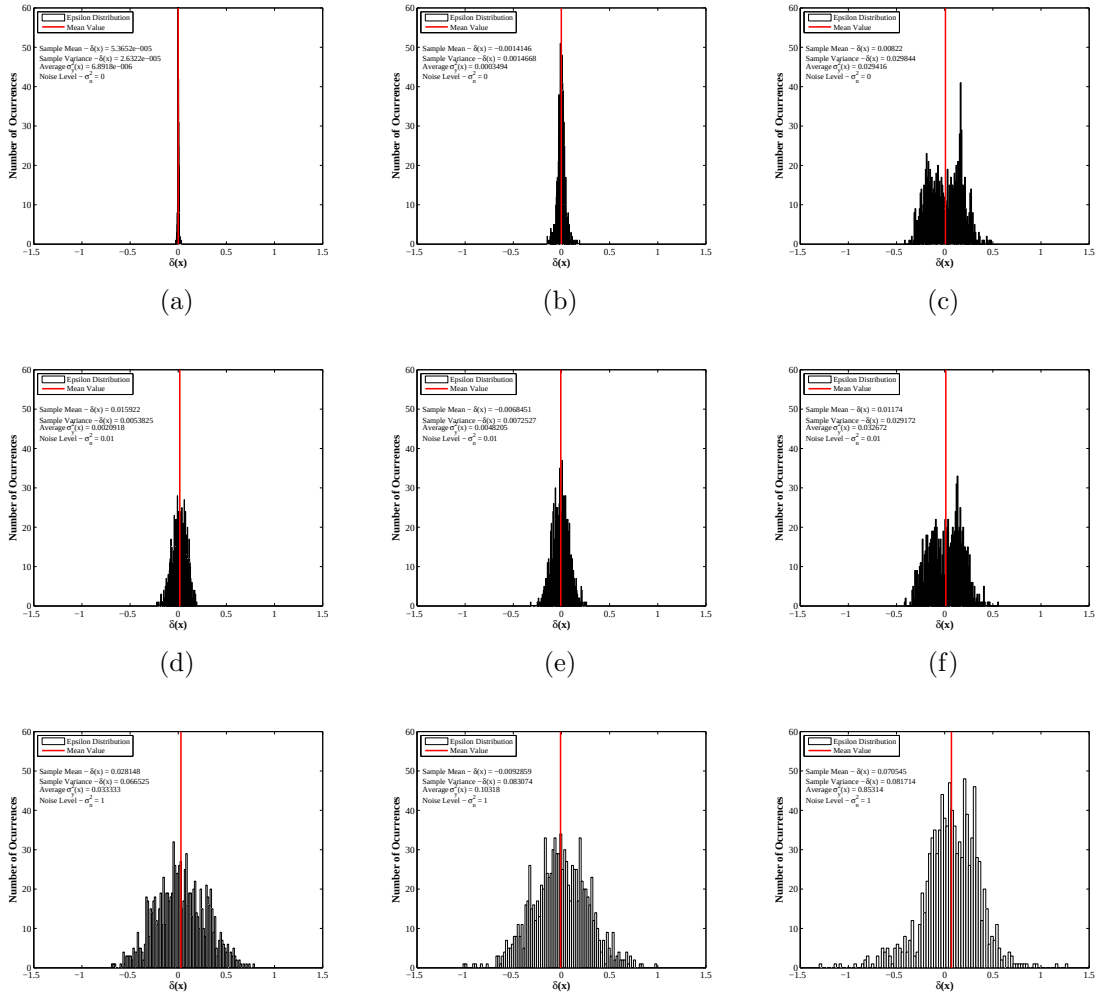


Figure 11: Delta $\delta(\mathbf{x})$ distributions of three different bins at three different noise levels for the Camelback test function. Figures (a), (b) and (c) corresponds to $\sigma_n^2 = 0$. Figures (d), (e) and (f) corresponds to $\sigma_n^2 = 1 \times 10^{-2}$. Figures (g), (h) and (i) corresponds to $\sigma_n^2 = 1$.

is a correlation between the value of the local error estimator σ_y^2 and the shape of the δ distribution. Moreover, this result suggests that there is a threshold value of σ_n^2 where the Delta distribution changes from a normal distribution to, in the case of the Camelback function, a bimodal distribution. However, even in the highest noise bins, the peaks of the distribution are overlapping and the distribution may still be approximated as a Gaussian distribution.

This section of results concludes with a more detailed exploration of the linear correlation between the sample variance of the δ distribution and σ_y^2. Figure 12 shows the relationship between the sample variances and the average value of σ_y^2 up to a noise level of $\sigma_n^2 = 4 \times 10^{-2}$ for the three test problems. These figures indicate a monotonic relationship between σ_y^2 and the variance of the prediction error distribution $\delta(\mathbf{x})$, plus $\sigma_y^2(\mathbf{x})$ have similar orders of magnitude with this variance. This results reinforces the idea of using the GPM prediction variance for finding regions with high uncertainty in its prediction. Although these figures show more clearly the linear correlation between the variables, it is also interesting to see a small portion of the data that levels-off at high values of σ_y^2. This threshold value corresponds to the variance on the residuals of the underlying true function when it is fitted by a constant regression function. In other words, this threshold value represents the data variability of the true mean values of the test problem across the design space. The value of this cannot be predicted directly from the stochastic observations, since it requires a complete knowledge of the true mean values of the function to be approximated.

3.4.3 Error estimation of Gaussian process models under limited number of function evaluations

In order to aid in the estimation of the noise level, a user could include repetitions of the function evaluations at some sample points. When the total number of function evaluations to be used in the GPM is fixed, the user must make the decision of how to distribute the resources between the repetitions and covering the design space for a good mean prediction. In this section, the effects of incorporating repetitions into the regression GPM are evaluated, as well as its effect parameter estimation and on the error estimation of GPM using stochastic simulations. The number of total function evaluations in the different test problems is defined according to the values in Table 2. All the repetitions in the calculations are obtained at the center of the normalized design space. In each of the figures in this section, the fraction of function evaluations

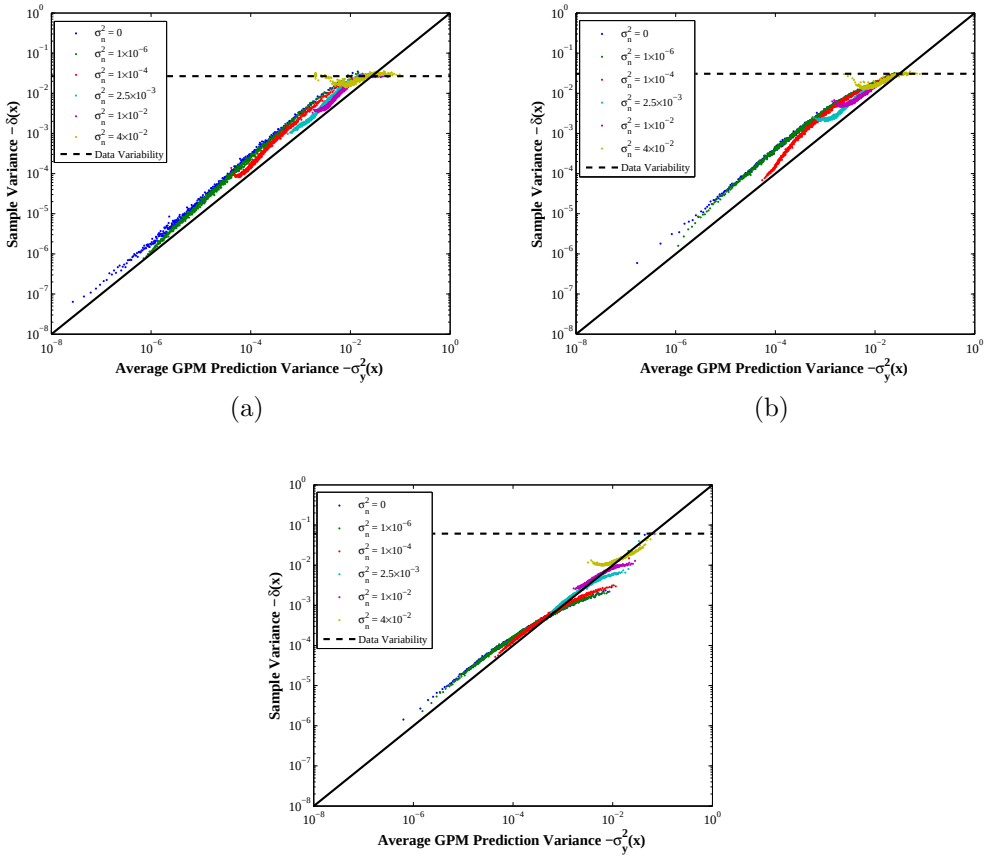
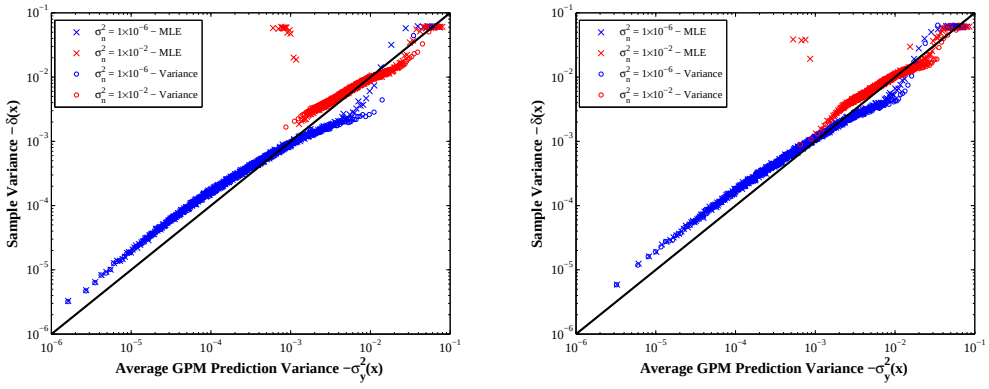


Figure 12: Describing the noise level limit in the Delta $\delta(\mathbf{x})$ distribution for different test problems. Figure (a): Branin-Hoo function. Figure (b): Camelback function. Figure (c): Hartman-3 function.

that are used for repetitions is specified (e.g. in the case of a Branin-Hoo function, a fraction of 0.3 means that 6 out of 20 total function evaluations are repetitions at the center of the design space), while the remainder of the function evaluations are spread out over the design space, following a Latin Hypercube design.

The repetitions at the center of the design space are intended to improve the identification of the uncorrelated variance parameter σ_u^2 by the maximum likelihood method. To compare the performance of the MLE in this situation, an additional approach for the estimation of σ_u^2 was implemented. Instead of estimating all the

GPM parameters using the MLE simultaneously, the sample variance of the repetitions at the center point of the design space is used as the estimated σ_u^2, and then the remaining of the GPM parameters are estimated by MLE. In this way, an independent estimation of the uncorrelated variance parameter is guaranteed. Figure 13 presents a comparison between the two GPM parameter estimation procedures using the relationship between the sample variance of the error distribution and the average value of σ_y^2. Figure 13 shows that the maximum likelihood estimator performs almost identically to the proposed sample variance approach for the estimation of noise in the simulations. However at a small number of sample points in the regression GPM, the MLE yields unreliable results for the higher noise level, whereas the sample variance approach is more robust. MLE is capable of identifying the noise level in the stochastic simulations, aided by a few repetitions in the dataset or with a large number of single evaluations on the design space.



(a) (b)

Figure 13: Comparison between different parameter estimation when repetitions are used in the Gaussian process model at different σ_n^2 noise levels. Test problem: Hartman-3 function. $n = 70$. Figure 13a as a fraction of repetitions equal to 0.1, Figure 13b as a fraction of repetitions equal to 0.3.

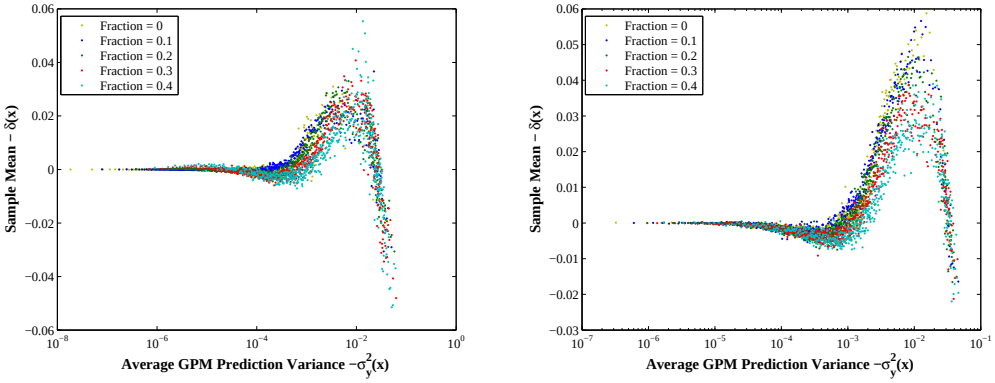
After looking into how the maximum likelihood estimator works with repetitions in the sample points, the next question is how the error distribution of the regression GPM changes when some of the function evaluations are used for repetitions.

Figure 14 illustrates the effect of the modified experimental design in the mean and variance of the prediction error distribution using MLE as a parameter estimation methodology.

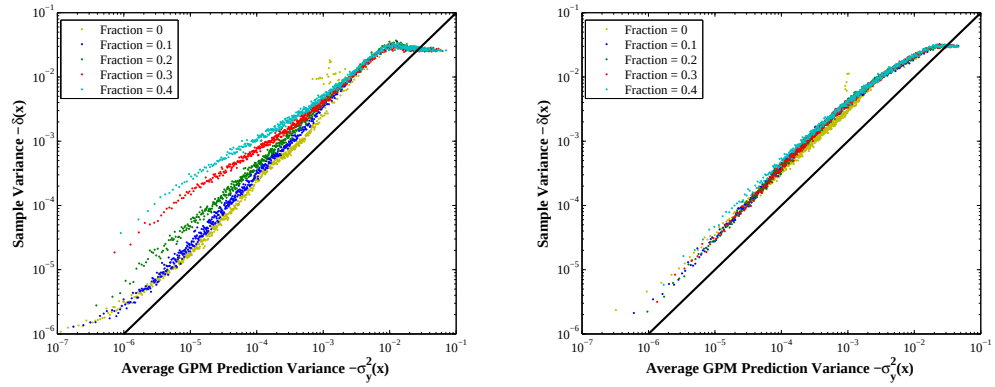
Figures 14c to 14f show interesting patterns, related to the presence of repetitions in the regression GPM and the variance of the prediction error distribution. At the lower noise level, the two test problems exhibit completely different behaviors. For the Branin-Hoo function (Figure 14c), the presence of repetitions in the regression GPM seems to not improve the potential 1:1 relationship between the variance of δ and σ_y^2. In fact, σ_y^2 becomes an even greater underestimator of the variance of the prediction error distribution. In contrast, for the Camelback function at a low noise level (Figure 14d), there is no significant effect in the scatter plots due to the number of repetitions in the GPM. Figure 14e shows the effects of the repetitions on the δ distribution at an intermediate noise level for the Branin-Hoo function. In this case, the repetitions do improve the relationship between the variance of the prediction error distribution and σ_y^2, by making the scatter plot closer to the 1:1 line up to a fraction of repetitions equal to 0.2. Once the fraction of repetitions in the GPM reaches 0.3, the scatter plot moves away from the 1:1 line. This behavior of the variance of the δ distribution can be seen more clearly on Figure 14f for the Camelback function. The results on Figures 13 and 14 could be associated with a trade-off between the exploration of the underlying true function and the repetitions used to identify the noise in the stochastic simulations.

3.5 Discussion

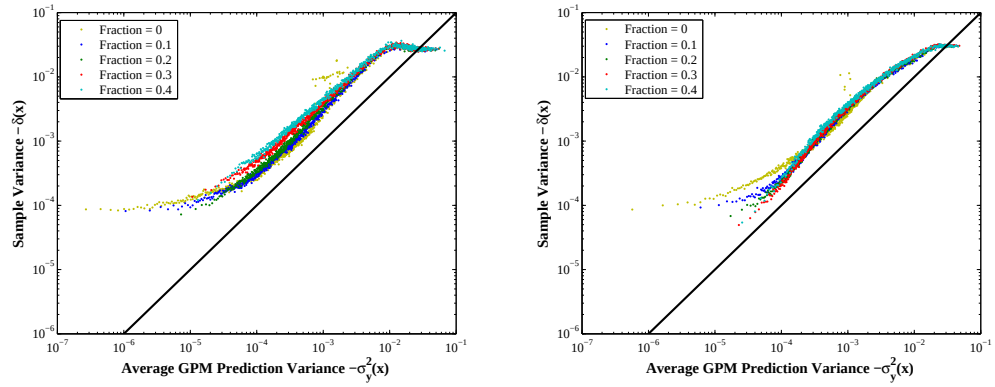
Although the main subject on this work is the analysis of error estimation on Gaussian process models using stochastic simulations, the results describe how the GPM balances its behavior between the noise and the signal in the observations. In the



(a) (b)



(c) (d)



(e) (f)

Figure 14: Mean and variance of the prediction error distribution as a function of the number of repetitions and noise level σ_n^2. The fraction of function evaluations used as repetitions is labeled in each of the figures. Noise level in Figures (a) to (d): $\sigma_n^2 = 1 \times 10^{-6}$. Noise level in Figures (e) and (f): $\sigma_n^2 = 1 \times 10^{-4}$. Figures (a), (c) and (e) corresponds to the Branin-Hoo function and Figures (b), (d) and (f) corresponds to the Camelback function.

end, this relationship defines all the error estimation aspects of the Gaussian process model. If the GPM is capable of recognizing the underlying true function to be approximated, via its local correlation features, then the model has a clear correspondence between the prediction error of the approximate model and its prediction variance. When this correlation is present in the GPM, the prediction error at a particular point in the design space can be modeled as a normally distributed, zero-mean random variable, whose variance is approximately equal to the GPM prediction variance, Equation (14), at that point. If the GPM does not recognize the true function (because of the presence of noise in the observations), then GPM exhibits the same results as a much more simple model: the ordinary least-squares model. Because of this limitation in the implementation of the GPM, the recommendation is to use a Gaussian process model when the signal-to-noise ratio in the stochastic simulations is larger than 1×10^1, (i.e. 10 % noise or less).

In the implementation of Gaussian process models for stochastic simulations, the first step is to decide an appropriate number of function evaluations to be used in the model. The rule of thumb $n_t = 10 \times d$ has been used over the years in the area of metamodeling to build approximate models [17, 62]. Recently, Loeppky and coworkers [90] verified this recommendation using an interpolator Gaussian process model in high-dimensional deterministic simulations. However, the rule was not evaluated for stochastic simulations or using a regression GPM, where the rule must be modified to account for the noise effects in the simulations.

Section 3.4.3 offers a different perspective for the area of design of computer experiments. The use of repetitions in the dataset is advantageous for the parameter estimation of a GPM, and the MLE takes advantage of these repetitions. Fixing the number of function evaluations to be used in the regression GPM proves to be an interesting scenario for the discussion. With increasing repetitions, the GPM decreases in its ability to identify the underlying true function via the local correlation,

because of the reduction in the exploration of the design space. This conclusion is particularly true when the noise level in the simulations is small. These ideas suggest that an experimental design for a Gaussian process model should favor the space-filling of the design space, with a few repetitions that can aid the parameter estimation of σ_u^2. In this way, the regression GPM can preserve its local correlation features and, therefore the error estimation characteristics that have been described throughout this study. Although the MLE is capable of estimating the noise from the repetitions, it may be more robust to estimate the uncorrelated variance parameter σ_u^2 using the sample variance of the repetitions. In this way, the user guarantees to decompose the data variability in the stochastic observations between local correlation and measurement noise.

Several review papers have summarized the large number of design of computer experiments for Gaussian process models [117, 127]. Unfortunately, most of these experimental designs were developed for deterministic simulations. A sequential experimental design for random simulations in Gaussian process models is proposed [143] using several repetitions of the sample points and a bootstrapping approach to quantify the uncertainty in the GPM mean prediction. Their proposed approach allows for multiple stopping criteria, including a fixed number of function evaluations in the approximate model, but it fits an interpolator GPM using the sample mean of the repetitions at each sample points as if they were deterministic outcomes. Although this approximation is reasonable to filter the noise from the stochastic simulations, there is some uncertainty in the value of the sample mean relative to the true mean value of the function that could be captured by σ_u^2, especially at small number of function evaluations. In addition, the interpolation of noisy observations could induce oscillations in the approximated surface, making the optimization more difficult.

All these effects become even harder to estimate when a non-constant noise level

is present in the stochastic simulation. Some authors have proposed to modified the diagonal of the covariance matrix in the Gaussian process model by estimating the variances at each of the sample points [6, 158], although it would required additional function evaluations. All these issues could be use to redefine the concept of design of computer experiments and to foster new and original ideas for the research area.

Further studies and discussions in the error estimation of Gaussian process models could include the selection of the remaining elements in the model. The problem of the identification of local correlation from the stochastic simulation is independent of the correlation function used in the GPM. The selection of a correlation function only has an effect in the diagnosis of the error estimation and in the conditions when the local correlation of the GPM is lost. A different situation occurs in the selection of the regression functions for the Gaussian process models. The results presented here are limited to a constant regression function, which explains the behavior of the model at high-noise levels in the simulations. If the basis set of regression functions used in the GPM was input-dependent (like a polynomial basis functions), the identification of the noise limit for the GPM would be more difficult, since part of the data variability in the observations is now explained by the regression functions and not by the local correlation. This situation suggests that the value of the estimated σ_c^2 with a complex regression function will be smaller than the estimated σ_c^2 when a constant regression function is used. As a result of this, it becomes harder to identify the local correlation features of the GPM and, the recommended signal-to-noise ratio to preserve the error estimation characteristics discussed in this study will be above the current limit of 1×10^1.

To summarize this discussion, this study focuses on the error estimation properties of the Gaussian process models, not the accurate mean prediction made by the model. The objective in this study is to verify the consistency between the prediction error of the approximate model and its GPM prediction variance, even if the prediction error

is above the desired target value for the GPM user. Based on the results of this chapter, and if the user has some flexibility in the total number of function evaluations that can be used, the GPM implementation presented by Ankenman et al. [6] for building Gaussian process models with stochastic simulations will be recommended. Their GPM implementation contains a preliminary data processing stage where the noise in the stochastic simulations is filtered by calculating repetitions at the sample points and considering the sample mean of these repetitions as the outcome to used in the GPM. The major difference of this GPM implementation, compared to the GPM implementation in Reference [143], is that the sample means are treated as random values, which allows for the implementation of a regression GPM instead of a interpolator GPM. By changing the number of repetitions in the data processing stage, the GPM implementation can be tuned to satisfy the recommended signal-to-noise ratio in the stochastic simulations. The estimation of the uncorrelated variance parameter σ_u^2 could be made using MLE, or using the sample variance of the repetitions at each sample point as a non-constant noise level in the diagonal of the correlation matrix, for a maximum usage of the available information. This implementation can be complemented with a tailored selection of the number of repetitions at each sample point as it is described by Kleijnen and coworkers [75].

CHAPTER IV

PROPAGATION OF ERROR IN AN ITERATIVE MAPPING USING GAUSSIAN PROCESS MODELS

Discrete time models are a traditional approach to understand and explain system dynamics in science and engineering. A discrete time model is a recursion function that maps dynamic information from a previous time step to the next time step at a fixed (discrete) time interval. The scenario presented in this thesis is that the user is building an approximate closed-form equation for the system dynamics, based on the simulated data from the expensive dynamic simulations. This dynamic data is used to approximate the recursion function with an empirical model, in this case a Gaussian process model. Like any other empirical model, there is an error in the prediction every time that the model is used but, because of the recursive nature of the discrete time model, errors in previous time steps will also propagate into future time steps.

The propagation of error is a critical aspect in the analysis and implementation of data-driven dynamic models. This chapter analyzes the effects of error propagation when a Gaussian process model is used in the recursive dynamic framework for a simple chemical kinetics model. In particular, this chapter addresses the questions:

1. Can the error estimation properties of a Gaussian process model, explored in Chapter 3, be used to describe the propagation of error in the dynamic framework?

2. What is the effect of using stochastic data on the propagation of error made by the Gaussian process model?

4.1 Local and dynamic error in the GPM dynamic framework

Consider a stochastic dynamic system described as a discrete-time model using Equation (49) and the approximate discrete-time model of the GPM dynamic framework in Equation (50). Each of these two expressions map the dynamic information from $\mathbf{x}(s)$ to the next time step $\mathbf{x}(s+1)$, making a one-step-ahead prediction every time. Figure 15 shows a scheme describing the one-step-ahead prediction made by the true discrete-time model of the system and the mean of the Gaussian process model. The accuracy of the approximate discrete-time model can be evaluated using the Euclidean distance between the true state $\mathbf{x}(s+1)$, and the predicted state from the approximate dynamic GPM $\hat{\mathbf{x}}(s+1)$.

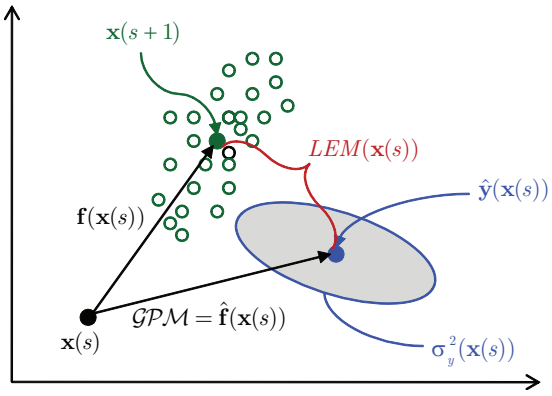


Figure 15: Local Error Analysis. The black lines represents the prediction paths using the true recursion function and the GPM. The red bracket represents the local error measurement (LEM) associated with the prediction at $\mathbf{x}(s)$. The shaded region describes a statistical tolerance region centered on the GPM mean prediction $\hat{y}(\mathbf{x}(s))$ using the GPM prediction variance $\sigma_y^2(\mathbf{x}(s))$.

This research work defines the local error measurement ($LEM\left[\mathbf{x}(s)\right]$) as the variable that quantifies the one-step-ahead prediction error of the GPM at the state $\mathbf{x}(s)$:

$$\begin{aligned} LEM\left(\mathbf{x}(s)\right) &= \frac{1}{d}\sum_{i=1}^{d}\left(x_i(s+1)-\hat{x}_i(s+1)\right)^2 \\ &= \frac{1}{d}\sum_{i=1}^{d}\left(x_i(s+1)-\hat{x}_i\left(\mathbf{x}(s)\right)\right)^2 \\ &= \frac{1}{d}\sum_{i=1}^{d}\left(x_i(s+1)-\hat{y}_i\left(\mathbf{x}(s)\right)\right)^2 \end{aligned} \tag{110}$$

where $x_i(s+1) \in \mathbb{R}$ is the true state value in the i^{th} dimension, and $\hat{y}_i\left(\mathbf{x}(s)\right)$ is the GPM mean prediction in the i^{th} dimension, using Equation (13). Notice that the local prediction error depends on the location in the state space $\mathbf{x}(s)$, therefore the accuracy of the GPM will depend on the local correlation of information in the neighborhood of $\mathbf{x}(s)$. The definition of LEM can be used to define a global one-step-ahead prediction error, by computing several values of $LEM\left(\mathbf{x}(s)\right)$ at different states $\mathbf{x}(s)$ across the input space, and averaging them.

Once the first dynamic prediction is made by the GPM, the local prediction error propagates during the recursive dynamic prediction. Figure 16 describes the relationship between the propagation of error and the local prediction error made at each discrete time step. Consider a particular dynamic trajectory, whose initial state is $\mathbf{x}_0 \in \mathbb{R}^d$ at the discrete time step $s = 0$. This thesis defines the dynamic error measurement ($DEM\left(\mathbf{x}_0, s\right)$) as the variable that quantifies the prediction error made by the GPM dynamic framework for the dynamic trajectory as:

$$\begin{aligned} DEM\left(\mathbf{x}_0, s\right) &= \frac{1}{d}\sum_{i=1}^{d}\left(x_i(s+1)-\hat{x}_i(s+1)\right)^2 \\ &= \frac{1}{d}\sum_{i=1}^{d}\left(x_i(s+1)-\hat{x}_i\left(\hat{\mathbf{x}}(s)\right)\right)^2 \\ &= \frac{1}{d}\sum_{i=1}^{d}\left(x_i(s+1)-\hat{y}_i\left(\hat{\mathbf{x}}(s)\right)\right)^2 \end{aligned} \tag{111}$$

DEM uses the Euclidean distance between the true state in the i^{th} dimension of the dynamic system $x_i(s+1)$, and the GPM mean prediction $\hat{y}_i\left(\hat{\mathbf{x}}(s)\right)$ using the uncertain input state from the previous iteration $\hat{\mathbf{x}}(s)$. Comparing Equations (110) and (111), LEM is based on a known input state $\mathbf{x}(s)$, whereas DEM evaluates the prediction error using the estimated $\hat{\mathbf{x}}(s)$. The definition of DEM can be used to define a dynamic error for a complete dynamic trajectory that starts at $\mathbf{x}_0$, by calculating the average $DEM\left(\mathbf{x}_0, s\right)$ over n_s, the number of time steps or iterations necessary to describe it. Notice that $LEM\left(\mathbf{x}(s=0)\right) = DEM\left(\mathbf{x}_0, s=0\right)$ since it is assumed that the initial value of the dynamic trajectory $\mathbf{x}_0$ does not have uncertainty.

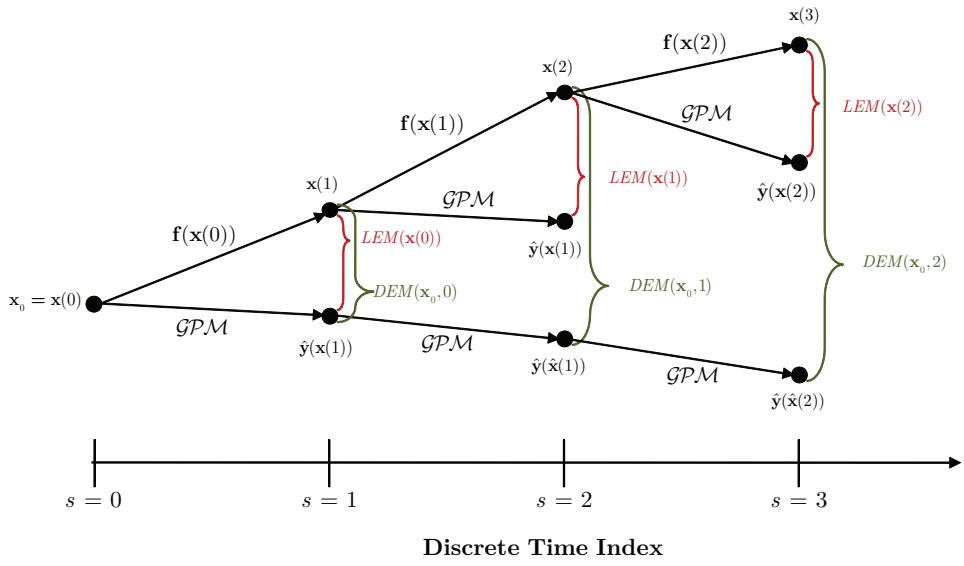


Figure 16: Dynamic Error Analysis. The figure describes how the prediction error propagates from one discrete time index to the next one by a recursive dynamic GPM and compares the use of LEM and DEM in a dynamic context.

4.2 Case study: Second-order reaction rate

To describe the propagation of error in the GPM dynamic framework, a second-order reaction rate equation is chosen, described as

$$\frac{dC}{dt} = -kC^2, \quad k = 0.005 \ \frac{\mathrm{m}^3}{\mathrm{mol\ s}} \tag{112}$$

where C represents the concentration of a species in $\frac{\text{mol}}{\text{m}^3}$, and k represents the reaction rate constant. The set of initial conditions C_0 that are used to evaluate the dynamic model is

$$0 \ \frac{\text{mol}}{\text{m}^3} \leq C_0 \leq 100 \ \frac{\text{mol}}{\text{m}^3} \tag{113}$$

The reasons to choose such simple dynamic model are:

1. It is a one-dimensional dynamic system, which avoids the problem of not having a full state covariance matrix S during the implementation of a multidimensional GPM dynamic framework.

2. The state space of the dynamic system is fully specified to be between 0 and 100 $\frac{\text{mol}}{\text{m}^3}$, thanks to the decaying behavior of the system. This complete description of the state space allows one to have complete control of the sampling design used in the GPM. Later in Chapter 5, this subject will be revisited and explained in more detail for a multidimensional Gaussian process model.

3. It has a known and analytical solution

$$C(t) = \frac{C_0}{1 - kC_0 \left(t - t_0\right)} \tag{114}$$

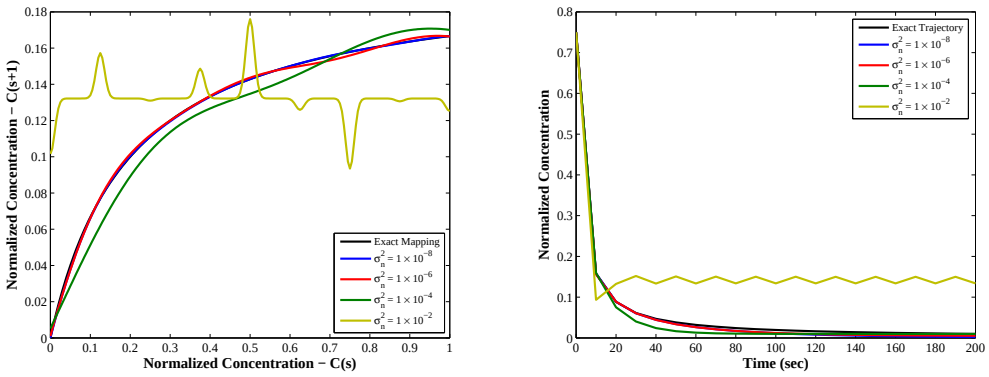
   where C_0 and t_0 are the initial concentration and time values to solve the ordinary differential equation in Equation (112).

4. This simple model has a closed-form equation that could be used as a discrete-time version of it. If a fixed discrete time interval Δt is chosen, Equation (114) can written as a discrete-time model.

$$C(s+1) = \frac{C(s)}{1 - kC(s)\Delta t} \tag{115}$$

   In other words, the Gaussian process model should approximate Equation (115), in the input space between 0 and 100 $\frac{\text{mol}}{\text{m}^3}$ for a complete description of the dynamic trajectories in the recursive dynamic framework.

Figure 17 shows an example of the implementation of the GPM dynamic framework as an approximate discrete-time model for Equation (115). Similar to the case studies in Chapter 3, the information provided by the second-order reaction model is scaled between 0 and 1. The difference in the scaling procedure is that the input and output information are in the same space (i.e. between 0 $\frac{\text{mol}}{\text{m}^3}$ and 100 $\frac{\text{mol}}{\text{m}^3}$), thus the input and output information are scaled with the same values. This situation explains why the y-axis on Figure 17a goes between 0 and 0.16, instead of the range 0 to 1.



(a) (b)

Figure 17: Description of a second-order reaction model. (a) Scaled dynamic mapping function for the second-order reaction rate in Equation (115) with $\Delta t = 10$ s and the approximated GPM mapping functions at different noise levels. (b) Scaled dynamic prediction made by the GPM for the initial concentration $C_0 = 75$ $\frac{\text{mol}}{\text{m}^3}$ at different noise levels in the observations. The GPM uses $n = 20$ sample points equally spread across the input space 0–1.

After the scaling, a zero-mean, Gaussian noise with variance σ_n^2 is added to the recorded output information. Following the signal-to-noise recommendation in Chapter 3, the noise levels evaluated in this comparison are:

$$\begin{aligned} \sigma_n^2 &= \big[1 \times 10^{-8}, 1 \times 10^{-6}, 4 \times 10^{-6}, 2.5 \times 10^{-5}, \\ &\quad 1 \times 10^{-4}, 4 \times 10^{-4}, 2.5 \times 10^{-3}, 1 \times 10^{-2}\big] \end{aligned} \tag{116}$$

The addition of noise to the output offers the possibility of creating negative measured concentrations, especially at high noise levels. When the dataset contains negative concentrations, the GPM correlates this erroneous output information, generating unphysical dynamic predictions. To avoid this situation, a consistency check is added during the creation of stochastic output information, making any negative concentration to be replaced by a zero concentration value. With this consistency check in place, a GPM is build at different noise levels, and dynamic predictions are made in the scaled input space. Figure 17b shows the dynamic predictions made by the GPM recursive framework at different noise levels. The major disadvantage in enforcing this consistency check is an induced bias in the stochastic observations, especially in the cases of low scaled concentrations and high noise levels.

For this case study, two types of experimental designs are evaluated. One is an experimental design where $n = 20$ sample points are equally spaced in the region between 0 and 1. The second experimental design uses $n = 20$ sample points randomly selected from a uniform distribution in the same range. Once an experimental design is generated, 5 additional points are included as repetitions of the output information at the scaled input $\mathbf{x} = 0.5$. With this dataset of input/output information, a regression GPM is built, using a constant regression function and maximum likelihood as the parameter estimation methodology. To evaluate the local error measurement (LEM), $n_t = 101$ sample points equally spaced in the range between 0 and 1 are used, to compare the GPM mean prediction $\hat{y}(\mathbf{x})$ with the exact dynamic mapping function in Equation (115). For a robust quantification of the local prediction error, average values of LEM and $\sigma_y^2(\mathbf{x})$ at each of the n_t test locations are calculated from the prediction results of 1000 different experimental designs, for each of the two types of experimental designs. In the particular case of the equally spaced experimental design, the locations of the sample points in the dataset remain constant, but their stochastic output information in each experimental design is different due to the

random noise component.

To evaluate the performance of the GPM dynamic framework, 51 equally spaced initial values $\mathbf{x}_0$ were selected in the range between 0 and 1 of the input space. Then, a dynamic trajectory was calculated and stored from each of them, using a sampling rate $\Delta t = 10$ s, until a final time $t_f = 200$ s. For each of these dynamic trajectories, a GPM dynamic prediction is generated recursively using the GPM. According to the selected sampling rate and final time, the GPM dynamic framework uses $n_s = 20$ iterations in order to make a dynamic prediction. These dynamic trajectories are used to evaluate the propagation of error in a Gaussian process model. This comparison uses the two modified GPM predictive distributions (Taylor-series approximation and Gaussian approximation) that consider input uncertainty, which were presented previously in Section 2.5.2. The comparison uses the modified mean predictions of these two GPM versions, Equations (59) and (70), and the true value in each of the dynamic trajectories, Equation (114), to calculate DEM values. Additionally, an error estimation analysis similar to the one described in Section 3.3.3 is implemented at each discrete time step of the dynamic predictions, to investigate how the corrected forms of the GPM predicted variance $\sigma_y^2(\mathbf{x})$, Equations (60) and (71), correlate with the propagated error. The analysis of propagation of error in this chapter does not include the GPM correction due to parameter uncertainty described in Section 2.3.1. Preliminary simulations using the GPM parameter uncertainty corrections show that this correction does not improve significantly the error estimation properties of the GPM.

4.3 Results

4.3.1 Understanding the one-step-ahead prediction error

Although the second-order reaction model is a simple dynamic model, it allows to isolate and illustrate general aspects that also appear in a more complex and multidimensional model. Figure 18 shows the comparison between the LEM and the GPM prediction variance $\sigma_y^2(\mathbf{x})$ at different locations in the input space. The first element to notice in all of the figures is the effect of the 5 repetitions at the center of the input space, especially in the cases of low noise levels in the observations. It is clear that these repetitions not only decrease the LEM local error at $\mathbf{x}_i = 0.5$, but also decrease the LEM values of the neighboring input values.

Figures 18a and 18b correspond to the average values of LEM and $\sigma_y^2(\mathbf{x})$ at the $n_t = 100$ different locations of the scaled input space when the random experimental design is used in the GPM. The GPM has substantial problems in the one-step-ahead prediction at the boundaries of the input space. This situation occurs because these regions are farther away from the sample points used to generate the model, which means that less information is available for the distance-based local correlation. The potential consequences of this situation could be severe in the dynamic recursive framework, especially when one of the boundaries in the input space represents a physical constraint or stable attractor, like the zero concentration limit in this example.

In contrast, the implementation of the equally spaced experimental design in the GPM, shown in Figures 18c and 18d, exhibits a better performance in the prediction over the input space. The LEM and $\sigma_y^2(\mathbf{x})$ are more uniform across the input space. As the noise level increases in the stochastic observations, the value of the GPM prediction variance $\sigma_y^2(\mathbf{x})$ approaches to the value of $\frac{\sigma_n^2}{n}$, like in the results of Chapter 3. Also, as the noise level increases, the location of the training points of the GPM becomes more evident, as seen in the oscillations in Figure 18d. Including sample

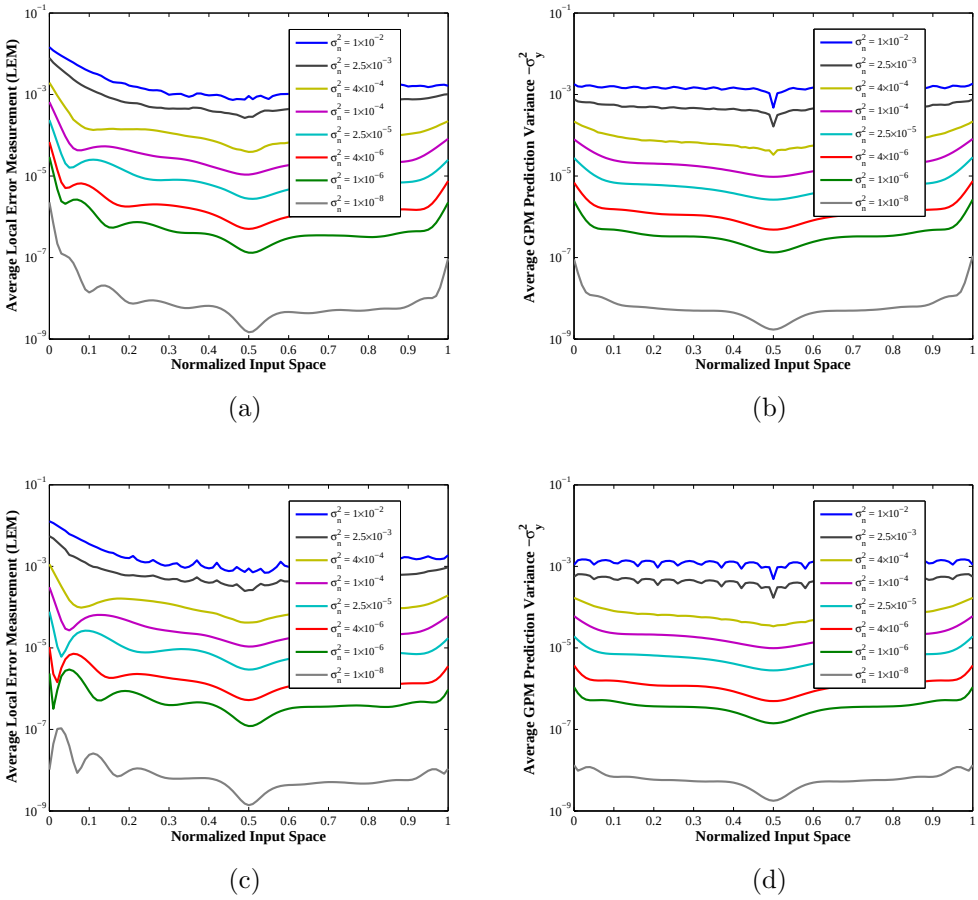
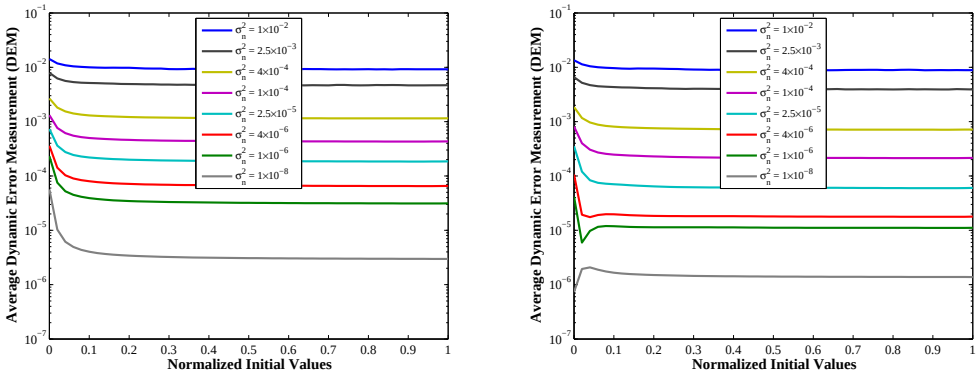


Figure 18: Average Local Error Measurement (LEM) and average GPM prediction variance σ_y^2 for the discrete time model, Equation (115), $\Delta t = 10$ s. Figures (a) and (b) correspond to 1000 random experimental designs, each of $n = 20$ sample points. Figures (c) and (d) correspond to 1000 realizations of an equally spaced experimental design with $n = 20$ sample points.

points at the boundaries of the scaled input space decreases the error prediction in those regions. Figure 18c shows that, despite the equally spaced experimental design across the scaled input space, the LEM prediction error is not uniform in the input space. There is a significant increase in the LEM values in the left side of the scaled input space. This situation is related with the presence of a larger gradient of the mapping function in the region, as shown in Figure 17a. Overall, there is a strong correspondence between LEM and $\sigma_y^2(\mathbf{x})$, indicating that the GPM prediction variance can be used to inform the user about the expected prediction error of $y(\mathbf{x})$.

4.3.2 Understanding the prediction error in a complete dynamic trajectory

Similar to the one-step-ahead prediction error, it is important to see how the error in a complete trajectory behaves, for different initial values. To do this, the definition of DEM in Equation (111) and the 51 pre-selected dynamic trajectories are used. For each of the dynamic trajectories, an average DEM value over the $n_s = 20$ iterations necessary to construct a GPM dynamic prediction is calculated (this number comes from the discrete time interval $\Delta t = 10$ s and the final time $t_f = 200$ s). Then, this average calculation is repeated for 1000 different experimental designs, either using an equally spaced or a random experimental design. Those values are averaged again in order to construct Figure 19. The predictions in this figure are computed using the traditional GPM mean prediction in Equation (13). The average DEM value on the equally spaced experimental design is smaller compared to the DEM values in the random experimental design. The reason for the increase in the DEM values of the random experimental design is the high prediction error near the boundaries, particularly at low noise levels.



(a) (b)

Figure 19: Average Dynamic Error Measurement (DEM) over complete dynamic trajectories for the discrete time model, Equation (115), $\Delta t = 10$ s. Figures (a) corresponds to 1000 random experimental designs, each of $n = 20$ sample points. Figures (b) corresponds to 1000 realizations of a equally spaced experimental design with $n = 20$ sample points.

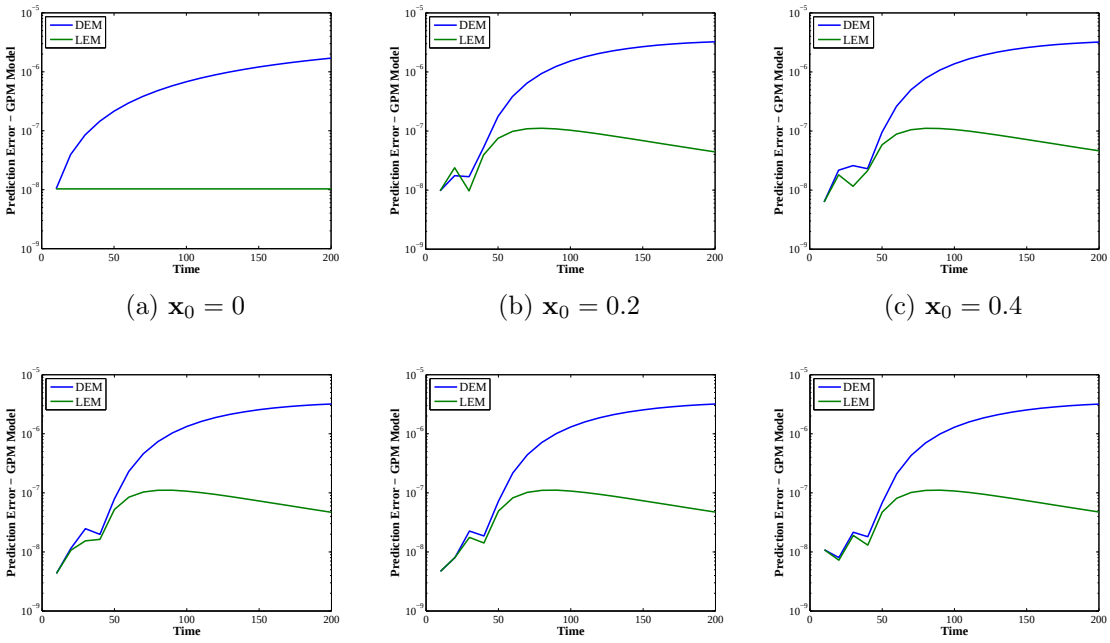
It is interesting to see in Figure 19 that there are not significant differences in the average DEM for different initial values in the input space except when $\mathbf{x}_0$ is very small, less than about 0.05. Similarly, the spacing between each of the training points in the uniform experimental design is approximately 0.05. This result is because all dynamic trajectories are monotonically decreasing to zero concentration. Since all dynamic trajectories are moving across the same regions in the input space, it is expected that most of the trajectories will have similar prediction errors, which later on results in similar average DEM values. This result is a consequence of the system dynamics and not because of a mathematical property in the GPM dynamic framework.

Figure 19 also shows results on the propagation of error in the GPM dynamic framework, as a function of the noise level in the observations used in the model. Based on Figure 18c, the average value of LEM in the scaled input space at $\sigma_n^2 = 1 \times 10^{-8}$ is approximately 1×10^{-8}, while the average value of DEM at the same noise level in Figure 19b, is approximately 1×10^{-6}. The rationale here is that the average value of DEM should be larger than the average value of LEM, due to the propagation of error. But, when the noise level σ_n^2 in the observations increases, the difference between the average DEM and the average LEM decreases. The reason for this behavior could be related to the prediction properties of a GPM at high noise levels. In Chapter 3, it was shown that, at high noise levels in the observations, the GPM prediction becomes similar to the prediction of an ordinary least-squares estimator of the regression functions used in the model (Section 3.4.1). In this chapter, all GPMs use a constant regression function, so it means that at high noise levels the GPM prediction is close to a constant number. (To be more precise, the sample mean of the stochastic observations used in the GPM). If the GPM prediction is a constant number, the recursive dynamic prediction made by the GPM will predict the same constant number at each time step of the dynamic trajectory. Therefore,

the propagated error in the recursive dynamic framework is a constant value, which is in the same order of magnitude of the one-step-ahead prediction error of the model.

Figure 19 is a summary of the global performance of the GPM dynamic framework, but it does not give many indications about specific details of the GPM dynamic prediction for a single trajectory. Figure 20 provides a comparison between DEM and LEM for six different dynamic trajectories, at $\sigma_n^2 = 1 \times 10^{-8}$. In this figure, the GPM uses the equally spaced experimental design to compare the DEM and LEM at each discrete time step throughout the dynamic prediction. Instead of calculating an average DEM for an entire trajectory over the n_s recursive iterations, Figure 20 calculates the average DEM values at each discrete time step for 1000 different dynamic predictions of the same dynamic trajectory, one from each of the 1000 experimental designs. Then, the figure compares this dynamic progression with the LEM values that would correspond to each discrete time step of the trajectory, if the input from the previous time step does not have uncertainty (similar to Figure 16).

From all the dynamic trajectories presented in Figure 20, the most interesting and important is Figure 20a, which corresponds to $\mathbf{x}_0 = 0$. Since the equally spaced experimental design used in the GPM collects information at $\mathbf{x}_i = 0$, the value of the LEM remains constant and close to the noise level in the observations. But, because the type of GPM used in the dynamic framework performs regression and not *interpolation*, the GPM prediction deviates from zero at that sample point. As a result, the propagation of error occurs in the dynamic prediction and the predicted trajectory is completely different from the true, zero constant behavior. As seen in Figure 17b, the system decays to a scaled concentration of $C \leq 0.052$, within the first 2 or 3 discrete time steps. Thus, the error for all simulations in Figure 20 is dominated by the accuracy of the GPM near $\mathbf{x} = 0$, as shown in Figure 20a.



(a) $\mathbf{x}_0 = 0$ (b) $\mathbf{x}_0 = 0.2$ (c) $\mathbf{x}_0 = 0.4$

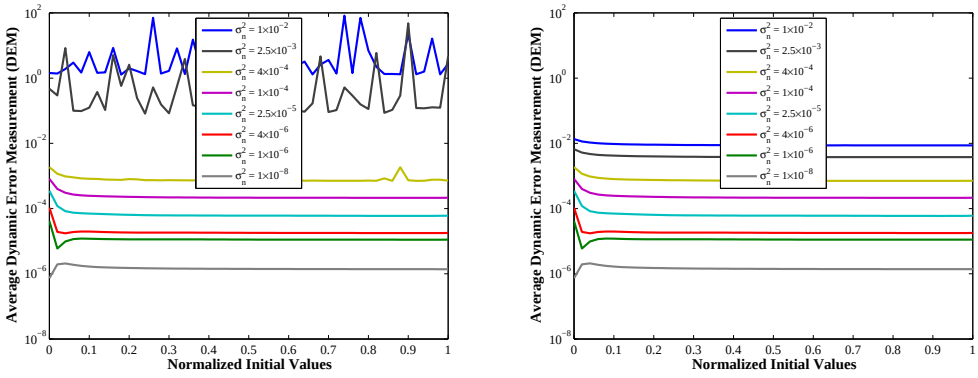
(d) $\mathbf{x}_0 = 0.6$ (e) $\mathbf{x}_0 = 0.8$ (f) $\mathbf{x}_0 = 1$

Figure 20: Propagation of error in predicted dynamic trajectories by the GPM recursive framework. The figures show the average values of LEM and DEM for 6 different dynamic trajectories at each discrete time step s, averaged over 1000 different experimental designs. The discrete time interval is $\Delta t = 10$ s and, the noise level in the stochastic observations is $\sigma_n^2 = 1 \times 10^{-8}$.

4.3.3 Propagation of error in the GPM dynamic framework

Section 2.5.2 presented two options for an uncertain input distribution in the GPM. In this section of the results, both of these approaches will be used and implemented to evaluate their performance in the GPM dynamic prediction. Figure 21 shows the average DEM values for both the Taylor-series and Gaussian approximations used to account for the input distribution in a GPM. The results in Figure 21 were made similarly to the results in Figure 19. Comparing Figures 19 and 21, both corrected GPMs using the Taylor-series approximation and the Gaussian approximation show similar features to the traditional implementations of the GPM that do not consider input uncertainty, particularly at low noise levels. The substantial difference in the figure is the dramatic effect that high noise observations have on the Taylor-series

approximations. Table 3 summarizes the average DEM values of the GPM at different noise levels and experimental designs for the mean prediction of the dynamic trajectories. The results presented on Table 3 are clear about the performance between the classical GPM mean prediction in Equation (13), and the two corrected GPM implementations. By using either the Taylor-series or the Gaussian approximation in the GPM dynamic framework, there is approximately a 0.1% difference in the performance compared to the common GPM mean prediction at low noise levels.



(a) (b)

Figure 21: Average Dynamic Error Measurement (DEM) over complete dynamic trajectories in the GPM dynamic framework, for the discrete time, $\Delta t = 10$ s. The GPM in this figure uses an equally spaced experimental design with $n = 20$ sample points. (a) corresponds to the GPM mean prediction with the Taylor-series approximation in Equation (59). (b) corresponds to the GPM mean prediction with the Gaussian approximation in Equation (70).

Perhaps the corrected GPM expressions for $\hat{y}(\mathbf{x})$ did not improve the dynamic predictions of the framework, but the corrected GPM expressions for $\sigma_y^2(\mathbf{x})$, Equations (60) and (71), may still improve the propagation of error at each discrete time step. To perform this analysis, the residual prediction error of the GPM recursive framework in a dynamic trajectory is computed at each discrete time step. This computation is different from Equation (111), where the DEM is computed with the squared value of the prediction error at each discrete time step. For the case of this one dimensional dynamic model, the residual prediction error of the GPM dynamic framework starting

Table 3: Average DEM values for the GPM dynamic framework, over 1000 different experimental designs and several initial values in the scaled input space. The table compares the classical implementation of a GPM, with its Taylor-series and Gaussian approximations for input uncertainty. These different GPM implementations are evaluated at several noise levels and experimental designs.

Noise	**Classical GPM**	
	Equally Spaced	**Random**
1×10^{-8}	1.4546×10^{-6}	4.4616×10^{-6}
1×10^{-6}	1.1724×10^{-5}	3.8123×10^{-5}
4×10^{-6}	1.9883×10^{-5}	7.6341×10^{-5}
2.5×10^{-5}	6.9356×10^{-5}	2.0787×10^{-4}
1×10^{-4}	2.3760×10^{-4}	4.7252×10^{-4}
4×10^{-4}	7.6760×10^{-4}	0.0012
2.5×10^{-3}	0.0041	0.0049
1×10^{-2}	0.0092	0.0095

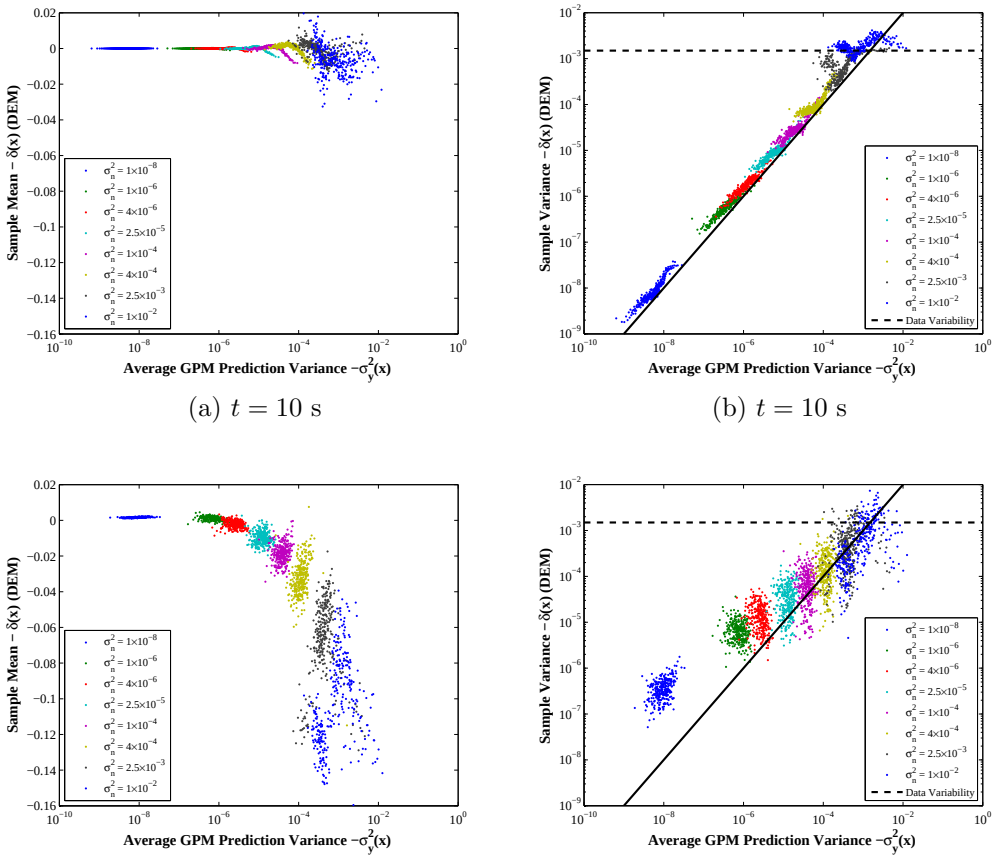
Noise	**Taylor-series approximation**	
	Equally Spaced	**Random**
1×10^{-8}	1.4549×10^{-6}	4.4609×10^{-6}
1×10^{-6}	1.1742×10^{-5}	3.8107×10^{-5}
4×10^{-6}	1.9897×10^{-5}	7.6310×10^{-5}
2.5×10^{-5}	6.9245×10^{-5}	2.0776×10^{-4}
1×10^{-4}	2.3738×10^{-4}	4.7260×10^{-4}
4×10^{-4}	8.0534×10^{-4}	0.1038
2.5×10^{-3}	2.3448	2.8058
1×10^{-2}	8.0537	75.7516

Noise	**Gaussian approximation**	
	Equally Spaced	**Random**
1×10^{-8}	1.4556×10^{-6}	4.4599×10^{-6}
1×10^{-6}	1.1760×10^{-5}	3.8099×10^{-5}
4×10^{-6}	1.9904×10^{-5}	7.6310×10^{-5}
2.5×10^{-5}	6.9182×10^{-5}	2.0774×10^{-4}
1×10^{-4}	2.3729×10^{-4}	4.7260×10^{-4}
4×10^{-4}	7.6352×10^{-4}	0.0012
2.5×10^{-3}	0.0040	0.0047
1×10^{-2}	0.0090	0.0093

at $\mathbf{x}_0$ at each discrete time step is

$$\delta\left(\mathbf{x}_0, s\right) = x(s+1) - \hat{y}\left(\hat{x}(s)\right) \tag{117}$$

Note that the error is not squared here. This modification in the DEM calculation, allows one to use the error estimation analysis of the $\delta(\mathbf{x})$ residual distribution, similar to the analysis perform in Section 3.3.3. Since the dynamic trajectories generate residual prediction errors at each discrete time step, the error estimation analysis can be performed at each time step, following closely how the prediction errors propagate. Figure 22 shows the error estimation analysis of the GPM at $t = 10$ s and $t = 200$ s using an equally spaced experimental design. In this figure, the error estimation analysis uses the common expression for the GPM prediction variance in Equation (14).



(a) $t = 10$ s (b) $t = 10$ s

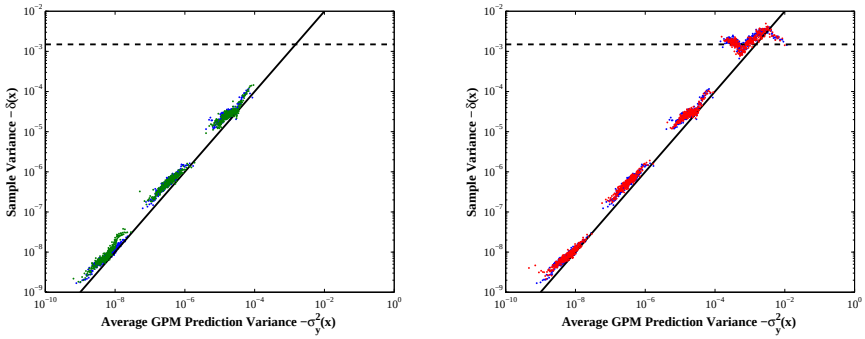
(c) $t = 200$ s (d) $t = 200$ s

Figure 22: Sample mean and sample variance for the $\delta(\mathbf{x})$ residual distribution of the GPM at different time steps in the dynamic prediction. The figure shows the correlation between the GPM residuals and the classical GPM prediction variance $\sigma_y^2(\mathbf{x})$ in Equation (14).

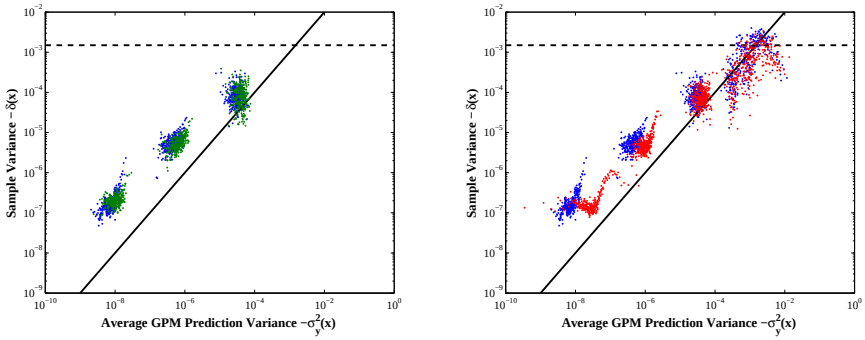
Figures 22a and 22b are relevant to this dynamic analysis, because they represent the error estimation properties of the GPM without input uncertainty, like in Chapter 3. These two figures show similar behaviors compared to the case studies in Chapter 3. On average the sample mean of the $\delta(\mathbf{x})$ residual distribution is mostly centered on zero except at the highest noise levels in the observations, and the sample variance of the residual distribution correlates well with the GPM prediction variance $\sigma_y^2(\mathbf{x})$. On the other hand, Figures 22c and 22d represent the effects of error propagation throughout the dynamic prediction. The sample mean of the residual distribution is no longer centered at zero, and the deviation from the zero mean behavior grows as the noise level in the observations increases. The sample variance of the residual distribution is no longer correlated on a 1:1 relationship with $\sigma_y^2(\mathbf{x})$, and now $\sigma_y^2(\mathbf{x})$ underestimates the sample variance of the distribution at several noise levels.

Figure 23 shows the implementation of the corrected GPM prediction variance expression from the Taylor-series and Gaussian approximation, to correlate the sample variance of the δ residual distribution at different discrete time steps. The rationale for this implementation is that when the propagation of error occurs, the corrected GPM prediction variance expressions will be able to track the uncertainty in the prediction, preserving the 1:1 relationship exhibited in the one-step-ahead prediction (Figure 21a). Based on this rationale, Figure 23 shows the error estimation analysis at later discrete time steps during the dynamic prediction.

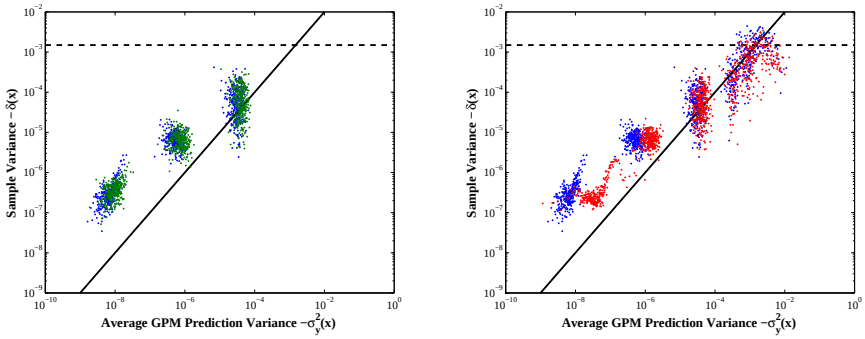
Figures 23a, 23c and 23e shows the scatter plots of the error estimation at three different discrete time steps and three noise levels $\sigma_n^2 = (1 \times 10^{-8}, 1 \times 10^{-6}, 1 \times 10^{-4})$ for the classical GPM equations and the Taylor-series approximation. These figures show that the Taylor-series approximation does not offer any improvement in tracking the propagated error during the prediction, compared to the traditional GPM expressions for error propagation. In contrast, Figures 23b, 23d and 23f shows how the implementation of the Gaussian approximation improves the correlation between



(a) $t = 10$ s (b) $t = 10$ s



(c) $t = 100$ s (d) $t = 100$ s



(e) $t = 200$ s (f) $t = 200$ s

Figure 23: Error estimation properties of the GPM dynamic framework during a dynamic prediction. Blue dots corresponds to the error estimation using the classical GPM equations. Figures (a), (c) and (e) corresponds to the Taylor-series approximation (green dots), while figures (b), (d) and (f) corresponds to the Gaussian approximation (red dots).

the sample variance of the residual distribution and the GPM prediction variance $\sigma_y^2(\mathbf{x})$ at low noise levels, since these scatter plots are closer to the 1:1 line. The figures showing the Gaussian approximation performance uses four noise levels for the

error estimation analysis, $\sigma_n^2 = (1 \times 10^{-8}, 1 \times 10^{-6}, 1 \times 10^{-4}, 1 \times 10^{-2})$. To complement this comparison, Figure 24 shows the CPU time for a GPM prediction, where the additional calculations in the Taylor-series approximation and the Gaussian approximation increases the computational cost by approximately one order of magnitude compared to the classical expressions used in the GPM. Also notice in Figure 24 that the Taylor-series approximation is less costly to compute compared to the Gaussian approximation. As a result, the Gaussian approximation improves the tracking of the propagation of error in the dynamic predictions, at expense of a larger computational cost.

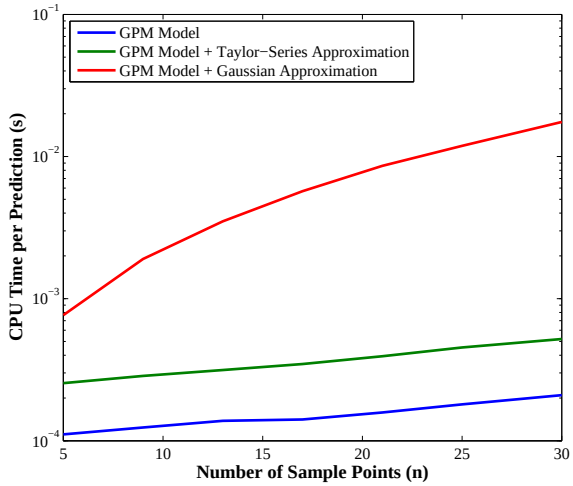
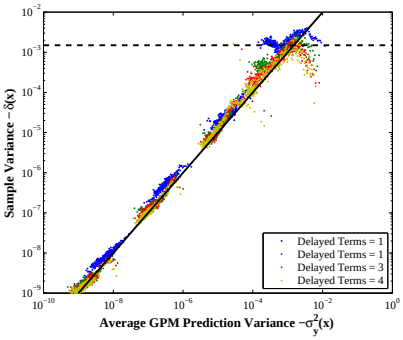


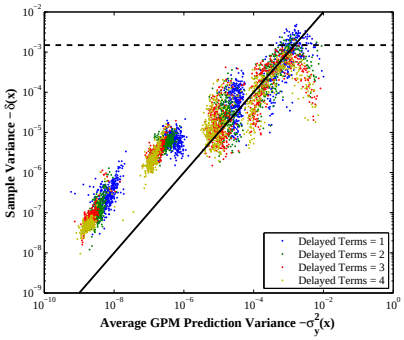
Figure 24: CPU Time per GPM input prediction in each of the three GPM implementations: Classical GPM, Taylor-series approximation and Gaussian approximation as a function of the number of sample points n in the model. The CPU time calculations were computed with a Intel®Core^{TM} 2 @ 2.4 GHz, Matlab Version R2009b.

To exploit the potential advantages of the Gaussian approximation in the propagation of error, two additional options are considered. The first of these options is to expand the input space in the GPM by incorporating more delayed terms of the system dynamics in the model in Equation (112). The Markov assumption in the GPM dynamic framework of Equation (50) limits the implementation of the recursive mapping function to one delayed term. Although this should be sufficient to represent

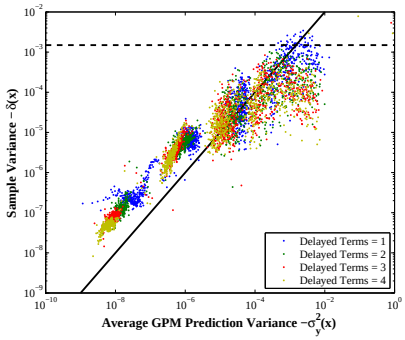
the one-state model, the prediction of this stochastic system might be improved by adding more delay terms. Figure 25 shows the implementation of multiple delayed terms in the GPM dynamic framework, comparing the classical GPM equations with the Gaussian approximation. The implementation of the Gaussian approximation requires a complete description of the state covariance matrix of the multiple delayed terms in the GPM, especially because these terms are correlated with each other. For a complete description of how to construct the covariance matrix of the uncertain input distribution when multiple delayed terms are used in the GPM dynamic framework, I refer the reader to the Reference [40].



(a) $t = 10$ s



(b) $t = 200$ s



(c) $t = 200$ s

Figure 25: Error estimation properties of the GPM dynamic framework during a dynamic prediction using multiple delayed terms at different noise levels $\sigma_n^2 = [1 \times 10^{-8}, 1 \times 10^{-6}, 1 \times 10^{-4}, 1 \times 10^{-2}]$. Figures 25b corresponds to the classical GPM equations, and Figures 25c corresponds to the Gaussian approximation.

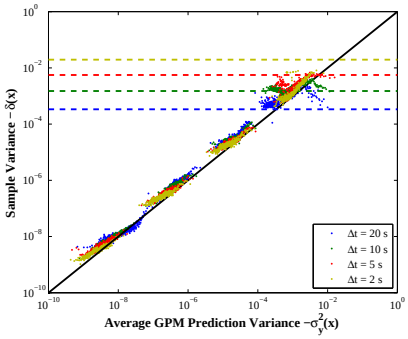
Figure 25 shows that the variance of the $\delta(\mathbf{x})$ residual distribution decreases when the number of delayed terms in the GPM dynamic framework increases. This result

means that the GPM dynamic framework improves its prediction with the additional terms in the model. Figure 25a shows this improvement on the one-step-ahead prediction error. A consequence of improving the one-step-ahead prediction error is that the propagation of error in the recursive framework decreases, as it is shown on Figures 25b and 25c. Despite the use of multiple delayed terms in the Gaussian approximation on Figure 25c, there is not significant improvement in the *correlation* between the sample variance and the corrected GPM prediction variance.

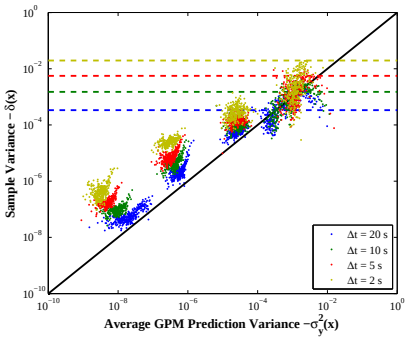
The second option to consider in the propagation of error using the Gaussian approximation is using different sampling rates to describe the system dynamics. The recursive use of the GPM has many resemblances with the implementation of iterative numerical methods like the Euler method or the Runge-Kutta method, to approximate the dynamics of a system. The efficiency in these numerical methods is associated with the step size used to discretize the process. The recursive GPM dynamic framework described in this research can not be compared directly with these numerical methods since these approaches assume that the closed-form solution of the system dynamics is known. Nevertheless, the sampling rate used in the GPM dynamic framework will have an impact in the framework, since the recursive mapping function that needs to be approximated by the GPM will change.

Figure 26 shows the implementation of four sampling rates to describe the dynamics of the second-order reaction model in Equation (112), and its effects on the propagation of error. Figure 26a shows the correlation between the variance of the $\delta(\mathbf{x})$ residual distribution and the GPM prediction variance for the different sampling rates for a one-step-ahead prediction error. This figure shows that lower one-step-ahead prediction errors can be obtained by using a smaller Δt in the construction of the GPM dynamic framework.

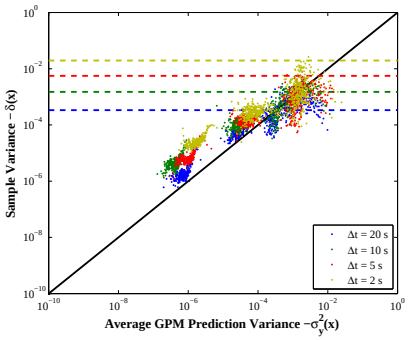
The problem with using a smaller sampling rate is the increase in the number of iterations n_s necessary to describe a dynamic trajectory. Figures 26b to 26e compares



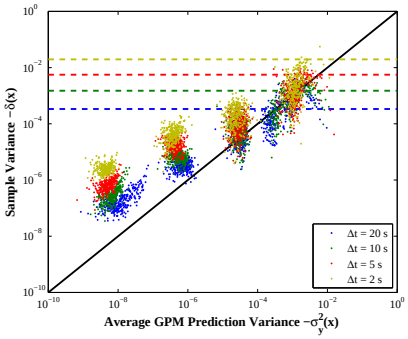
(a) One-step-ahead prediction



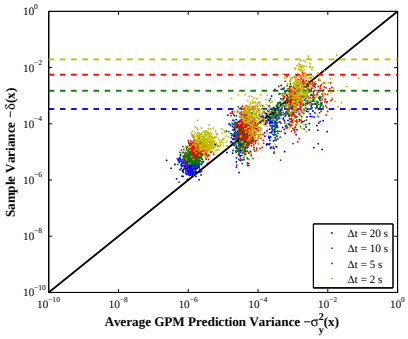
(b) $t = 60$ s



(c) $t = 60$ s



(d) $t = 200$ s



(e) $t = 200$ s

Figure 26: Error estimation properties of the GPM dynamic framework during a dynamic prediction using different sampling rates. Figure 26a shows the one-step-ahead prediction results for four different sampling rates at different noise levels $\sigma_n^2 = (1 \times 10^{-8}, 1 \times 10^{-6}, 1 \times 10^{-4}, 1 \times 10^{-2})$. Figures 26b and 26d corresponds to the classical GPM equations, while Figures 26c and 26e corresponds to the Gaussian approximation.

the propagation of error of different sampling rates at two different instances in the dynamic predictions. For similar instances in time, the smaller sampling rates propagate more error, through more iterations of the dynamic prediction, compared to the

larger sampling rates, meaning that these predictions are ultimately less accurate. These figures show that the propagation of error is a problem that is not solved by simply reducing Δt. In fact, the best results are obtained with the Gaussian approximation, by increasing Δt to 20 s. This reduces the propagation of error by decreasing the number of iterations.

4.4 Discussion

The propagation of error is an unavoidable problem in the implementation of a data-driven approximate model for system dynamics. Several researchers have been using empirical models as a recursive mapping function to describe system dynamics, but few attempt to characterize the propagated error in their predictions. The main reasons for not doing it could be:

- their empirical models do not offer a good estimation of its prediction error
- their methodologies uses large datasets that have been tailored specifically for their objectives, so there is sufficient data to characterize the system dynamics without worrying much about the propagation of error
- it is difficult to characterize the prediction error due to the large sequence of data analysis tools used in the methodology

This research is unique and distinct relative to many of these implementations because of the level of uncertainty during the construction of the data-driven *dynamic* model. The identification of the dynamics of a complex system involves an exploration of the state space, that is usually made by a careful selection of data points. Potentially, the information about the propagation of error in the data-driven dynamic model could be fed back and used in the exploration stage, improving the selection of new data points for the model.

In my opinion, Gaussian process models seem to have all the theoretical framework to estimate the propagation of error in a data-driven dynamic models. Chapter 3 described the properties and advantages of using the GPM prediction variance to correlate the error prediction in the model and, the theory about how to propagate uncertain inputs through the GPM has been developed previously. This chapter does not try to prevent the propagation of error in a GPM dynamic framework, instead it asks what the user can do with the tools and properties of the GPM to quantify this problem. In particular, this chapter goes back to the research question presented in Section 1.3: can the uncertainty in the GPM prediction be estimated? With this idea in mind, the corrected GPMs were implemented as part of the dynamic recursive framework. As a result, the modified GPM equations using the Gaussian approximation is a promising approach to account for the propagation of error in the Gaussian process model. Future studies regarding this particular issue in the GPM dynamic framework is to propose a heuristic rule, based on the GPM prediction variance that can improve the correlation with the variance of the residual distribution. This may require additional sampling near the boundaries, sampling in regions with high gradients of the output information or the use of a different correlation function. Model selection methods could be used to tailor the correlation function to the particular dataset, potentially improving the error estimation.

Despite using a simple dynamic model as a second-order reaction rate, this chapter illustrates many significant points for a GPM user in dynamic systems modeling. The first lesson learned in this chapter is the necessity of using samples at the boundaries of the input space. In the particular case of the second-order reaction model, these boundary samples not only reduce the prediction error in their neighborhood, but also contribute in the specification of the dynamic behavior of the system. The second lesson learned in this chapter is that an empirical model by itself will not satisfy the physical principles that govern a chemical system, as it was shown in the case of a

dynamic prediction for $\mathbf{x}_0 = 0$.

As important as the boundary behavior in a GPM is, the predictions of a Gaussian process model in areas with large gradients are also important. An example of a large gradient situation is when the GPM dynamic framework approximates a stiff dynamic system. Stiffness has been a recurrent topic in the mathematical modeling of multiscale dynamic systems, leading to the creation of specific numerical methods to handle it. Moreover, this chapter shows how a usual solution to model stiff systems, which is decrease the time step Δt, will not work in the recursive dynamic framework due to the large propagation of error. Therefore, the data-driven approach presented in this work has to address this issue, and analyze how to account for this situation. When the function to be estimated has areas with large positive or negative gradients, the assumption of a stationary correlation function does not represents accurately the behavior of the residuals in the GPM. One solution for this issue is to modify the structure of the correlation function into a non-stationary function, where the residuals are now related with the specific location in the input space. The second solution could be the addition of new sampled data on the regions of large gradients to improve the prediction in those regions. A possible third option to be considered is to design a time-dependent GPM dynamic framework. With this option, the database is not restricted to record dynamic data at a fixed discrete time interval, allowing some flexibility in the identification of stiff systems.

This discussion has centered the attention on the different challenges that a data-driven approach faces in the identification of dynamic systems. Most of the issues discussed here are not related with a particular empirical model, rather they are general and they should always be considered in data-driven dynamic models. The main conclusion from this chapter is that, without a good dataset to characterize the system dynamics, any empirical model will suffer of all these problems. Ultimately, the improvement of the dynamic recursive framework comes from improving

the experimental design on the one-step-ahead prediction.

CHAPTER V

RECURSIVE DYNAMIC FRAMEWORK USING MULTIVARIATE GAUSSIAN PROCESS MODELS FOR STOCHASTIC SIMULATIONS

The goal of this thesis is to implement a data-driven empirical model to describe the dynamics of stochastic systems, with the purpose of reducing the computational cost in expensive dynamic simulations. Chapter 2 provides all the tools required to build up such data-driven approach using the Gaussian process model. Chapter 3 describes the effect that stochastic data have on a Gaussian process model and defines the limitations of the model regarding noise and statistical identification of local correlations. Chapter 4 explains the challenges that a Gaussian process model faces when it is implemented as a recursive mapping function for dynamic predictions, and its consequences in the propagation of predicted errors. With these tools and lessons learned, Chapter 5 expands the implementation of Gaussian process models to predict the dynamics of systems with more than one state variable. This chapter offers a more practical description of the dynamic implementation of Gaussian process models than Chapter 4. This chapter not only explains the dynamic prediction of multiple state variables with Gaussian process models, but also describes the preliminary steps in the identification of the state space and the dataset construction for the model. For this purpose, a mathematical model describing the dynamics of a non-adiabatic continuous stirred tank reactor (CSTR) with a cooling jacket [144] is used as a case study. The reasons to selected this particular chemical process system are:

1. It is a simple deterministic mathematical model, common in the field of chemical engineering, with a well studied dynamic behavior, including multiple stable and

unstable steady states.

2. It is small enough for a complete implementation and interpretation of all elements in the dynamic framework for Gaussian process models.

3. It allows for a precise control of the additive noise in the deterministic simulations, compared to a more complex simulation like a kinetic Monte Carlo simulation.

This chapter begins with a mathematical background section explaining the construction of cross-covariance functions for multivariate Gaussian process models (mGPM), and the error estimation analysis for predictions made by a multivariate Gaussian process model, based on the one-dimensional error estimation analysis in Chapter 3. Then, this chapter follows with a brief introduction of the non-adiabatic continuous stirred tank reactor as a case study, and a step-by-step description to construct datasets for the mGPM based on pre-collected dynamic trajectories. To conclude, this chapter analyzes and discusses the implementation of mGPM and its error estimation analysis for the case study, providing conclusions on how Gaussian process models could be used for complex simulations with multiple state variables.

5.1 Background

5.1.1 Implementing mGPM for dynamic systems modeling

Section 2.6 presented a brief description of multivariate Gaussian process models. In that section, mGPM is shown as a generalized version of GPM, where the local correlation is enhanced by using additional information or observations from other variables that can be correlated simultaneously. This simultaneous local correlation between the residuals of two or more variables is made possible by the cross-covariance function $k_{ij}(\mathbf{x}_a, \mathbf{x}_b)$. Here in this chapter, this background section describes the construction of cross-covariance functions using the linear coregionalization model [45],

the implementation and parameter estimation of the linear coregionalization model for mGPM, and the usage of mGPM as a recursive mapping function to generate multivariate dynamic predictions.

Consider a set $\mathcal{D}$ of n input/output pairs $\{\mathbf{x}_i, \mathbf{y}(\mathbf{x}_i)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{y}(\mathbf{x}_i) \in \mathbb{R}^m$, $i = 1 \ldots n$. For simplicity, assume that $m = 2$ and $d = 2$. The regression covariance matrix V of a multivariate Gaussian process model corresponds to

$$V \in \mathbb{R}^{2n \times 2n} = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \tag{118}$$

where $V_{11} \in \mathbb{R}^{n \times n}$ and $V_{22} \in \mathbb{R}^{n \times n}$ are the regression covariance matrices that locally correlate the residuals within the same variable, and $V_{12} \in \mathbb{R}^{n \times n}, V_{12} = V_{21}$ is the regression cross-covariance matrix that locally correlates the residuals between different variables. Generally, to define the mGPM regression covariance matrix V, $\frac{m(m+1)}{2}$ covariance and cross-covariance matrices are needed, where m is the number of variables to be predicted. At first glance, all covariance and cross-covariance matrices could be modeled similarly as it has been done throughout this thesis work. In the case of using a Gaussian correlation function, Equation (9), the regression covariance and cross-covariance matrices V_{11}, V_{22} and V_{12} could be modeled as

$$V_{11,ij} = \sigma^2_{u,11}\delta_{ij} + \sigma^2_{c,11} \exp\left[-\frac{1}{2}\sum_{a=1}^{d}\frac{(x_{i,a} - x_{j,a})^2}{\ell^2_{a,11}}\right] \tag{119}$$

$$V_{22,ij} = \sigma^2_{u,22}\delta_{ij} + \sigma^2_{c,22} \exp\left[-\frac{1}{2}\sum_{a=1}^{d}\frac{(x_{i,a} - x_{j,a})^2}{\ell^2_{a,22}}\right] \tag{120}$$

$$V_{12,ij} = \sigma^2_{u,12}\delta_{ij} + \sigma^2_{c,12} \exp\left[-\frac{1}{2}\sum_{a=1}^{d}\frac{(x_{i,a} - x_{j,a})^2}{\ell^2_{a,12}}\right] \tag{121}$$

where $i, j = 1, \ldots, n$, δ_{ij} is the Kronecker's delta, and $\boldsymbol{\theta}_{11}$, $\boldsymbol{\theta}_{22}$, $\boldsymbol{\theta}_{12}$ are the GPM parameter sets in each of the correlation functions of the regression covariance and cross-covariance matrices in Equations (119)–(121) . For a mGPM, all parameters should be estimated simultaneously using, for example, a maximum likelihood estimator.

By inspecting the mGPM predictive distribution in Equations (89)–(98), the estimation of the mGPM parameter set must satisfy the positive semidefinite condition in the regression covariance matrix V. This condition guarantees that V can be inverted during the prediction of mGPM. At the end, the inversion of V is associated with the values of the estimated mGPM parameters and the mathematical structure of covariance and cross-covariance matrices. In the way Equations (119)–(121) are written, it is very challenging to describe a set of constraints on the mGPM parameters such that the inversion of V is guaranteed. This estimation problem is even harder when one of the variables already has a GPM parameter set, and other variables become available as additional information for local correlation. As a result of these challenges, research in geostatistics have been focusing on alternative mathematical formulations to model the parameters in the covariance and cross-covariances matrices.

A linear coregionalization model (LCM) is a simple approach to model covariance and cross-covariance matrices simultaneously in mGPM. The key assumption of the LCM is that all of the variables can be modeled with the same local correlation features acting additively at different spatial scales [45]. This assumption means, in practical terms, that all covariance and cross-covariances matrices will have the same correlation functions, but they will be weighted differently in each of the matrices. The weights in each of the matrices correspond to the estimated values of the correlated and uncorrelated variance parameters $\left(\sigma^2_{u,ij}, \sigma^2_{c,ij}\right)$. For example, in the case of Equations (119)–(121), an LCM will look like

$$V_{11,ij} = \sigma^2_{u,11}\delta_{ij} + \sigma^2_{c,11}\ \exp\left[-\frac{1}{2}\sum_{a=1}^{d}\frac{\left(x_{i,a}-x_{j,a}\right)^2}{\ell_a^2}\right] \tag{122}$$

$$V_{22,ij} = \sigma^2_{u,22}\delta_{ij} + \sigma^2_{c,22}\ \exp\left[-\frac{1}{2}\sum_{a=1}^{d}\frac{\left(x_{i,a}-x_{j,a}\right)^2}{\ell_a^2}\right] \tag{123}$$

$$V_{12,ij} = \sigma^2_{u,12}\delta_{ij} + \sigma^2_{c,12}\ \exp\left[-\frac{1}{2}\sum_{a=1}^{d}\frac{\left(x_{i,a}-x_{j,a}\right)^2}{\ell_a^2}\right] \tag{124}$$

where $\boldsymbol{\theta} = \left[\ell_1, \ell_2, \sigma^2_{u,11}, \sigma^2_{u,22}, \sigma^2_{u,12}, \sigma^2_{c,11}, \sigma^2_{c,22}, \sigma^2_{c,12}\right]$ corresponds to the mGPM parameter set when $d = 2$ and $m = 2$. Notice in Equations (122)–(124) that all the matrices have the same range parameters ℓ_a, meaning that all of them have the same local correlation features. Because of the additive characteristic of the LCM, a user could add more local correlation functions to each of the matrices, weighted by their set of correlated variance parameters. By using the LCM in a mGPM, the positive semidefinite condition of the regression covariance matrix V will depend on the estimated values of the weights $\left(\sigma^2_{u,ij}, \sigma^2_{c,ij}\right)$. To guarantee the inversion of the regression covariance matrix V in the mGPM, the uncorrelated and correlated variance parameters must satisfy the following constraints [109, 156]

$$\sigma^2_{u,ii}\ \sigma^2_{u,jj} \geq \left(\sigma^2_{u,ij}\right)^2 \quad i, j = 1, \ldots, m \tag{125}$$

$$\sigma^2_{c,ii}\ \sigma^2_{c,jj} \geq \left(\sigma^2_{c,ij}\right)^2 \quad i, j = 1, \ldots, m \tag{126}$$

where m is the number of variables to be predicted by the mGPM. These constraints guarantees that V in Equation(118) is invertible.

Once a particular set of correlation functions have been selected to model the covariance and cross-covariance matrices, an LCM can be used to model the regression covariance matrix V in Equation (118). The parameter estimation of a mGPM can be written as a nonlinear constrained optimization problem, using the likelihood function in Equation (21). For the particular case of an LCM defined by Equations (122)–(124), the mGPM parameter estimation of normalized input/output information using the

likelihood function is made as follows

$$
\begin{aligned}
&\min_{\boldsymbol{\theta}} \left[-\ln \mathcal{L}\left(\boldsymbol{\theta}|\mathcal{D}\right)\right] \\
&\quad -\ln \mathcal{L}\left(\boldsymbol{\theta}|\mathcal{D}\right) = \frac{n}{2}\ln\left(2\pi\right) + \frac{1}{2}\ln\left(|V|\right) + \frac{1}{2}\left(\mathbf{y} - H\boldsymbol{\beta}\right)^T V^{-1}\left(\mathbf{y} - H\boldsymbol{\beta}\right) \\
&\quad \text{subject to} \\
&\quad \boldsymbol{\beta}\left(\boldsymbol{\theta}\right) = \left(H^T V^{-1} H\right)^{-1} H^T V^{-1}\mathbf{y} \\
&\quad \sigma_{u,ii}^2\ \sigma_{u,jj}^2 \geq \left(\sigma_{u,ij}^2\right)^2 \quad i,j = 1,2 \\
&\quad \sigma_{c,ii}^2\ \sigma_{c,jj}^2 \geq \left(\sigma_{c,ij}^2\right)^2 \quad i,j = 1,2 \\
&\quad 1\times 10^{-9} \leq \sigma_{u,ii}^2, \sigma_{c,ii}^2 \leq 1\times 10^{-1} \quad i = 1,2 \\
&\quad -1\times 10^{-1} \leq \sigma_{u,ij}^2, \sigma_{c,ij}^2 \leq 1\times 10^{-1} \quad i \neq j, i,j = 1,2 \\
&\quad 1\times 10^{-3} \leq \ell_i \leq 1 \quad i = 1,2
\end{aligned}
\tag{127}
$$

where $\boldsymbol{\theta} = \left[\ell_1, \ell_2, \sigma_{u,11}^2, \sigma_{u,22}^2, \sigma_{u,12}^2, \sigma_{c,11}^2, \sigma_{c,22}^2, \sigma_{c,12}^2\right]$, and $\mathbf{y}$, H and V are defined by Equations (92), (93) and (98) respectively. The upper and lower bounds of the mGPM parameters in Equation (127) correspond to the case where the observations $\mathbf{y}$ have been scaled previously, as it was done in Section 3.3.2. This is done to avoid potential numerical problems regarding the different orders of magnitude between the predicted variables.

Similar to the GPM in Chapter 4, mGPM can be used as a recursive mapping function to describe system dynamics. The mGPM multivariate predictive distribution in Equation (89) can be used to map information from the current vector state $\mathbf{x}(s)$ at the discrete time step s, to the next vector state $\mathbf{x}(s+1)$. According to this description, a one-step-ahead predictive distribution using mGPM can be written as

$$
\left(\mathbf{x}(s+1)|\mathbf{x}(s), \mathcal{D}\right) \sim \mathcal{N}\left(\hat{\mathbf{x}}(s+1), S\right) \tag{128}
$$

$$
\hat{\mathbf{x}}(s+1) = \hat{\mathbf{y}}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) \tag{129}
$$

$$
S = \Sigma_y^2\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) \tag{130}
$$

where $\hat{\mathbf{y}}\left(\mathbf{x}(s), \mathcal{D}\right)$ and $\Sigma_y^2\left(\mathbf{x}(s), \mathcal{D}\right)$ are defined by Equations (90) and (91) respectively.

Previously in Chapter 2, Equations (53) – (55) presented a simpler implementation of GPM for multivariate dynamic predictions,

$$
\begin{aligned}
(\mathbf{x}(s+1)|\mathbf{x}(s), \mathcal{D}) &\sim \mathcal{N}\left(\hat{\mathbf{x}}(s+1), S\right) \\
\hat{x}_i(s+1) &= \hat{y}_i(\hat{\mathbf{x}}(s), \mathcal{D}) \qquad i = 1, \ldots, m \\
S_{ij} &= \begin{cases} \sigma_{y,i}^2\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) & \text{if } i = j, \quad i, j = 1, \ldots m \\ \\ 0 & \text{if } i \neq j \end{cases}
\end{aligned}
$$

based on the independent construction of m GPMs, one for each of the predicted state variables. Hernandez et al. [56] used this simpler implementation of GPM for stochastic dynamic predictions, but it has not been compared against the mGPM dynamic implementation.

5.1.2 Error estimation framework for multivariate Gaussian process models

In previous chapters, the error estimation properties of GPM have been described and analyzed for two scenarios. Chapter 3 discusses how the GPM error estimation properties respond when stochastic observations are used in the dataset $\mathcal{D}$. In Chapter 4, the error estimation properties of GPM are applied to the field of dynamic systems modeling, to address the challenge of estimating the propagation of error in recursive mapping functions. This chapter concludes the implementation of the GPM error estimation properties, by expanding their interpretation to multivariate stochastic dynamic predictions based on the mGPM predictive distribution.

To build up the error estimation properties of a mGPM, it is necessary to define the multivariate prediction error distribution of the model. The mGPM prediction error distribution and its error estimation analysis, are made following the same concepts for the GPM prediction error distribution back in Chapter 3. As a reminder, the

GPM prediction error $\delta(\mathbf{x})$ is the difference between the GPM mean prediction $\hat{y}(\mathbf{x})$ and the true value of the function $y_{tr}(\mathbf{x})$

$$\delta(\mathbf{x}) = y_{tr}(\mathbf{x}) - \hat{y}(\mathbf{x})$$

Consider the prediction made by a mGPM predictive distribution in Equations (89)–(91). By analogy with the definition of $\delta(\mathbf{x})$, the mGPM mean prediction error $\boldsymbol{\delta}(\mathbf{x}) \in \mathbb{R}^m$ is defined as

$$\boldsymbol{\delta}(\mathbf{x}) = \mathbf{y}_{tr}(\mathbf{x}) - \hat{\mathbf{y}}(\mathbf{x}) \tag{131}$$

In the one-dimensional case, $\delta(\mathbf{x})$ is defined as a prediction error between the GPM mean prediction and the true value of the function. However, due to the uncertainties associated with the observations in the dataset $\mathcal{D}$, such as sample points locations and stochastic observations, the GPM mean prediction $\hat{y}(\mathbf{x})$ is treated as a random variable, therefore $\delta(\mathbf{x})$ is also treated as a random variable. In Chapter 3, the error estimation analysis characterized the GPM prediction error distribution using the GPM prediction variance $\sigma_y^2(\mathbf{x})$. Using $\sigma_y^2(\mathbf{x})$, it was possible to estimate the variance of the predicted error, under the assumption that the GPM prediction error distribution follows a Gaussian distribution. Using the same principle, the mGPM prediction covariance $\Sigma_y^2(\mathbf{x})$ can be used to described the uncertainty in the multivariate mGPM prediction error distribution of $\boldsymbol{\delta}(\mathbf{x})$.

Since $\boldsymbol{\delta}(\mathbf{x})$ is a vector of predicted errors in each of the m predicted variables, the mGPM prediction error distribution is modeled as a multivariate Gaussian distribution

$$\boldsymbol{\delta}(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, D(\mathbf{x})) \tag{132}$$

where $\mathbf{0}$ is the mean of the multivariate Gaussian distribution based on the unbiased constraint of the mGPM, and $D \in \mathbb{R}^{m \times m}$ is the covariance matrix of the multivariate distribution. The uncertainty in the mGPM prediction error distribution is described by the covariance matrix D. By analogy with the one-dimensional error analysis

for GPM, the covariance matrix $D(\mathbf{x})$ should be similar to the mGPM prediction covariance matrix $\Sigma_y^2(\mathbf{x})$. To establish an error estimation analysis for mGPM, it is necessary to compare how similar these two matrices are. The comparison between the matrices must be easy to understand, and it has to account all the elements in them. A simpler approach could be to compare diagonal elements in both matrices, one by one, as if they were independent one-dimensional GPMs, but it would not consider the effects of cross-covariance matrices on the off diagonal terms in each of the matrices.

The determinant of a covariance matrix is frequently used as a metric to characterize information of multivariate distributions. In the case of the multivariate Gaussian distribution, the determinant of the covariance matrix is part of the normalization constant of the probability density distribution. The determinant of a covariance matrix has also served as an indicator of uncertainty for parameter estimation, as it is the case of D-optimal designs, where maximizing the determinant of the design matrix implies minimizing the covariance of the parameter estimates [19]. In addition to these aspects, the mathematical computation of a determinant considers all elements in the covariance matrix. Based on these ideas, the error estimation analysis of the mGPM will be based on the comparison between $|D(\mathbf{x})|$, the determinant of the mGPM prediction error distribution, and $\left|\Sigma_y^2(\mathbf{x})\right|$, the determinant of the mGPM prediction covariance matrix. This means that these two matrices are similar if their determinants are also similar.

By using a scalar like the determinant of a covariance matrix, it is possible to make error estimation plots like in Chapter 4, for multivariate stochastic dynamic predictions. Based on the recursive mapping description on Equations (128) – (130), the error estimation analysis for mGPM at different discrete time steps compares $|D(\mathbf{x}(s))|$ and $|S(\mathbf{x}(s))|$. Figure 27 is a brief description of how the error estimation analysis for mGPM is made, based on $|D|$ and $|S|$. Figure 27a shows a scatter

plot made from $\boldsymbol{\delta}(\mathbf{x})$ prediction error vectors of 790 different test sample points. The mGPM prediction error vectors are estimated at each test sample point, using 10 different mGPM models built from different datasets of stochastic observations. The noise level in the stochastic observations is $\sigma_n^2 = 1 \times 10^{-8}$. Figure 27a shows a strong negative linear correlation between the prediction error of the normalized concentration, and the prediction error of the normalized temperature for the case study in this chapter. The implementation of mGPM in this case study, not only considers the identification of this negative correlation, but also its effect in the error estimation analysis via $|S|$.

Each of the prediction error vectors has a mGPM prediction covariance $S(\mathbf{x})$ associated with its prediction. Figure 27b shows on a color bar the different magnitudes of $\log_{10}|S|$ for each of the prediction error vectors. Notice that prediction error vectors which are closer to the origin have smaller values of $|S|$. This means that accurate mGPM predictions have smaller $|S|$.

The next step to characterize the multivariate prediction error distribution is to sort all the prediction error vectors in ascending order, according to their $|S|$ values. The sorted list of 7900 prediction error vectors is then divided into groups with 790 prediction error elements each. Figure 27c shows a scatter plot of prediction error vectors for one of these groups. The assumption in the error estimation analysis is that mGPM predictions with similar $|S|$ covariance matrix, will also have a similar $|D|$ prediction error covariance matrix. Therefore the prediction errors in each of these groups can be used altogether to estimate D, regardless of the location of the sample points where the prediction was obtained. In this sense, each group represent different levels of uncertainty and/or accuracy in the mGPM prediction. To characterized the multivariate $\boldsymbol{\delta}$ prediction error distribution in each group, a multivariate Gaussian distribution is fitted using all the prediction error vectors in each group. The estimated mean of the multivariate Gaussian distribution represents, on

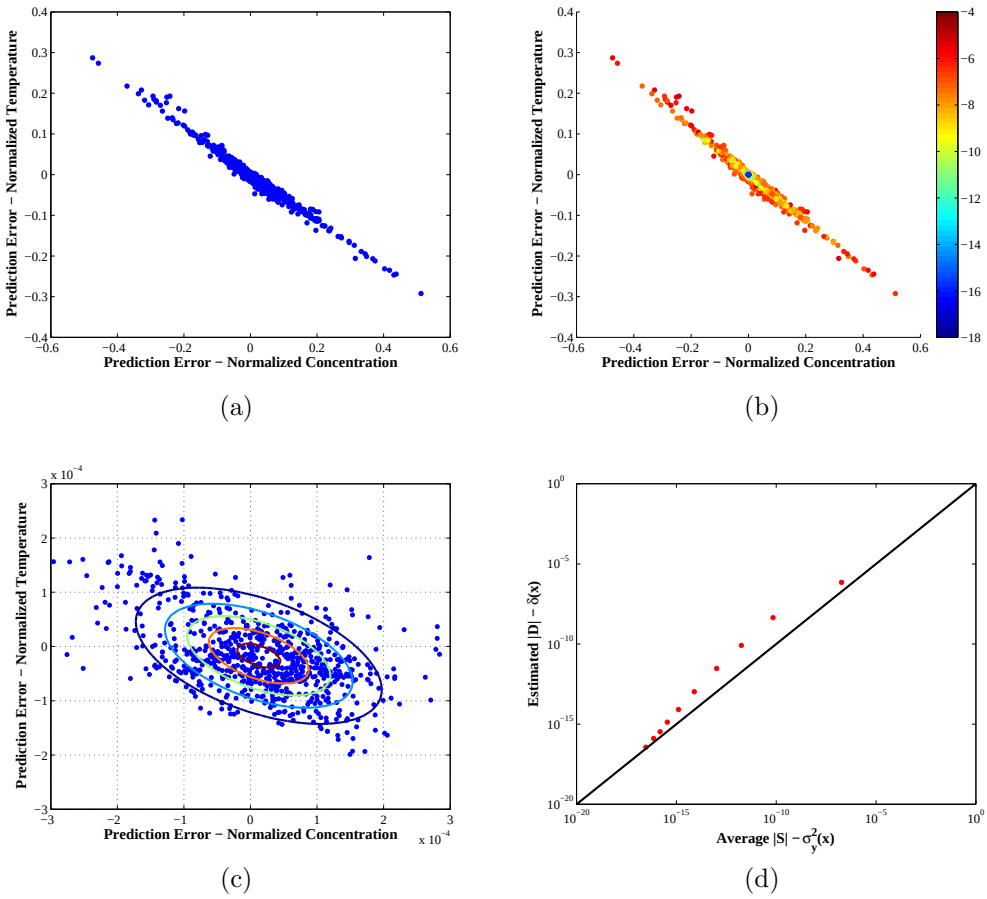


Figure 27: Error estimation analysis for multivariate Gaussian process models. Case study: Non-adiabatic CSTR (Section 5.2). (a) Scatter plots of prediction errors in a multivariate GPM. (b) Ranking of prediction errors. The color bar represents the magnitude of $\log_{10}(|S|)$ in each of the predictions made by the mGPM. (c) Estimation of multivariate Gaussian distributions for a set of prediction error vectors according to their $\log_{10}(|S|)$. The ellipses represent automatic contour levels of the estimated multivariate $\boldsymbol{\delta}(\mathbf{x})$ Gaussian distribution. (d) Comparison between the estimated $|D|$ of the multivariate Gaussian distribution, and the average $|S|$ in each of the sets of prediction error vectors.

average, the mGPM mean prediction error in the group, and the estimated covariance matrix represents an approximation to the covariance matrix of the multivariate mGPM prediction error distribution D. As an example, the estimated multivariate

$\boldsymbol{\delta}(\mathbf{x})$ Gaussian distribution for the prediction error vectors in Figure 27c is

$$\boldsymbol{\delta} \sim \mathcal{N}\left(\begin{bmatrix} 0.1251 \times 10^{-4} \\ -0.1710 \times 10^{-4} \end{bmatrix}, \begin{bmatrix} 0.1025 \times 10^{-7} & -0.0349 \times 10^{-7} \\ -0.0349 \times 10^{-7} & 0.0473 \times 10^{-7} \end{bmatrix} \right)$$

Finally, the error estimation analysis of mGPM concludes with the comparison between the determinant of the estimated covariance matrix in each of the groups, against the average value of $|S|$ in each group. Figure 27d shows an example of this comparison. The closer this line is to the 1:1 line in the figure, the mGPM has better chances to estimate its prediction error. To provide some context of this comparison, the mGPM predicted covariance matrix S of one of the test points in Figure 27c is

$$S = \begin{bmatrix} 0.1377 \times 10^{-7} & -0.0193 \times 10^{-7} \\ -0.0193 \times 10^{-7} & 0.0243 \times 10^{-7} \end{bmatrix}$$

which is close to the estimated covariance matrix in $\boldsymbol{\delta}$. This whole procedure can be repeated with a larger number of mGPM, at different noise levels in the stochastic observations, and at different discrete time step, for a complete error estimation analysis of mGPM in dynamic predictions.

5.2 Case study for dynamic GPM prediction of multiple variables: Non-adiabatic CSTR

The GPM dynamic framework described in Section 2.5 is applied to the dynamics of a non-adiabatic continuous stirred tank reactor. Figure 28 shows a schematic of the CSTR with a cooling system that removes the heat generated from the reaction.

The CSTR system shown in Figure 28 illustrates the dynamics of a first-order reaction inside the reactor. The process is defined by a system of differential equations,

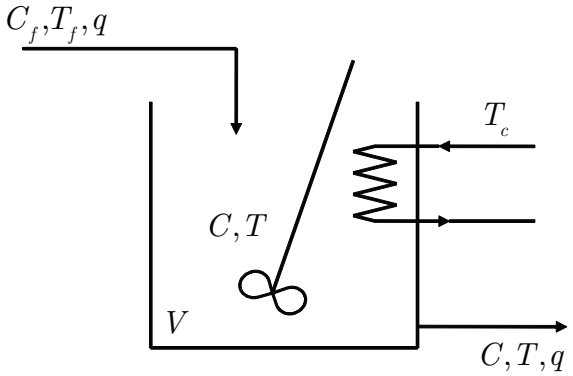


Figure 28: A continuous stirred tank reactor with cooling system

based on mass and energy balances around the reactor [144].

$$V\frac{dC}{dt} = q\left(C_f - C\right) - Vf(C,T) \tag{133}$$

$$VC_p\rho\frac{dT}{dt} = qC_p\rho(T_f - T) - UA(T - T_c) + (-\Delta H)\,Vf(C,T) \tag{134}$$

$$f(C,T) = k_0 C \exp\left[\frac{-E}{R_g T}\right] \tag{135}$$

$$0 \leq t \leq t_f \quad t_f = 500 \text{ s}$$

$$C = C_0, T = T_0 \quad \text{at } t = 0$$

where C is the reactant concentration in [mol/m^3], T is the reactor temperature in Kelvin [K], t is time in seconds [s] and $f(C,T)$ is the reaction rate in $\left[\frac{\text{mol}}{\text{m}^3\ \text{s}}\right]$. The remaining variables in Equations (133)–(135) represent physical properties of the reactants and design settings of the CSTR. These parameters are summarized and described in Table 4.

Using different combinations of the values in the Table 4, different dynamic behaviors can be observed, particularly in the number of steady states and their stability. In addition to Equations (133)–(135), a set of initial values of C_0, T_0 is also specified. The set of initial values for $[C_0, T_0]$ is defined as:

$$0\ \frac{\text{mol}}{\text{m}^3} \leq C_0 \leq 8000\ \frac{\text{mol}}{\text{m}^3} \qquad 300\text{ K} \leq T_0 \leq 450\text{ K} \tag{136}$$

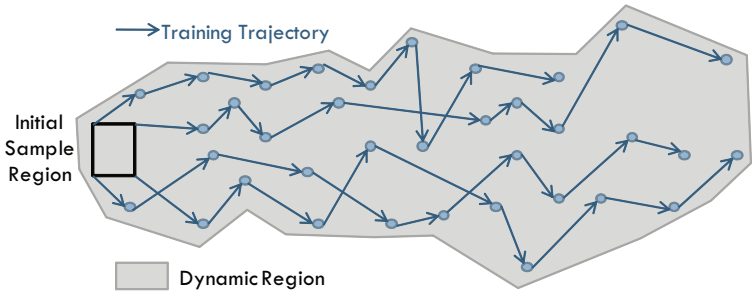
Table 4: Summary of physical properties and design settings used in the non-adiabatic CSTR case study.

Variable	Description	Units	Value
C_f	Feed concentration	$\mathrm{mol/m^3}$	8000
T_f	Feed temperature	K	300
A	Heat transfer area	$\mathrm{m^2}$	0.1
C_p	Heat capacity of reaction mixture	$\mathrm{J/\,(kg\ K)}$	4181.3
q	Volumetric flow rate	$\mathrm{m^3/s}$	1
T_c	Coolant temperature	K	300
U	Heat transfer coefficient	$\mathrm{J/\,(m^2\ s\ K)}$	2.5×10^6
V	Reactor volume	$\mathrm{m^3}$	60
ΔH	Heat of reaction	J/mol	-70000
ρ	Density of reaction mixture	$\mathrm{kg/m^3}$	1000
k_0	Pre-exponential in Arrhenius equation	1/s	2.5×10^5
E	Activation energy	J/mol	-50000
R_g	Universal gas constant	$\mathrm{J/\,(mol\ K)}$	8.314

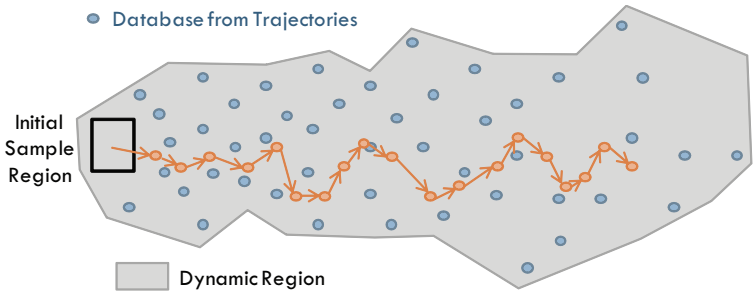
5.2.1 Collecting dynamic trajectories

The construction of the GPM dynamic framework starts with an exploration step to delineate the sampling region used to build the empirical model. The exploration of the dynamic region is made by selecting a potential set of initial conditions and running dynamic trajectories from each of them. These dynamic trajectories describe the reachable area of the system dynamics of interest. The description of the reachable area has direct implications for the GPM, since this area is the input space used by the GPM for its dynamic prediction. To simplify the description, the set of initial conditions is called the *initial sample region* and the reachable dynamic area is called the *dynamic region*. Figure 29a is a graphical representation of these two concepts. For the case study described here, the initial region is defined by the set of initial conditions in Equation (136).

The objective with the GPM dynamic framework is to provide a good prediction of any dynamic trajectory that starts in the initial sample region. In some specific applications like process control, the interest could be the approximation of a particular

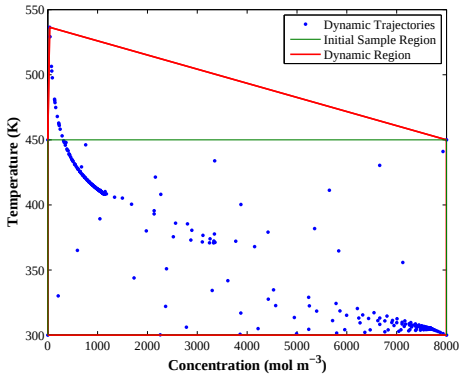


(a)

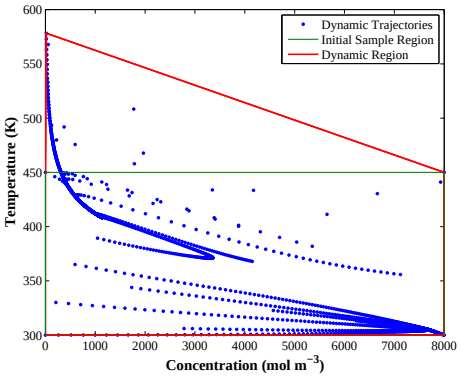


$(\Delta t) =$ Data density and shape of dynamic region

(b)



(c) n = 520 points



(d) n = 5020 points

Figure 29: Collecting dynamic trajectories for GPM dynamic framework. (a) Representation of the initial and dynamic region concepts. (b) Relationship between the sampling time Δt and the dynamic region. Figures (c) and (d) shows the initial and dynamic regions for the non-adiabatic CSTR case study using 20 dynamic trajectories and sampling times of $\Delta t = 20$ s and $\Delta t = 2$ s respectively. The description of the initial sample region is defined by Equation (136).

dynamic trajectory to be tracked, but here the aim with the dynamic implementation of GPM is a complete identification and prediction over the reachable region from the initial sample region. Therefore, the description of the dynamic region depends on

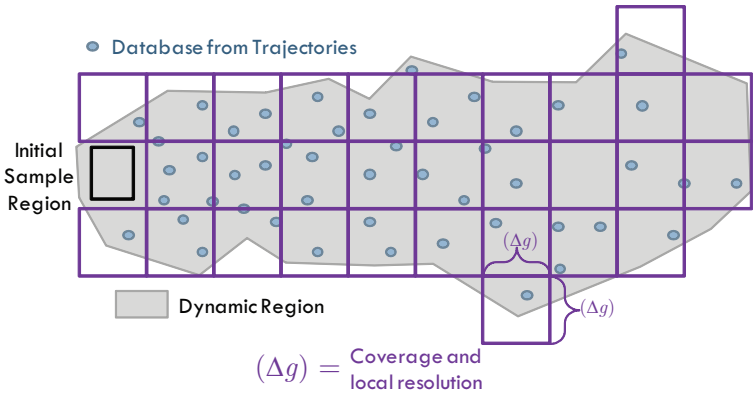
how well explored the initial sample region is. For this case study, where the initial sample region is square, a space-filling Latin Hypercube design is used to explore the region and to select a finite number of dynamic trajectories. In addition to 16 interior points from the Latin Hypercube design, 4 additional dynamic trajectories are added at the corners of the region to ensure sampling at the boundaries. This yields a total of 20 dynamic trajectories.

There are two additional elements that have an effect on the dynamic region (see Figure 29b). One is the final time of the dynamic trajectory t_f, and it is important because it defines the boundaries of the dynamic area. However, for this particular case study, the final time has been selected long enough so the dynamic trajectories could reach the steady states. The sampling rate Δt is associated with the shape and data density of the information inside the dynamic region. A shorter Δt means a larger number of points in the dynamic trajectory, providing more information to be used in the GPM. Figures 29c and 29d show the effects of different values of Δt on the dynamic region in the case study, for a final time $t_f = 500$ s. To visualize the changes in the shape of the dynamic regions as a function of the sampling rate, convex hulls are drawn to enveloped all 20 dynamic trajectories. Using these convex hulls, it is easier to see the change in the shape of the dynamic regions near $T = 550$ K. It is also easy to see the significant difference in data density by decreasing the sampling rate from $\Delta t = 20$ s to $\Delta t = 2$ s.

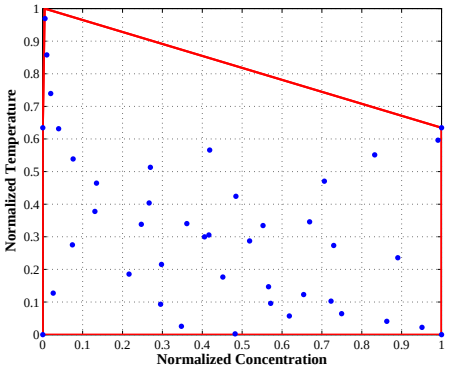
5.2.2 Creation of a database for Gaussian process models

Due to the known problems that GPM faces with a large number of sample points, a sparsification procedure is implemented to reduce the number of sample points in the model. If many points are included that are highly correlated, the covariance matrix will be ill-conditioned, and the redundant points will provide little new information. The sparsification is made not only to reduce of the number of sample points, but

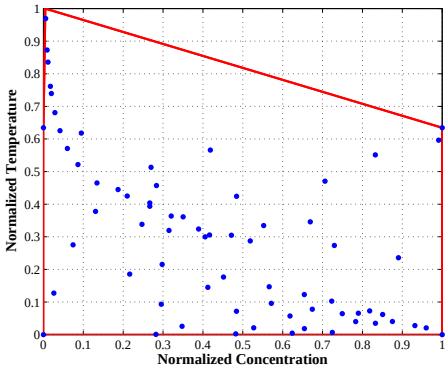
also to preserve the coverage of the information in the dynamic region. To achieve this task, the sample points from the dynamic trajectories are clustered based on a square lattice grid of size Δg in all dimensions of the input space. Then, from each of the occupied lattice sites, one sample point is chosen to be included in the database $\mathcal{D}$ used in the GPM. When two or more sample points are found in an occupied lattice site, the selected sample point is the one closer to the centroid of all sample points in the lattice. Figure 30 shows the implementation of the described sparsification approach for the case study.



(a)



(b) n = 43 points



(c) n = 70 points

Figure 30: Sparsification to create a database of dynamic information. (a) Representation of the sparsification procedure. Figures (b) and (c) shows the implementation of the sparsification on 20 dynamic trajectories collected at $\Delta t = 20$ s, using as grid sizes $\Delta g = 0.1$ and $\Delta g = 0.05$ respectively.

In order to implement a unique Δg grid size value over the dynamic trajectories, a

scaling procedure was performed over the collected data. The scaling is similar to the one used in Section 3.3.1, making all collected information be between 0 and 1. The scaling is also important to reduce effects of orders of magnitude between the different predicted variables in the framework. After rescaling and sparsification, all selected sample points are collected as input points $\mathbf{x}_i$ for the GPM, and their corresponding scaled one-step-ahead predictions $\mathbf{x}_i(s+1)$ are the output information $\mathbf{y}_i$ in the set $\mathcal{D}$. For the non-adiabatic continuous stirred tank reactor case study, $\mathbf{x}_i \in \mathbb{R}^2$ and $\mathbf{y}_i \in \mathbb{R}^2$, such that $d = 2$ and $m = 2$.

5.3 Analysis of a GPM dynamic framework under stochastic simulations

This section describes the analysis used to test the GPM dynamic framework when stochastic simulations are included in the set $\mathcal{D}$. To do this analysis the scaled output information $\mathbf{y}_i$ is corrupted with a white constant noise $\mathcal{N}(0, \sigma_n^2)$ to both the scaled concentration and scaled temperature data at different noise levels σ_n^2. This manipulation will test the robustness of the GPM dynamic model to predict the mean response of the system using noisy information. Following the signal-to-noise ratio of 1×10^{-1} recommended for the implementation of GPM models in Chapter 3, the evaluated noise levels are:

$$\begin{aligned} \sigma_n^2 \;=\; & \big\{1 \times 10^{-8}, 1 \times 10^{-6}, 4 \times 10^{-6}, 2.5 \times 10^{-5}, \\ & 1 \times 10^{-4}, 4 \times 10^{-4}, 2.5 \times 10^{-3}, 1 \times 10^{-2}\big\} \end{aligned} \tag{137}$$

The analysis of the GPM dynamic framework is constructed as follows: 20 dynamic trajectories are selected over the initial sample region as described in Section 5.2.1. The dynamic trajectories are collected over a time frame $t_f = 500$ s using $\Delta t = 20$ s as a sampling rate. The set of dynamic trajectories are scaled and sparsified using the procedures in Section 5.2.2 with a constant grid size $\Delta g = 0.05$. To the scaled output information in $\mathcal{D}$, a white noise normally distributed zero-mean random

number with variance σ_n^2 is added, using the values in Equation (137) to create the stochastic simulations for the GPM dynamic framework. Similar to the second-order reaction model in Chapter 4, the scaling of the input and output information uses the same constant values, since they belong to the same dynamic region. Also, to prevent negative concentrations in the dataset, a consistency check was added in the generation of data, like the one used in the second-order reaction model in Chapter 4.

Based on the input-output information in the set $\mathcal{D}$, and the mathematical background explained in Section 5.1.1, a GPM dynamic model can be written in two different ways. One of the approaches builds the multivariate dynamic model based on m GPMs, one for each of the m predicted variables. For now on, this approach is refer as *independent* GPMs or just iGPM. A iGPM approach to approximate the dynamics of the non-adiabatic CSTR system in Section 5.2, is described as:

$$
\begin{gathered}
\hat{\mathbf{x}}(s) = [\hat{x}_C(s), \hat{x}_T(s)] \\
(\mathbf{x}(s+1)|\mathbf{x}(s), \mathcal{D}) \sim \mathcal{N}\left(\hat{\mathbf{x}}(s+1), S\right) \\
\hat{\mathbf{x}}(s+1) = \begin{bmatrix} \hat{x}_C(s+1) \\ \hat{x}_T(s+1) \end{bmatrix} = \begin{bmatrix} \hat{y}_C\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) \\ \hat{y}_T\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) \end{bmatrix} \\
S = \begin{bmatrix} \sigma^2_{y,CC}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) & 0 \\ 0 & \sigma^2_{y,TT}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) \end{bmatrix} \\
s = 0, 1, \ldots, n_s \\
t = s\Delta t
\end{gathered}
\tag{138}
$$

where the subscripts C, T correspond to the predicted normalized concentration and normalized temperature by independent GPM predictive distributions in Equation (15), and n_s is the total number of discrete time steps required for the prediction of a dynamic trajectory. Based on the ratio between the final time $t_f = 500$ s and the sampling rate $\Delta t = 20$ s used in the database $\mathcal{D}$, $n_s = \frac{t_f}{\Delta t} = 25$.

The second approach is using the mGPM as a recursive mapping function to describe the dynamics of the non-adiabatic CSTR in the case study. Based on Equations (128)–(130), this GPM dynamic implementation is as follows

$$\hat{\mathbf{x}}(s) = [\hat{x}_C(s), \hat{x}_T(s)]$$

$$(\mathbf{x}(s+1)|\mathbf{x}(s), \mathcal{D}) \sim \mathcal{N}\left(\hat{\mathbf{x}}(s+1), S\right)$$

$$\hat{\mathbf{x}}(s+1) = \begin{bmatrix} \hat{x}_C(s+1) \\ \hat{x}_T(s+1) \end{bmatrix} = \hat{\mathbf{y}}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) \tag{139}$$

$$S = \begin{bmatrix} \sigma^2_{y,CC}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) & \sigma^2_{y,CT}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) \\ \sigma^2_{y,TC}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) & \sigma^2_{y,TT}\left(\hat{\mathbf{x}}(s), \mathcal{D}\right) \end{bmatrix} = \Sigma^2_y\left(\hat{\mathbf{x}}(s), \mathcal{D}\right)$$

$$s = 0, 1, \ldots, n_s$$

$$t = s\Delta t$$

Figure 31 is a graphical representation of the dynamic prediction made by either of the GPM dynamic frameworks. Each of the GPMs in the iGPM uses a Gaussian correlation function in Equation (9) to describe the local correlation in each variable, whereas the mGPM uses the LCM in Equations (122)–(124). The parameter estimation methodology for both of the GPM dynamic frameworks is a maximum likelihood estimator (MLE). In the case of the iGPM, m MLE constrained optimizations are computed to complete the parameter estimation of the model, compared to a single MLE constrained optimization for mGPM. Despite the differences between iGPM and mGPM for this case study, both of them have the same number of GPM parameters to be estimated.

The analysis section of this chapter begins with a discussion regarding the use of regression functions in GPM. Up to this point, the selection of regression functions have been left out of the examples, limiting all GPM implementations to used the constant regression function $\mathbf{h} = 1$. Here this additional factor is included as part of the dynamic implementation of GPM. The performance of the constant regression

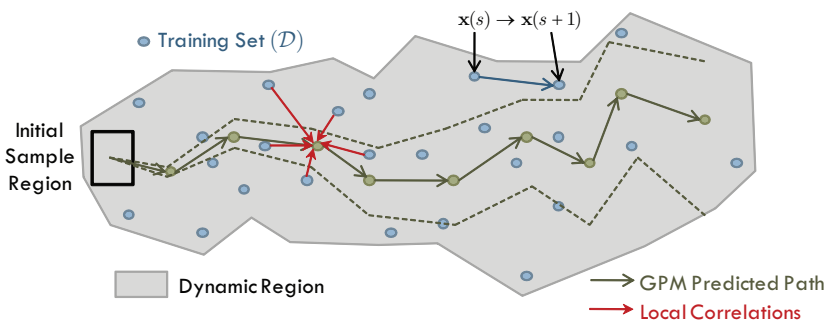


Figure 31: Graphical representation of the GPM dynamic framework.

function is compared against two polynomial subsets of regression functions. The polynomial subsets used as regression functions $\mathbf{h}(\mathbf{x})$ are:

Linear regression function

$$\mathbf{h}(\mathbf{x}) = [1, x_C, x_T] \tag{140}$$

Quadratic regression function

$$\mathbf{h}(\mathbf{x}) = \left[1, x_C, x_T, x_C x_T, x_C^2, x_T^2\right] \tag{141}$$

The analysis of regression functions is made using the iGPM dynamic implementation in Equation (138). For this analysis, both GPMs use the same regression function, excluding the possibility of a GPM using a constant regression function and the other GPM using a quadratic regression function. However, since the constant regression function is a special case of the quadratic regression functions, this is not particularly restrictive. Table 5 summarizes the 24 different combinations, between the noise level in the observations and regression functions, that are evaluated in the analysis of the regression functions. To ensure a robust analysis of the regression functions, each of the 24 combinations is repeated 1000 times, each time with a different experimental design of 20 dynamic trajectories drawn from the initial sample region.

The analysis discusses the effect of regression functions in the estimated parameters of the iGPM, and on the iGPM dynamic predictions. The accuracy of the

Table 5: Analysis of regression functions for the non-adiabatic CSTR case study, using a iGPM dynamic implementation.

Analysis Topic	**Description**
Noise levels σ_n^2	Eight different values, Equation (137)
Regression functions $\mathbf{h}(\mathbf{x})$	Constant, Linear, Quadratic

iGPM is measured by comparing its dynamic prediction with the numerical solution of the ODE model in Equations (133)–(135). As it was with the one-dimensional case study in Chapter 4, LEM (Equation (110)) and DEM (Equation (111)) are used to quantify the error in the predictions. The test dataset used to evaluate the LEM across the dynamic region is generated from $n_{dyn} = 200$ dynamic trajectories starting in the initial sample region, Equation (136), whose data has been normalized and sparsified using a value of $\Delta g = 0.01$. These trajectories were selected using a Latin hypercube design. Each of these dynamic trajectories is recorded with $\Delta t = 20$ s and $t_f = 500$ s. After the sparsification, the test dataset for LEM has 783 sample points, and the reported LEM value is the average LEM over these test points. The test dataset used to compute the DEM uses the same $n_{dyn} = 200$ dynamic trajectories that were previously used for the LEM test dataset. For each initial value $\mathbf{x}_0$ in the test dataset, a dynamic trajectory is computed using the ODE system of equations and then compared with its iGPM dynamic trajectory prediction. The final reported DEM value is the average prediction error over all n_s discrete time steps in a trajectory, and over all 200 dynamic trajectories. Remember that all these computations are repeated 1000 times, for each of the 24 combinations of noise level and regression functions in Table 5. The goal with this analysis of regression functions is to understand the role of the regression functions, for selection in future implementations of the GPM dynamic framework.

Once a regression function has been selected for the iGPM dynamic framework, based on the results of the previous analysis, the error estimation methodology described in Section 5.1.2 is performed. The prediction error vectors used in this iGPM

error estimation analysis are the ones that previously were used in the analysis of regression functions. For example, in the case of a noise level $\sigma_n^2 = 1 \times 10^{-8}$, there are 783×1000 prediction error vectors that can be used to describe the error estimation properties of the iGPM, for a one-step-ahead prediction. Similarly, at each discrete time step s, there are 200×1000 prediction error vectors that can be used to describe the error estimation and propagation of error in the iGPM dynamic framework. Section 5.4.3 in this chapter is the comparison between GPM implementations for multivariate dynamic predictions. Using the selected regression function from the analysis, a mGPM dynamic framework in Equation (139) is built. Then, the entire study that was described in the previous paragraphs for the iGPM dynamic framework is repeated for the mGPM dynamic framework. These research studies will allow a comparison between iGPM and mGPM dynamic frameworks, including analysis of the estimated parameters for both models, LEM and DEM calculations for mean prediction, and the error estimation analysis in Section 5.1.2.

5.4 Results

5.4.1 Effects of regression functions in the iGPM dynamic framework

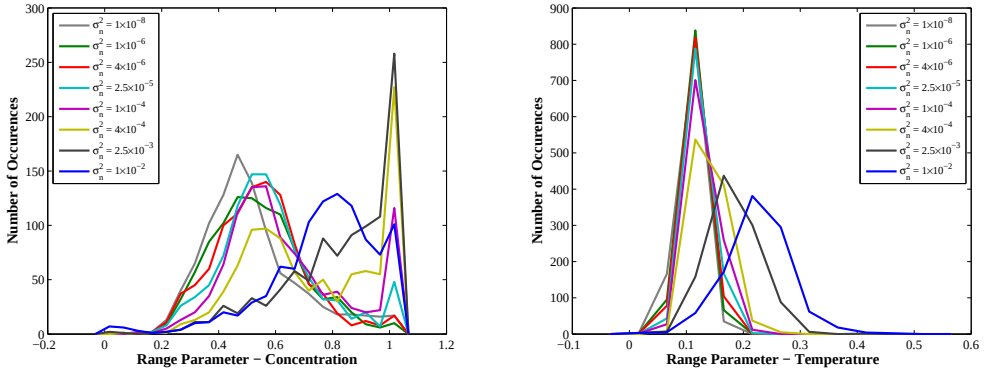
5.4.1.1 Identification of GPM parameters in dynamic systems modeling

Chapter 3 described the difficulties that the maximum likelihood estimator has in the estimation of GPM parameters. The identification of local correlations in a dataset is associated with n, the number of sample points used in the model. More sample points spread out in the sampling space makes the identification of local correlation easier. In addition to the total number of sample points, Chapter 3 shows how repetitions are helpful to characterize the additive noise in the simulations using the estimated σ_u^2. Because of these aspects, the decision of using a constant Δg in the sparsification has significant consequences in the iGPM parameter estimation and dynamic predictions. When a constant Δg is used in the sparsification, the number

of sample points used in the iGPM dynamic framework does not change significantly between different experimental designs of 20 dynamic trajectories from the initial sample region. Additionally, any repetitions that have been captured in the database of dynamic trajectories are neglected, due to the selection of only one data point per square lattice location in the dynamic region. Therefore, it is necessary to analyze how this sparsification alters the MLE parameter estimation and then, how these parameters change when different regression functions are used in GPM.

Figure 32 shows the distribution of estimated GPM parameters using MLE over 1000 different experimental designs of 20 dynamic trajectories from the initial sample region for one of the GPMs in the iGPM dynamic framework. For this case, the GPM uses a constant regression function. Figures 32a and 32b show how the range parameters of each input dimension in the GPM increase as the noise level in the observation increases. Larger range parameters in a GPM mean that the residuals of the underlying deterministic function are similar across the dynamic region. In this particular case study, the presence of noise in the observations makes the residuals of the underlying deterministic function look more similar to each other. Figure 32c demonstrates that the MLE parameter estimation is capable of identifying the local correlation in the iGPM in each of the evaluated noise levels.

The estimation of the uncorrelated variance parameter σ_u^2 deserves a separate discussion from the rest of the GPM parameters in this case study. Figure 33 shows the estimation of σ_u^2 in the GPM that predicts the scaled concentration in the iGPM. Figures 33a is the distribution of estimated σ_u^2 over 1000 different experimental designs, with 20 dynamic trajectories starting in the initial sample region and $\Delta g = 0.05$. This figure shows a clear estimation of the uncorrelated variance parameter, despite the potential problems that the sparsification procedure could carry in the estimation of this parameter. The reason for these estimation results is the particular sampling scheme generated by the sparsification for this non-adiabatic CSTR case study. To



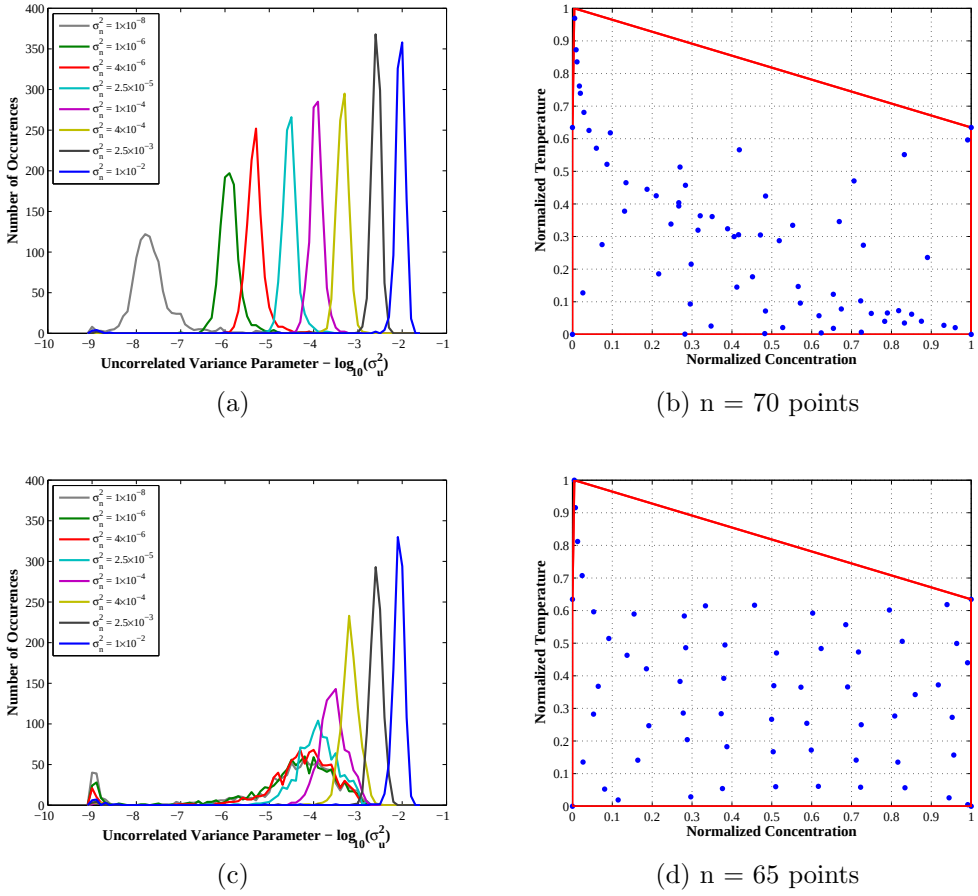
(a) (b)

(c)

Figure 32: Distribution of estimated GPM parameters during the dynamic framework. The estimated parameters corresponds to a GPM with a constant regression function describing the normalized concentration. The distributions are built over 1000 different sets of 20 dynamic trajectories from the initial sample region, using $\Delta t = 20$ s and $\Delta g = 0.05$.

explain this, Figure 33b (copy of Figure 30c) shows one of the experimental designs generated when $n_{dyn} = 20$ and $\Delta g = 0.05$. Notice how there is a higher data density in the neighborhood of the two steady states of the system. Since all the dynamic trajectories in the initial sample region are going to a steady state, it makes sense that most of the square lattices around the steady states are filled with sample points. Also, because of this behavior, the data density decreases as it moves away from any of the steady states, then the regions from where the trajectories started are not sampled as well. This is a trait of the dynamics in this case study, that has been capture

by the sparsification and the appropriate selection of Δg.



(a) (b) n = 70 points

(c) (d) n = 65 points

Figure 33: Estimation of the uncorrelated variance parameter σ_u^2 in iGPM using a constant regression function for the scaled concentration variable. Figures (a) and (b) corresponds to the non-uniform data sampling scheme generated by $n_{dyn} = 20$ and $\Delta g = 0.05$. Figures (c) and (d) corresponds to the uniform data sampling scheme generated by $n_{dyn} = 300$ and $\Delta g = 0.11$. The distributions were obtained by 1000 different experimental designs of the corresponding sampling scheme.

This non-uniform data sampling is going to play a major role during the analysis and the discussion of this chapter, starting from its effects on the parameter estimation. If there were replicates in the dataset $\mathcal{D}$ of the iGPM, those replicates will appear on the regression covariance matrix V as a highly correlated pair of sample points. Because of the sparsification and Δg selection, some sample points are located at shorter distances to each other than the estimated range parameters ℓ_i, resulting

in high local correlation entries in the regression covariance matrix V. Therefore, the iGPM uses these highly correlated values as if they were replicates, leading to an improved estimation of the uncorrelated variance parameter σ_u^2.

To see how significant is the effect of the non-uniform data sampling on the parameter estimation, Figures 33c and 33d show an uniform sampling scheme that can be obtained from the non-adiabatic CSTR case study. The uniform sampling scheme is generated by simulating $n_{dyn} = 300$ dynamic trajectories from the initial sample region, with a sampling rate $\Delta t = 20$ s and a grid spacing of $\Delta g = 0.11$. A example of this uniform sampling scheme for iGPM is shown in Figure 33d. On average the non-uniform and uniform sampling schemes have the same number of sample points in the iGPM —the only difference is the reorganization of the sample points over the dynamic region. Using the configuration for the uniform sampling scheme, 1000 different experimental designs were generated to be used in a iGPM with constant regression functions. Figure 33c shows the distribution of the estimated σ_u^2 for the uniform sampling scheme from this repeated exercise. The uniform sampling scheme is not able to properly estimate the low noise level in the observations. The noise level in the observations is so small that the only way to perceive them is by having repetitions, or "pseudo-replicates" of the sample information, a trait that this uniform sampling scheme does not have. As a result of that, the estimated σ_u^2 in the uniform sampling scheme for low noise cases is much higher than the true noise level. In contrast, the high noise levels are easily estimated because their presence in the stochastic observations is evident, even for this sparser sampling scheme.

After the discussion between the non-uniform and uniform data sampling schemes, the non-uniform data sampling scheme is used to analyze the estimated GPM parameters, when different regression functions are implemented in the iGPM. Similar to the presentation in Figure 32, a distribution of estimated parameters can be created when 1000 different experimental designs use either the linear or quadratic regression

function in the iGPM. Table 6 is a summary of the results of these distributions of estimated parameters. The table collects the mean values of the estimated σ_c^2 and σ_u^2 distributions at different noise levels and regression functions when $n_{dyn} = 20$ and $\Delta g = 0.05$. While the correlation functions explain the local correlation of the data in the dataset $\mathcal{D}$, the regression functions explain the global trends that can be found in the dataset $\mathcal{D}$. The prediction made by a GPM is a balance between these two types of data correlation.

Table 6: Mean values of σ_c^2 and σ_u^2 distribution of GPM parameters during the dynamic framework for different regression functions. The estimated parameters corresponds to the GPM for the scaled concentration. The distributions are build over 1000 different sets of 20 dynamic trajectories from the initial sample region, using $\Delta t = 20$ s and $\Delta g = 0.05$.

Noise	**Constant**		**Linear**		**Quadratic**	
	$\log_{10}\sigma_c^2$	$\log_{10}\sigma_u^2$	$\log_{10}\sigma_c^2$	$\log_{10}\sigma_u^2$	$\log_{10}\sigma_c^2$	$\log_{10}\sigma_u^2$
1.0×10^{-8}	-1.0025	-7.6475	-1.3674	-7.7475	-1.8198	-7.8189
1.0×10^{-6}	-1.0426	-5.8898	-1.3963	-5.9312	-1.8798	-5.9726
4.0×10^{-6}	-1.0485	-5.3311	-1.4013	-5.3557	-1.8974	-5.3996
2.5×10^{-5}	-1.0503	-4.5330	-1.4081	-4.5833	-1.9197	-4.6261
1.0×10^{-4}	-1.0375	-3.9444	-1.3978	-3.9818	-1.9321	-4.0307
4.0×10^{-4}	-1.0192	-3.3482	-1.3853	-3.3992	-1.9549	-3.5068
2.5×10^{-3}	-1.0004	-2.6106	-1.3966	-2.7398	-2.0530	-3.2068
1.0×10^{-2}	-1.0094	-2.1155	-1.4595	-2.5028	-2.2937	-3.8393

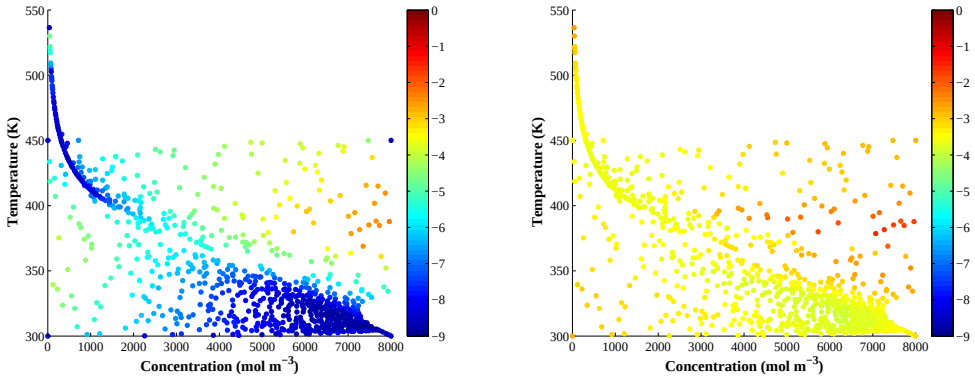
In Table 6, the two columns on the left represent the behavior of the constant regression function, where the MLE offers a good estimation of the noise level in the stochastic observations in σ_u^2, as well as a good identification of the local correlation in σ_c^2. When the linear and quadratic regression functions are used in the iGPM, the estimated value of σ_c^2 decreases at all noise levels in the stochastic observations. This means that the regression functions explains part of the data variability that previously was associated to the local correlation. Table 6 also shows that the presence of the linear and quadratic regression functions do not affect the identification of the low noise levels with the uncorrelated variance parameter σ_u^2. But, the correct estimation

of σ_u^2 at high noise levels is more difficult with the linear and quadratic regression functions, than with the constant regression functions. A possible explanation is that the regression functions are also explaining part of the data variability that should be associated to the noise level in the observations. This situation may be solved by increasing the number of sample points used in the iGPM, but here the analysis focuses on accurate estimations with small sample sets.

5.4.1.2 Local and dynamic predictions of iGPM using different regression functions

The comparison between the usage of the regression functions in iGPM will be evaluated using the local error measurement LEM and the dynamic error measurement DEM. Before reaching a conclusion about this comparison, it is necessary to describe how the non-uniform data sampling scheme, generated by using $n_{dyn} = 20$ dynamic trajectories and $\Delta g = 0.05$, affects the LEM and DEM prediction errors. Figure 34 shows the average LEM values at different locations in the dynamic region over 1000 different experimental designs of the non-uniform data sampling scheme. All LEM prediction errors were obtained with a iGPM using the constant regression function. The figure shows the LEM values at two different noise levels in the stochastic observations. As a consequence of the non-uniform data sampling, the LEM are not evenly distributed across the dynamic regions. The regions where the two steady states are located are the regions with better prediction errors, due to the higher data density. Areas with lower data density make larger LEM prediction errors. The non-uniform LEM prediction errors becomes less significant as the noise level in the observations increases, this is a reasonable consequence of using stochastic observations in the predictions.

The non-uniform LEM prediction error across the dynamic region is associated with the differences in the data density in the region. When the data density decreases, the LEM prediction error increases. Due to the dynamics of this case study,



(a) $\sigma_n^2 = 1 \times 10^{-8}$ (b) $\sigma_n^2 = 1 \times 10^{-2}$

Figure 34: Average LEM for different test sample points in the dynamic region. The figures show the location of the test points in the dynamic region, and the color scale represents the average value of $\log_{10}(LEM)$ created by iGPM over 1000 different sets of 20 dynamic trajectories from the initial sample region. The LEM values were calculated for the GPM of the scaled concentration using a constant regression function.

the regions with the lower data density correspond to regions visited by the dynamic trajectories at earlier discrete time steps, and also to regions with larger LEM prediction errors. This non-uniform sampling scheme has the potential of create large propagation of errors due to the large errors at the beginning of the dynamic trajectories. Similarly, because the data density increases in the direction of the steady states, the iGPM dynamic predictions decrease the amount of propagated error over the predicted dynamic trajectory. This behavior of the non-uniform data sampling makes the prediction error of the first discrete time step to be exceptionally important for the appropriate prediction of the system dynamics.

The local prediction errors shown in Figure 34 describe the performance of a iGPM at specific locations in the dynamic region, but a user might be interested in a single number that summarizes these pointwise values of LEM. An alternative to do this is to average all the local average LEM values over the dynamic region. This average number can be computed for each of the evaluated noise levels in the iGPM. Moreover, it can be calculated for different iGPM implementations with different

regression functions. This analysis is made to compare the performance of the three regression functions and is summarized in Table 7. Since the iGPM uses two different GPMs in its implementation, the results in Table 7 are presented separately for both the scaled concentration and the scaled temperature. There are not many surprises in the LEM values as a function of the noise level in the observations. It is interesting to notice that the prediction of the two GPMs used in the iGPM is quite similar, despite the fact that they were build independently.

Table 7: Mean values of $\log_{10}(LEM)$ over the test samples in the dynamic region for each of the scaled variables predicted by the iGPM. The mean values have been computed at different noise levels and for the three different types of regression functions. The iGPM uses the non-uniform data sampling scheme with $n_{dyn} = 20$ and $\Delta g = 0.05$.

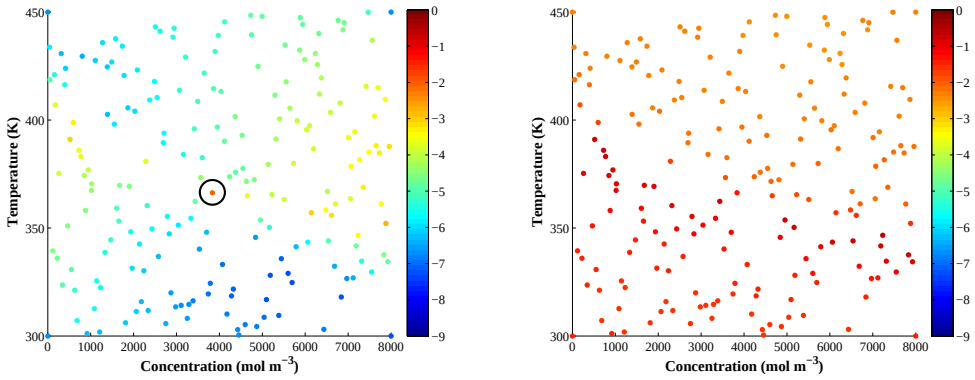
Noise	**Concentration**			**Temperature**		
	Constant	Linear	Quadratic	Constant	Linear	Quadratic
1.0×10^{-8}	-6.6077	-6.6482	-6.6747	-6.9417	-6.9850	-7.0163
1.0×10^{-6}	-5.7405	-5.7596	-5.7993	-5.9614	-5.9854	-6.0230
4.0×10^{-6}	-5.4039	-5.4223	-5.4549	-5.6159	-5.6441	-5.6761
2.5×10^{-5}	-4.9509	-4.9684	-4.9970	-5.1382	-5.1669	-5.1882
1.0×10^{-4}	-4.5904	-4.6121	-4.6298	-4.7714	-4.7783	-4.7967
4.0×10^{-4}	-4.2073	-4.2227	-4.2336	-4.3644	-4.3751	-4.3435
2.5×10^{-3}	-3.6781	-3.6764	-3.5790	-3.8071	-3.7700	-3.5703
1.0×10^{-2}	-3.2908	-3.2111	-2.9734	-3.4334	-3.1372	-3.1131

Table 7 shows that a more descriptive regression function does not necessarily guarantee an improvement in the LEM prediction errors. In general, the prediction of a GPM is a balance between the localized correlation of the data, and the global trends captured in the regression functions. In this case study, using a more descriptive regression function, like the linear or quadratic function, does not significantly decrease the iGPM predictions. At low noise levels, the quadratic regression function improves the LEM prediction error in 1% relative to the results obtained with the constant regression function. It is interesting to contrast the behavior of the LEM at low noise levels with the observed behavior at high noise levels. At high noise levels,

a iGPM with a constant regression function has lower LEM prediction errors than a iGPM with a linear or quadratic regression functions. As an attempt to explain this result, the results on Table 7 were repeated with an iGPM using more sample points, generating the non-uniform data sampling with $n_{dyn} = 80$ dynamic trajectories and $\Delta g = 0.05$, but the LEM prediction errors exhibit the same behavior at high noise. When the noise level in the observations increases, the identification of the local correlation from the stochastic observations becomes more difficult, and the GPM prediction will depend heavily on the selected regression function. Thus it is possible that a high-noise levels, the linear and quadratic regression functions are enforcing a global trend on the stochastic observations when in reality there might not be such global behavior. The consequences of this situation are larger LEM prediction errors over the dynamic region than the constant regression function.

The comparison between the three different regression functions concludes with the performance of the iGPM dynamic predictions. Before making some final conclusions about the usage of regression functions, this section will describe the effects of the non-uniform data sampling in the dynamic predictions made by the iGPM. Figure 35 shows the average DEM prediction error of 200 dynamic trajectories starting from the initial sample region. The average DEM are calculated over $n_s = 25$ discrete time steps and 1000 different experimental designs using the non-uniform data sampling $n_{dyn} = 20$ and $\Delta g = 0.05$. In Figure 35b, it is difficult to distinguish any differences between the average DEM values in the initial sample region. Using stochastic observations with a high noise level hinders the effects of the non-uniform data sampling in the iGPM. In contrast, the effects of the non-uniform data sampling are clearly observed in Figure 35a, where the noise level in the observations is lower. Figure 35a shows two areas in the initial sample region where the DEM prediction errors are significantly lower (upper left and lower right corners in Figure 35a). These two regions of lower DEM prediction error are related with initial sample points that

are already close to the steady states in the system dynamics. But it is more important to recognized that these two regions have different DEM prediction errors. The upper left corner has DEM prediction errors around 1×10^{-6} and the lower right corner has DEM prediction errors of 1×10^{-7}. These differences in the DEM prediction values around the steady states means that there are two data densities in the non-uniform data sampling, represented by the dynamic trajectories going towards each of the steady states.



(a) $\sigma_n^2 = 1 \times 10^{-8}$ (b) $\sigma_n^2 = 1 \times 10^{-2}$

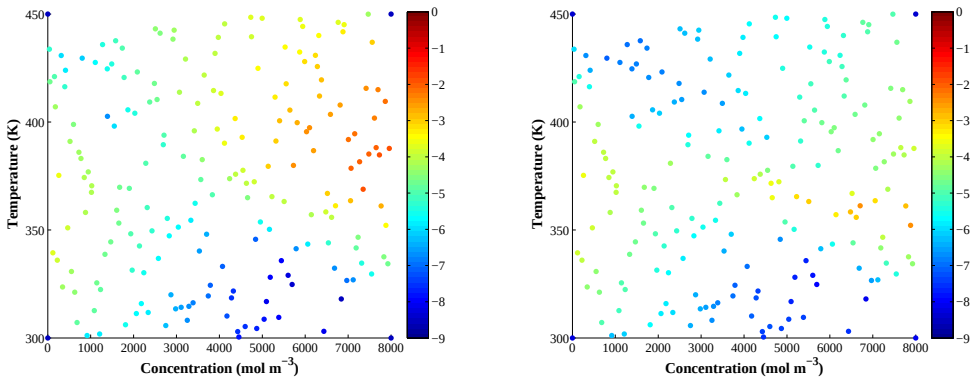
Figure 35: Average DEM values for different dynamic trajectories in the initial sample region. The figures show the location of the initial values of the trajectories in the initial sample region, and the color scale represents the average value of $\log_{10}(DEM)$ created by several iGPM over 1000 different sets of $n_{dyn} = 20$ dynamic trajectories and $\Delta g = 0.05$. The DEM values were calculated for the GPM of the scaled concentration using a constant regression function.

The presence of the two data densities in the non-uniform data sampling generates an additional challenge to the iGPM dynamic prediction: generating a dynamic prediction to the wrong steady state. There is a region in the initial sample region where the dynamic trajectories are more susceptible to be falsely predicted. As was mentioned previously, the most important discrete time step to be predicted in this case study is the first time step. If the dynamic prediction of the first time step is in the wrong direction, the next iterations of the mapping process will lead the dynamic trajectory in the wrong direction. There is no opportunity for the iGPM to recover

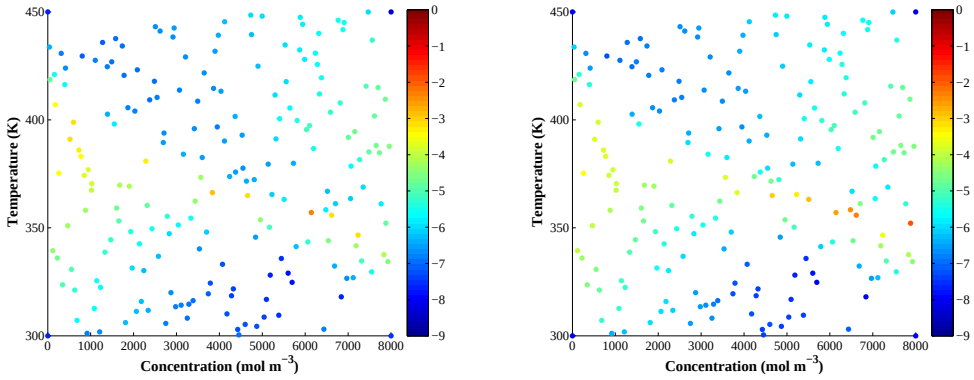
from the prediction error at the first time step, because the data density in the next discrete time steps increases, making more accurate predictions towards the wrong steady state. In Figure 35a, a circle is drawn around a particular dynamic trajectory in the middle of the initial sample region. This dynamic trajectory has the highest DEM prediction error in the initial sample region, ironically in the middle of the region, where the iGPM predictions are supposed to be more accurately. This dynamic trajectory has a unique situation, not only because it has been falsely predicted to go towards the wrong steady state, but also because it is close to an area that lacks sample points due to the repulsion characteristic of the unstable steady state.

The average DEM prediction errors in Figure 35 do not offer a view of how the iGPM dynamic predictions is made at each discrete time step. Figure 36 shows six different snapshots of the iGPM dynamic predictions at different discrete time steps. Using the non-uniform data sampling $n_{dyn} = 20$ and $\Delta g = 0.05$, this figure calculates the average over 1000 different experimental designs of the DEM prediction errors at different discrete time steps. This sequence of DEM prediction error figures shows the evolution of the iGPM dynamic prediction. Figure 36a shows the DEM prediction error at the first time step, which is the same as the LEM prediction error of the corresponding initial value points $\mathbf{x}_0$ in the dynamic trajectories. Figures 36b – 36e show how the DEM prediction errors decrease for all dynamic trajectories as time goes on. This is the consequence of the increase in the data density as the dynamic prediction progresses in time, making more accurate predictions.

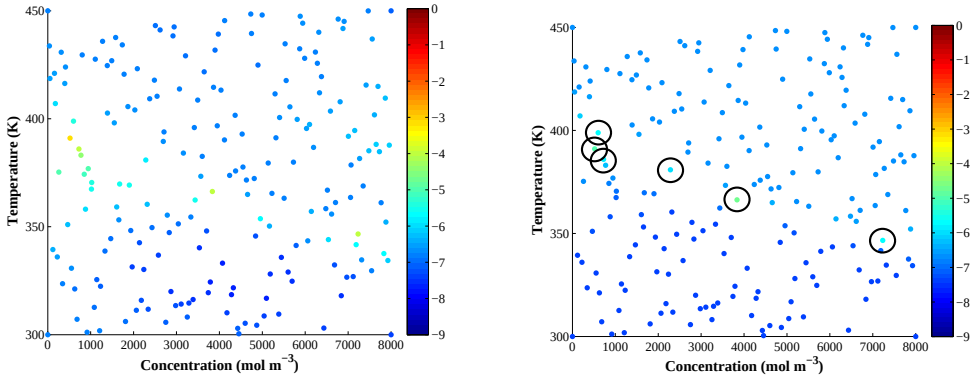
Finally, Figure 36f is a representation of the different data densities generated by the non-uniform sampling scheme and its consequences for the dynamic predictions. It is easy to see which dynamic trajectories are going towards each of the stable steady states. Also in Figure 36f, some dynamic trajectories have been circled in the initial sample region, to describe the trajectories with the largest DEM prediction errors. These dynamic trajectories can be used as a boundary between the two regions that



(a) $t = 20$ s (b) $t = 40$ s



(c) $t = 60$ s (d) $t = 100$ s



(e) $t = 300$ s (f) $t = 500$ s

Figure 36: Average DEM values for dynamic trajectories at different discrete times. The figures show the location of the initial values of the trajectories in the initial sample region, and the color scale represents the average value of $\log_{10}(DEM)$ created by iGPM over 1000 different sets of $n_{dyn} = 20$ dynamic trajectories and $\Delta g = 0.05$. The DEM values were calculated for the GPM of the scaled concentration using a constant regression function and a noise level of $\sigma_n^2 = 1 \times 10^{-8}$.

are present in the initial sample region. These trajectories are located in an area which it is more susceptible to generating dynamic predictions to the wrong steady state. This erroneous dynamic prediction does not occur the 100 % of the cases, since it depends of the selected sample points in the $\mathcal{D}$ dataset of the iGPM. After the figures and discussions about LEM and DEM prediction errors in this multivariate dynamic system, it is possible to conclude that Gaussian process models are a good choice to model system dynamics with one stable steady state, or models that are required to be around a single point in the state space. The reason for this is because in these two cases the higher data density around the region of interest will favor the performance of the dynamic implementation of Gaussian process model, in trying to obtain the best prediction possible.

The comparison between the three regression functions in iGPM for dynamic predictions in the non-adiabatic CSTR case study is shown in Table 8. The plots made in Figure 35, can be made also at the different noise levels, and iGPM using the three regression functions of this study. The reported values in Table 8 are the average DEM prediction error for the two independent GPM, over 1000 different experimental designs, 25 discrete time steps necessary to describe each trajectory, and 200 different dynamic trajectories over the initial sample region. This table shows a similar interpretation as the results for the LEM prediction error in Table 7. At low noise levels, there are not substantial differences between the three regression functions and its prediction qualities in iGPM. At high noise levels of the observations, the constant regression function has lower DEM prediction errors than a iGPM with either the linear or quadratic regression function.

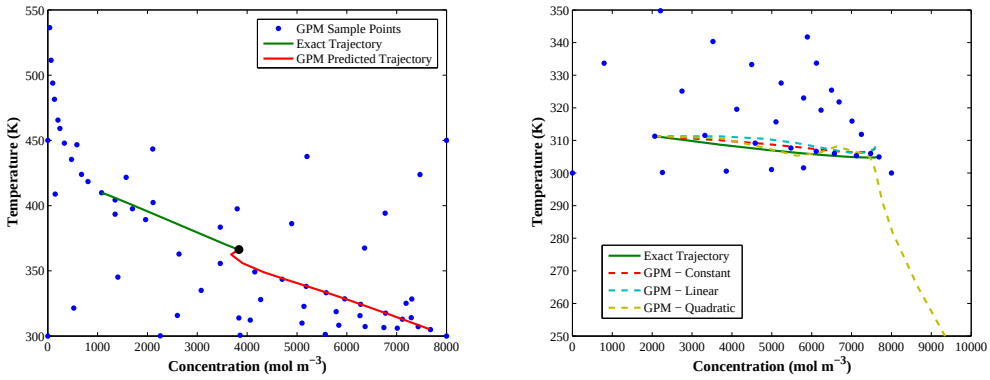
The implementation of iGPM as a recursive mapping function imposes a strong dependence on the sampled data in the dynamic region, in order to generate accurate dynamic predictions. When this dependence is weakened, for example by using stochastic observations with high noise level, the iGPM dynamic prediction

Table 8: Mean values of $\log_{10}(DEM)$ over the test samples in the state space. The mean values have been computed at different noise levels and for the three different types of regression function used in the iGPM. The values with an asterisk represent cases where some dynamic trajectories exhibit extrapolation problems during its iGPM prediction (see Figure 37b).

Noise	**Concentration**			**Temperature**		
	Constant	Linear	Quadratic	Constant	Linear	Quadratic
1.0×10^{-8}	-5.1719	-5.2397	-5.3139	-5.4440	-5.5614	-5.6233
1.0×10^{-6}	-4.4969	-4.5405	-4.5965*	-4.6456	-4.7074	-4.7497*
4.0×10^{-6}	-4.2006	-4.2374	-4.2679*	-4.3122	-4.3492	-4.3673*
2.5×10^{-5}	-3.7690	-3.7740	-3.5713*	-3.8281	-3.8182	-3.6109*
1.0×10^{-4}	-3.3962	-3.3509	-2.0797*	-3.4416	-3.3579	-2.1060*
4.0×10^{-4}	-2.9597	-2.9014	0.5355*	-3.0243	-2.9181	0.4818*
2.5×10^{-3}	-2.2319	-2.1599	0.5779*	-2.3393	-2.2041	0.4850*
1.0×10^{-2}	-1.8120	-1.4788	0.3552*	-1.8598	-1.5284	0.2387*

will depend more on the regression function. Therefore, the iterative usage of a set of regression functions that do not capture the global trend in the residuals of the function to be approximated could lead to an uncontrolled dynamic prediction by the iGPM. This problem can be explained as an extrapolation from the dataset $\mathcal{D}$ in the iGPM dynamic framework, shown in more detail on Figure 37. This situation describes why some of the DEM prediction errors for the quadratic regression function go to infinity in Table 8. The extrapolation problem in this case study does not appear in all the tested dynamic trajectories, and it does not happen in the 100% of the iGPM dynamic frameworks that were tested.

With the results from the DEM prediction errors, it is then possible to reach a conclusion regarding the usage of regression function in Gaussian process models for system dynamics. For the purposes of using Gaussian process modeling as a recursive mapping function from stochastic observations, the constant regression function offers the best overall performance in this task. This case study cannot disregard the usage of more descriptive regression functions, because this study shows that they improve the iGPM dynamic predictions at very low noise levels. However, the better



(a) (b)

Figure 37: Potential problems during the prediction of dynamic trajectories using iGPM for the non-adiabatic CSTR case study. (a). False prediction of a final steady state for an initial value near to the unstable steady state. (b) Extrapolation of GPM dynamic predictions from the training dataset for a GPM with a quadratic regression function. Both of these figures were made using a dataset of $n_{dyn} = 20$ dynamic trajectories and $\Delta g = 0.05$ from the initial sample region. Figure (a) uses observations with a noise level of $\sigma_n^2 = 1 \times 10^{-8}$, while Figure (b) uses observations with a noise level of $\sigma_n^2 = 1 \times 10^{-4}$.

prediction of the constant regression function at moderate to high noise levels, and the potential risks with the extrapolation problem, makes the constant regression function a more robust choice for Gaussian process models. There is one additional scenario to consider, in which a user has the knowledge to use a suitable set of regression functions to represent the dynamics of a system. While it seems that it will prevent the extrapolation problem from happening, the prediction of such iGPM will not depend on the local correlation of residuals. Remember that a more descriptive regression function limits the identification of a local correlation between the residuals in the GPM. Therefore, all the error estimation properties of the iGPM will be lost.

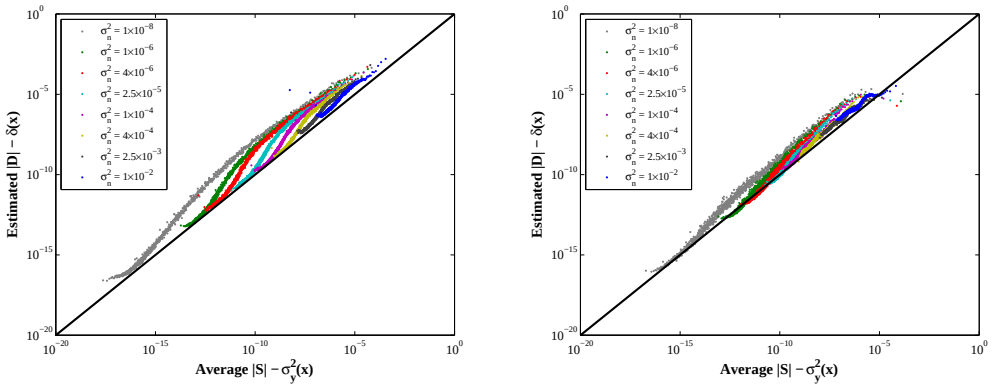
5.4.2 Error estimation of dynamic GPM predictions for multiple variables

For the remainder of this analysis, and based on the conclusion shown in the previous paragraph, all iGPM and mGPM will use a constant regression function as part of

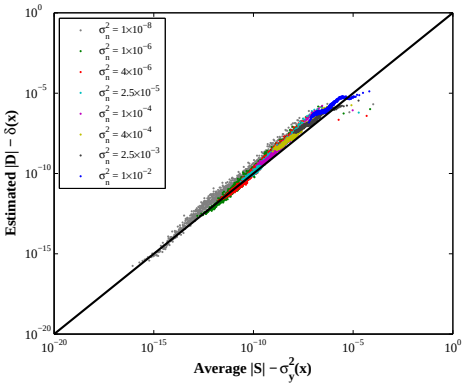
their implementation. With a more concrete selection of regression functions, the next step is show how the error estimation analysis described in Section 5.1.2 is implemented for the non-adiabatic CSTR case study. First, this section evaluates the error estimation analysis for the one-step-ahead prediction error. Figure 38 shows the implementation of the error estimation analysis for a iGPM using a constant regression function. This error estimation analysis uses the unbalaced data sampling scheme with $\Delta g = 0.05$, but changing the number of dynamic trajectories in the analysis. These figures shows the correlation between the estimated $|D|$ and the average $|S|$ in each of the prediction vectors of the model. Also, this figure shows the improvement in the error estimation properties of the iGPM by increasing the number of dynamic trajectories used for the database $\mathcal{D}$.

In previous chapters, the evaluation of the error estimation properties of a GPM has been made using some sort of uniform sampling across the sampling space. Remember that in Chapter 3, a Latin hypercube design was used to describe the error estimation analysis across the design space, and in Chapter 4 the equally-spaced design of sample points was used to evaluate the propagation of error in a GPM. Figure 38 shows the results of the error estimation analysis for the non-uniform data sampling, but the question here is if a more uniform data sampling scheme has better error estimation properties. Figure 39 shows the results of the error estimation analysis when a uniform data sampling scheme is used in the non-adiabatic CSTR case study. The uniform data sampling scheme was previously describe in Section 5.4.1.1 and Figure 33d as an example to explain the accurate estimation of the noise level in the non-uniform data sampling scheme, with only one repetition.

The results shown in Figure 39 are compared with the results in Figure 38a, since both of those sampling schemes have on average the same number of sample points, $n = 65$. The scatter plots for the uniform data sampling are much closer to the 1:1 line than the scatter plots for the non-uniform data sampling, indicating a better



(a) $n_{dyn} = 20$ (b) $n_{dyn} = 80$



(c) $n_{dyn} = 140$

Figure 38: Error estimation analysis of iGPM for a one-step-ahead prediction error. The figures shows the number of dynamic trajectories n_{dyn} used in each of the iGPM. The figures were computed using 1000 different experimental designs of the various dynamic trajectories with $\Delta g = 0.05$, at different noise levels. All iGPM used a constant regression function.

error estimation properties of its iGPM prediction. In contrast, the uniform data sampling does not have a good iGPM mean prediction, like the non-uniform data sampling. By using larger grid spacing $\Delta g = 0.11$, the uniform data sampling scheme does not reflect in its dataset the steady states that are present in the dynamics of the system. As a result, the scatter plots for the error estimation in Figure 39 are a higher range $[1 \times 10^{-11} - \ 1 \times 10^{-4}]$, compared to the scatter plots for the error estimation in the non-uniform data sampling $[1 \times 10^{-17} - \ 1 \times 10^{-4}]$. The idea with these error estimation studies is not to obtain the most accurate iGPM possible, but

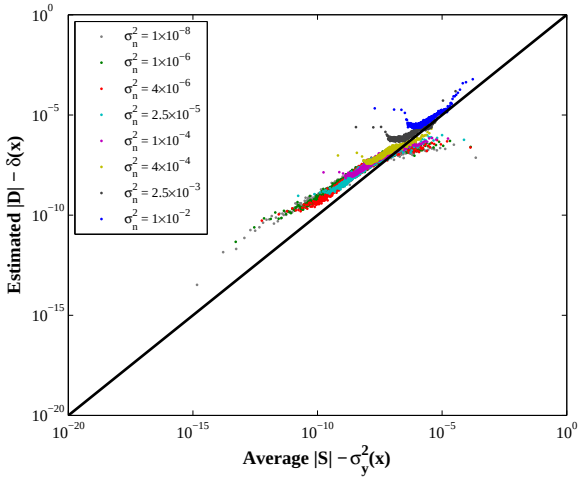
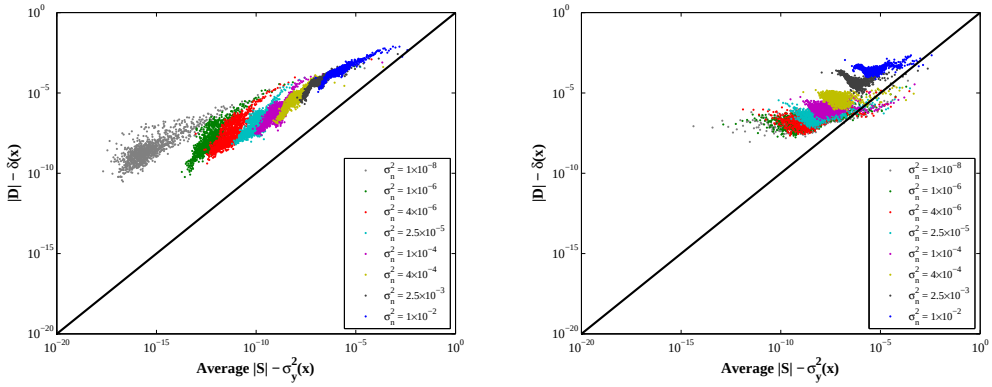


Figure 39: Error estimation analysis of multivariate Gaussian process models for a one-step-ahead prediction error using a uniform sampling scheme. Figure 39 was computed using 1000 different experimental designs with $n_{dyn} = 300$ dynamic trajectories and $\Delta g = 0.11$, at different noise levels. All iGPM were using a constant regression function.

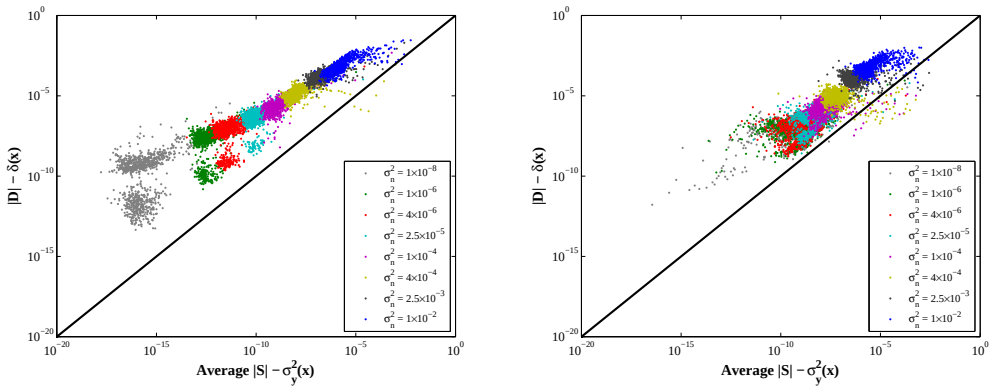
to describe which model is more capable to estimate its own error, no matter if it is a good or a bad prediction.

Chapter 4 shows a first glance of using the error estimation analysis to evaluate the properties of GPM in dynamics. Using a similar procedure, the error estimation analysis for multivariate dynamic systems can be implemented to characterize how the prediction error of the iGPM changes over the course of a dynamic prediction. Figure 40 shows the implementation of the error estimation analysis to the prediction error vectors of 200 dynamic trajectories at different discrete time steps. The figures on the left (Figures 40a, 40c and 40e) are generated by the non-uniform data sampling, while the figures on the right Figures 40b, 40d and 40f) correspond to the uniform data sampling scheme.

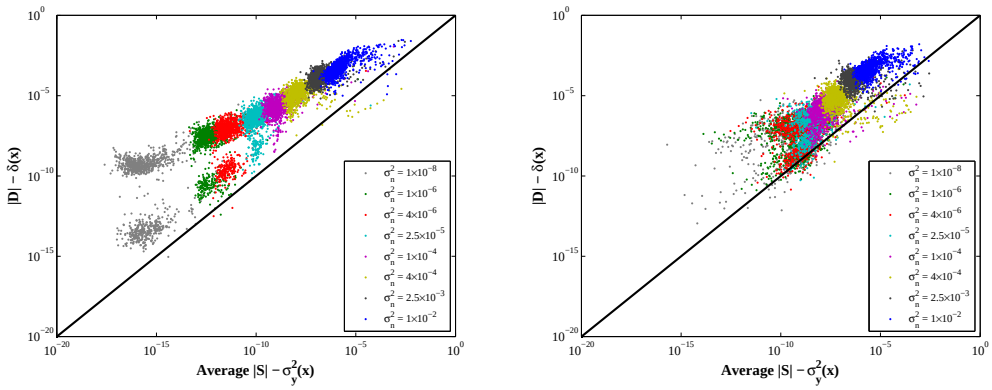
The effects of the non-uniform data sampling on the error estimation analysis are clear. Since the data sampling has two different data densities, with two different prediction errors, it was expected that the error estimation will also be divided between the two steady states. There is a clear division in the error estimation of the



(a) $t = 100$ s (b) $t = 100$ s



(c) $t = 300$ s (d) $t = 300$ s



(e) $t = 500$ s (f) $t = 500$ s

Figure 40: Error estimation analysis of iGPM at different discrete time steps. The figure describes the propagation of error during the dynamic prediction of the non-adiabatic CSTR case study, at three different discrete time steps. Figures (a), (c) and (e) were made using $n_{dyn} = 20$ dynamic trajectories and $\Delta g = 0.05$, while Figures (b), (d) and (f) were made using $n_{dyn} = 300$ dynamic trajectories and $\Delta g = 0.11$. All figures used 1000 different experimental design of their corresponding sampling scheme, at several noise levels and using a constant regression function.

iGPM as the time progresses in the dynamic prediction. The division between the error estimation properties between the two steady states is clearer at the low noise levels, where it is easier to identify both of them. As the noise level in the observation increases, the estimation of the steady states becomes harder, making the error estimation properties of the iGPM more similar across the dynamic region. To contrast the behavior of this divided error estimation analysis, a similar study was made with the uniform data sampling scheme, to evaluate its properties in a dynamic context. By increasing Δg to create the uniform data sampling scheme, the dataset $\mathcal{D}$ does not contain enough spatial resolution to capture the presence of the steady states in the input space. By removing the description of the steady states in the dataset $\mathcal{D}$, the error estimation is more uniform during the dynamic predictions made by the iGPM. Figures 39 and 40 support the conclusion that a uniform experimental design is a desired feature in a Gaussian process model to improve the error estimation properties of the model.

An additional comment in the error estimation analysis presented in Figure 40, is to notice how the scatter plots get closer to the 1:1 line, despite that many discrete time steps have passed. This situation occurs in both the non-uniform and the uniform data sampling schemes. At $t = 100$ s, both scatter plots are far away from the 1:1 line, compared to the results at $t = 500$ s. This change is more evident in the non-uniform data sampling scheme. This sudden improvement of the error estimation is based on the behavior of the iGPM when the dynamic prediction is getting closer to a high data density area. As the iGPM dynamic prediction progresses in time, the dynamic trajectory visits areas with higher data density than the previous time steps. A natural consequence in GPM of doing predictions in regions with high data density is that these predictions are more accurate. In the context of a recursive mapping function, more accurate predictions along the dynamic trajectory means less propagation of error, leading to better error estimation properties of the iGPM.

5.4.3 Predictions of a multivariate Gaussian process model for system dynamics

This results section concludes with the implementation of the multivariate Gaussian process model (mGPM) as a recursive mapping function for the non-adiabatic CSTR case study. This chapter has described key aspects to consider in the implementation of iGPM to predict multivariate dynamic systems. However, mGPM offers an alternative mathematical approach and implementation as a recursive mapping function for system dynamics. Based on the mathematical description in Section 5.1.1, mGPM is used to describe the dynamics of the non-adiabatic CSTR case study, using the non-uniform data sampling scheme with $n_{dyn} = 20$ dynamic trajectories and $\Delta g = 0.05$. This last section presents a comparison between mGPM and iGPM, where both of the dynamic implementations use a constant regression function. The first comparison between these models is the results of the parameter estimation for both of these models. Table 9 summarizes the estimation of σ_c^2 and σ_u^2 in both iGPM and mGPM models, for the prediction of the scaled normalization variable in the case study. This comparison between the estimated parameters of iGPM and mGPM is possible because both GPM dynamic implementations use the same correlation function. The results in Table 9 shows that the LCM parameterization in the mGPM is capable of estimating the noise level in the observations with the uncorrelated variance parameter σ_u^2 in the scaled normalization variable. The accurate identification of the noise level also occurs for the scaled temperature variable.

Table 9 shows that the estimated value of the correlated variance parameter σ_c^2 in a mGPM is greater than in the iGPM. This result in the parameter estimation appears in both the scaled concentration and scaled temperature variables. An increase in the nominal value of σ_c^2 represents that the mGPM is strengthening the local correlation between the residuals of the variables. The presence of additional variables is a synergistic effect in the identification of local correlations in the multivariate data,

Table 9: Comparison of the estimated σ_c^2 and σ_u^2 distributions in iGPM and mGPM. This table summarizes the mean values of the estimated parameter distributions for the scaled concentration. The distributions are build over 1000 different sets of $n_{dyn} = 20$ dynamic trajectories from the initial sample region, using $\Delta t = 20$ s and $\Delta g = 0.05$. All iGPM and mGPM used a constant regression function.

Noise	**iGPM**		**mGPM**	
	$\log_{10}\sigma_c^2$	$\log_{10}\sigma_u^2$	$\log_{10}\sigma_c^2$	$\log_{10}\sigma_u^2$
1.0×10^{-8}	-1.0025	-7.6475	-0.6094	-7.2803
1.0×10^{-6}	-1.0426	-5.8898	-0.6842	-5.6902
4.0×10^{-6}	-1.0485	-5.3311	-0.7033	-5.1299
2.5×10^{-5}	-1.0503	-4.5330	-0.7163	-4.3851
1.0×10^{-4}	-1.0375	-3.9444	-0.7490	-3.8286
4.0×10^{-4}	-1.0192	-3.3482	-0.8057	-3.2661
2.5×10^{-3}	-1.0004	-2.6106	-0.8553	-2.5447
1.0×10^{-2}	-1.0094	-2.1155	-0.9017	-2.0275

possibly leading to a more robust GPM dynamic implementation.

Implementing a multivariate Gaussian process models means modeling the cross-covariance terms in the regression covariance matrix V. Figure 41 shows the distributions of the estimated σ_c^2 and σ_u^2 for the cross-covariance matrix in the non-adiabatic CSTR case study. This figure is divided between the estimated negative and positive values of the σ_c^2 and σ_u^2 for the cross-covariance matrix. The MLE parameter estimation for mGPM shows a tendency to identify the negative correlation of the prediction errors, since most of the estimated values of σ_c^2 and σ_u^2 are negative. This is in agreement with the results shown in Figure 27a with the prediction error vectors for the error estimation analysis. Figures 41a and 41b shows how $\sigma_{c,12}^2$ identifies the underlaying negative correlation between the residuals of the scaled concentration and temperature. It is also clear from these figures that the identification of the negative correlation is not 100% of the cases accurate. Some of the estimated negative values and the few positive values of $\sigma_{c,12}^2$ are much more smaller ($\approx \pm 1 \times 10^{-8}$), than the correlated variance parameters $\sigma_{c,11}^2, \sigma_{c,22}^2$ $(0.1 - 0.2)$.

The estimated values for $\sigma_{u,12}^2$, shown in Figures 41c and 41d, indicate that part

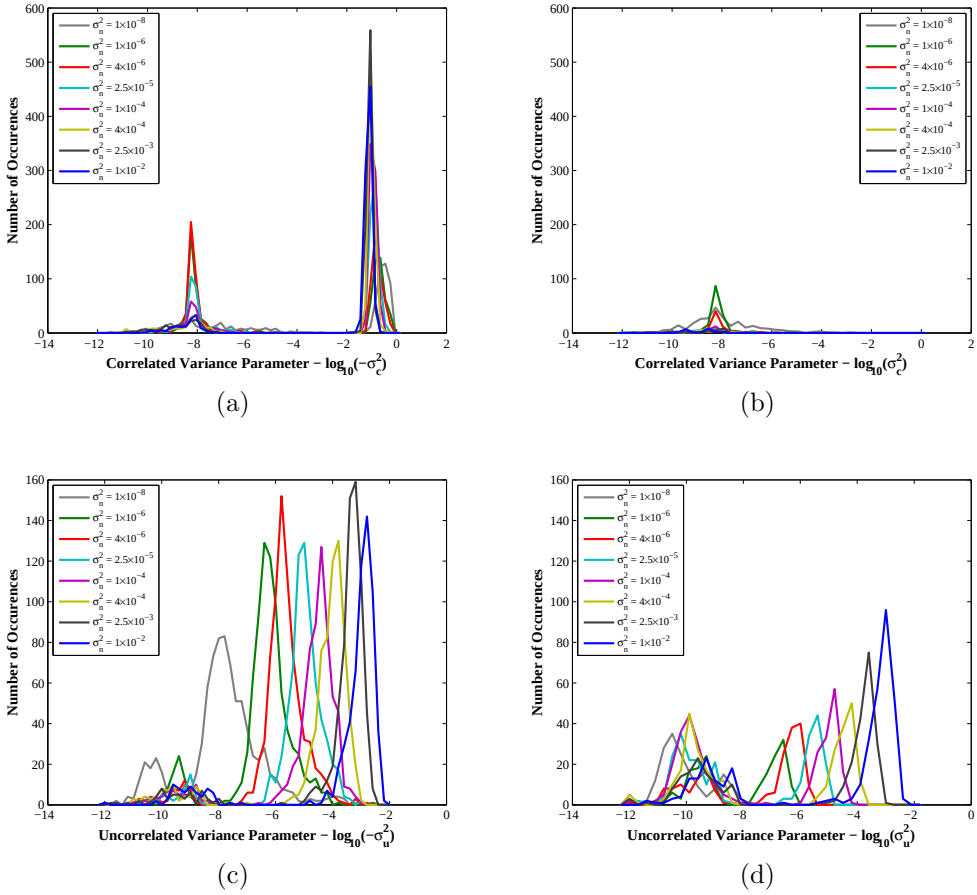


Figure 41: Distribution of estimated GPM parameters for the cross-covariance term in a mGPM using the linear coregionalization model. Figures 41a and 41c show the negative estimated values of the cross-covariance parameters $\sigma^2_{c,12}$ and $\sigma^2_{u,12}$, while Figures 41b and 41d show their positive estimated values. The distributions are build over 1000 different sets of $n_{dyn} = 20$ dynamic trajectories from the initial sample region, using $\Delta t = 20$ s and $\Delta g = 0.05$.

of the local correlation between the predicted errors is explained by a correlation of the measurement noise in the observations. This result is contradictory with the artificial and uncorrelated measurement noise that is added in the observations. Figures 41c and 41d exhibit clear mean distribution values of $\sigma^2_{u,12}$ that are slightly smaller than the estimated noise levels in $\sigma^2_{u,11}$ and $\sigma^2_{u,22}$. This indicates a strong correlation between these mGPM parameters. Perhaps the most important element in Figures 41c and 41d is the large number of positive estimated values, which could suggest the

erroneous correlation between the prediction error of the system.

The next step in the comparison between iGPM and mGPM is the mean prediction of the system dynamics. To do this, LEM and DEM prediction error values for the overall dynamic region are calculated for the mGPM dynamic implementation, as it was done for the iGPM in Tables 7 and 8 respectively. The LEM and DEM values are tabulated in Table 10, for comparison with the iGPM results. The values in this table correspond to the prediction of the scaled normalization. Although, the mGPM dynamic implementation does not improve the one-step-ahead prediction over the iGPM implementation, it does improve the accuracy of the dynamic predictions in the non-adiabatic CSTR case study, specially at the higher noise levels. The biggest improvements in the predictions are made at higher noise levels, due to the improvement in the identification of the local correlation in the estimated σ_c^2. In contrast with these mean prediction results, the mGPM identifies the negative correlation of the prediction errors in the mGPM predictive covariance matrix S. Using the elements in this matrix, it is possible to calculate an estimated Pearson linear correlation $\rho(\mathbf{x})$ at each of the test sample points, since it contains the variances and covariances between the predicted state variables. Using the definition of the state covariance matrix S in Equation (139), the estimated Pearson linear correlation in the mGPM for this case study is:

$$\rho(\mathbf{x}) = \frac{\sigma^2_{y,CT}(\mathbf{x})}{\sigma_{y,CC}(\mathbf{x})\ \ \sigma_{y,TT}(\mathbf{x})} \tag{142}$$

Figure 42 shows the distribution of the estimated Pearson linear correlation at each of the sample points in the computation of LEM, over 1000 different experimental designs. Figure 42 shows the mGPM is capable of predicting the strong negative correlation in the data at lower noise levels. This figure also shows that the mGPM identifies the negative correlation despite the increase in the noise level of the observations, a trait that would be impossible with the iGPM implementation.

The major advantage for the mGPM comes from its mathematical formulation

Table 10: LEM and DEM prediction errors of iGPM and mGPM for the scaled normalization variable. Both GPM dynamic implementations used the non-uniform data sampling scheme with $n_{dyn} = 20$ dynamic trajectories and a grid spacing of $\Delta g = 0.05$. These LEM and DEM prediction errors are averages over 1000 different experimental designs.

Noise	**$\log_{10}$(LEM)**		**$\log_{10}$(DEM)**	
	iGPM	mGPM	iGPM	mGPM
1.0×10^{-8}	-6.6077	-6.4561	-5.1719	-5.0876
1.0×10^{-6}	-5.7405	-5.6490	-4.4969	-4.4828
4.0×10^{-6}	-5.4039	-5.3394	-4.2006	-4.2004
2.5×10^{-5}	-4.9509	-4.9108	-3.7690	-3.7700
1.0×10^{-4}	-4.5904	-4.5509	-3.3962	-3.4132
4.0×10^{-4}	-4.2073	-4.1922	-2.9597	-3.0013
2.5×10^{-3}	-3.6781	-3.6959	-2.2319	-2.3772
1.0×10^{-2}	-3.2908	-3.2907	-1.8120	-2.0273

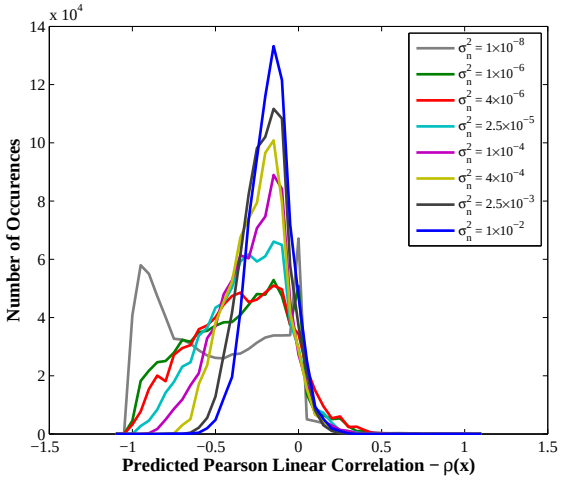
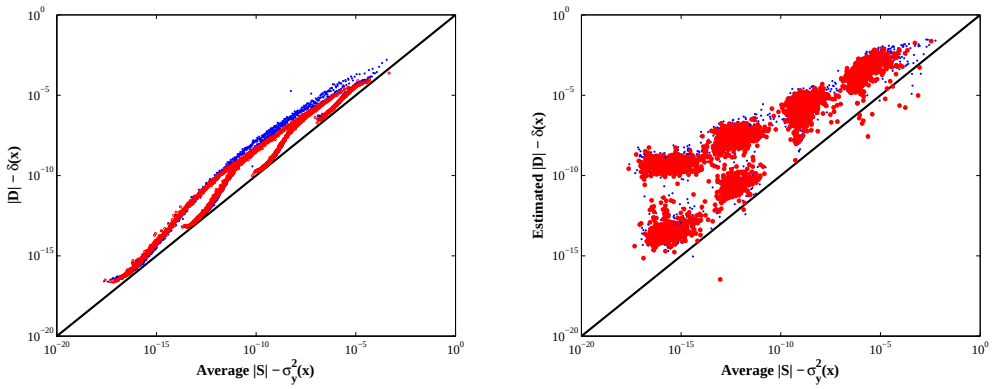


Figure 42: Distribution of estimated Pearson linear correlation $\rho(\mathbf{x})$ by the mGPM implementation for a one-step-ahead prediction at different noise level in the observations. The distributions were made using 1000 different experimental designs, each of them with $n_{dyn} = 20$ dynamic trajectories and a grid spacing of $\Delta g = 0.05$.

to represent correlations between the predicted variables in the mGPM predicted covariance matrix S. The role of these additional correlations could be more relevant at the moment of evaluating the error estimation properties of the mGPM. Figure 43 shows a final comparison of the error estimation properties of the iGPM and the mGPM. The construction of the scatter plots for mGPM is made in a similar fashion

as for the iGPM in Figures 38a and 40f. These scatter plots are made using the 1000 different experimental designs of the non-uniform data sampling scheme with $n_{dyn} = 20$ and $\Delta g = 0.05$. Figure 43a shows the results for the one-step-ahead prediction error, where the mGPM exhibit a slight improvement over the iGPM implementation. Figure 43b shows the error estimation analysis of a dynamic prediction at the end of all dynamic trajectories. This last figure shows that iGPM and mGPM have similar error estimation properties, despite the improve mathematical formulation of the multivariate Gaussian process model. Based on the results shown here demonstrated that mGPM is a more robust GPM dynamic implementation at higher noise levels in the observations, because of its improved identification of the local correlation in the multivariate dynamic data.



(a) (b) t = 500 s

Figure 43: Error estimation analysis of mGPM dynamic implementation and comparison with the iGPM dynamic implementation. The blue dots corresponds to the iGPM scatter plots of the predictions errors, while the red dots corresponds to the mGPM prediction errors. Figures (a) and (b) evaluate the results of iGPM and mGPM under four different noise levels $\sigma_n^2 = [1 \times 10^{-8}, 1 \times 10^{-6}, 1 \times 10^{-4}, 1 \times 10^{-2}]$. (a) Error estimation analysis for the one-step-ahead prediction error. (b) Error estimation analysis for the dynamic predictions at t = 500 s.

5.5 Discussion

This chapter discusses the implementation of Gaussian process models to approximate the dynamics of a multivariate stochastic system. At first, the non-adiabatic continuous stirred tank reactor seems to be a fairly easy mathematical model to describe. However, this simple case study illustrates some of the challenges of treating the system identification problem from the spatial statistics point of view. In most cases when GPM is used in spatial statistics, the user has full control of the experimental design that is going to be used in the model. Many applications use a hypercube input space, allowing space-filling designs to provide a good coverage over the region. In this case study, the relevant sampling region for the model is the dynamic region, which is indirectly constructed from the exploration of the initial sample region. This means that the user does not know *a priori* the input space for the GPM. In addition to this situation, the shape of the dynamic region in this case study is a non-convex area, with a non-uniform distribution of the sample points, thanks to the two stable steady states. These are not characteristics that are regularly faced in spatial statistics and design of computer experiments, which suggest that novel techniques will be required.

The approach used in this thesis to model the system dynamics is the construction of a time-invariant iterative model (in this case the GPM) based on a set of pre-recorded dynamic trajectories. This characteristic of the GPM dynamic implementation means this research is in the area of spatial statistics, and not in the time-series analysis area, where the correlation of sample points in time is explicitly stated. The database $\mathcal{D}$ in the Gaussian process model captures the dynamics of a system through the location of the sample points in the dynamic region, and through the analysis of the output information. The first one because it is necessary a good exploration of the dynamic region, and the second one, because it represents the dynamic evolution of the system. The analysis of the output information, in particular

the gradient and Hessian of the output, points the dynamic prediction in the appropriate direction. If the gradients of the outputs were also considered during the selection of sample points in the sparsification, then the number of falsely predicted steady-states might have be smaller. The selection of sample points for the GPM dynamic implementation must consider the gradient of the approximated function as part of their selection criteria.

While the exploration of the initial sample region is important to build the dynamic region, the sparsification procedure is the step that captures the unique dynamic features of this case study. This chapter shows the effects of using different spatial resolutions (measured by the different values of Δg) in the identification of the GPM parameters and later in its predictions. Although this chapter does not provide indications about the selection of the appropriate grid spacing, the value $\Delta g = 0.05$, on a normalized [0 - 1] space seems to be reasonable for a good spatial resolution. The major concern with this simple, but effective procedure is the quantification of the GPM prediction error due to lack of spatial resolution, and then later how this error propagates into the iterative mapping for dynamics. Researchers in the area of machine learning have proposed several methods to reduce the computations during the inverse of the regression covariance matrix V. A review about this subject can be found in Reference [118]. In many of these approaches, the researchers select a subset of the available sample points as an approximation of the whole dataset. In particular, Shen and cooworkers [128] present an approximation method for GPM based on kd-trees, where an adaptive procedure is implemented to subdivided the input space in hyper-rectangles, and then a weighted sum of the information in each hyper-rectangle is used for GPM prediction. This type of research could be important in the approximation of expensive stochastic dynamic simulations, since all available information will be considered, instead of just a subset of it.

Although the Gaussian process model has the capacity to include functions (either

linear or non-linear basis functions) as global trends in the dynamic information, this is not a recommended practice for iterative mapping functions. The reasons for this recommendation are clear: the extrapolation problem and the effects of those global trends in the identification of local correlations. The extrapolation problem might be caused by the undersampling in certain areas of the dynamic region, or because a problem of model mismatch in the selection of regression functions. Chapter 3 showed that the local correlation in GPM is responsible for all its error estimation properties. Then, by using global trends in the GPM, the user is losing one of the most important traits of the model.

This chapter shows that it is possible to quantify the prediction error of a multivariate GPM using the predicted covariance matrix S. This chapter also shows that using $|S|$ as a metric for error estimation in mGPM is a good idea, and that it correlates appropriately with the uncertainty level in a multi-dimensional prediction. Perhaps a problem with the implemented error estimation analysis for mGPM is that it does not allow one to separate between the different sources of propagation of error, meaning, it is not possible to identify which variables are less accurate than others. Future studies on this covariance matrix could explore the relationships between the off-diagonal terms in the predicted covariance matrix with the individual effects of each input dimension in the mGPM prediction.

Overall, the results of using mGPM for dynamic predictions in multivariate systems were below expectations. Although the LEM and DEM prediction error in mGPM decrease at all noise levels for the scaled concentration and temperature, there is not significant improvement over the iGPM results. It is important to realize that mGPM was capable of identifying the negative correlation in the prediction errors of this case study, but that identification did not improve the error estimation properties of the model. An initial hypothesis about why mGPM did not perform

better during the error estimation analysis might be associated with its local correlation structure. mGPM exhibits a synergistic effect between the variables in their mean prediction, at the cost of a fixed description of the local correlation features of the variables. Compared to the two independent GPM models, where there are two different range parameters in each of them to capture local correlations in each direction, the mGPM does not have enough flexibility to capture local correlations. Further studies must be carry out to explore this hypothesis, but the recommended GPM implementation as an approximation of multivariate, stochastic and dynamic systems is the iGPM framework with a constant regression function.

CHAPTER VI

EVALUATION OF METAMODELING APPROACHES FOR DISCRETE TIME APPROXIMATIONS IN NANOPARTICLE SYNTHESIS

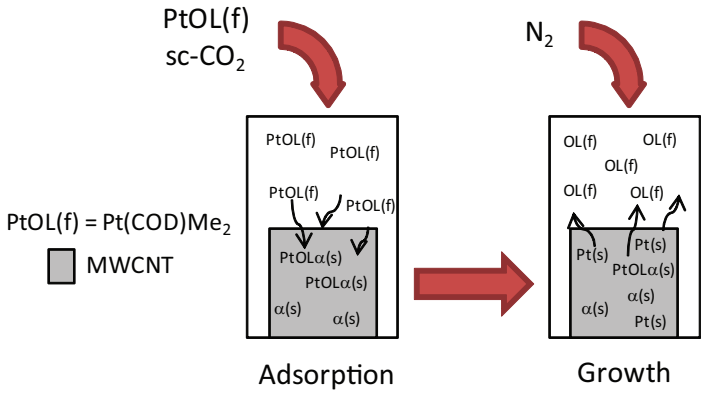
This last chapter closes the discussion regarding Gaussian process models, with a systematic comparison between metamodeling approaches as alternatives to approximate models of expensive dynamic simulations. So far, this thesis has focus on understanding, developing and exploiting a framework where Gaussian process models are used to describe the dynamics of a system. This chapter puts to the test such GPM dynamic implementation by comparing it with other metamodeling approaches. The objective in this chapter is to provide a more general context to the theoretical analysis developed from GPM, and to explore alternative solutions that can be implemented as approximate models. The case study used in this chapter is the nanoparticle dynamic model, which describes the growth of platinum nanoparticles on the surface of a carbon nanotube. This chapter starts with a detailed description of the nanoparticle dynamic model. Then, it continues with a description of how the nanoparticle dynamic model is implemented for metamodeling. Next, a brief mathematical description of different metamodeling approaches is presented to provide a general context to the systematic comparison. Finally, this chapter concludes with the results from the comparison of metamodels.

6.1 Case study: Deposition of platinum nanoparticles on carbon nanotubes under supercritical conditions

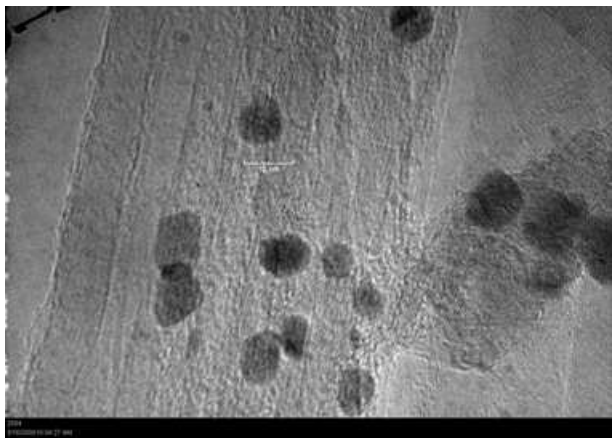
The chemical process used as a case study in this chapter is the deposition of platinum nanoparticles on carbon nanotubes in supercritical carbon dioxide (sc-CO_2) [57]. Large-scale production of nanoparticles of controlled size are needed for a variety of applications, from fuel cells to drug delivery. The sc-CO_2 process has been investigated recently for nanoparticle synthesis, with advantages including the high solubility of precursor in sc-CO_2, the lack of organic solvents which are environmental hazards, the scalability to three-dimensional processing via porous supports or powders, and the ease of separation of the final product from the solvent (by lowering the pressure) [34]. The mathematical model for this chemical process couples a mass action kinetics model of the supercritical fluid phase to a stochastic simulation for the surface kinetics on the carbon nanotubes. While a number of assumptions were made in constructing this model, it highlights the challenges and opportunities to create approximate models of nanoscale phenomena based on the dynamic data collected from a simulation.

A brief description of the process is provided in Figure 44a, motivated by the procedure of Bayrakceken and co-workers [13]. This synthesis involves two stages: the adsorption of the platinum precursor onto the surface, and the subsequent growth of platinum nanoparticles on the surface. Specifically, the organometallic platinum precursor dimethyl(1,5-cyclooctadiene)platinum(II) (PtOL) is first solubilized in a high pressure sc-CO_2 environment, so that it may be adsorbed by the functionalized multi-walled carbon nanotubes (CNT), both present in a chamber. Once the adsorption step is complete, the pressure is released, and N_2 gas flows into the chamber at an elevated temperature. The platinum precursor PtOL undergoes a thermal reduction in an inert N_2 atmosphere, leaving isolated platinum atoms (Pt_1) on the surface of the carbon nanotubes, and releasing the organic ligands (OL) back into the fluid

phase. The platinum atoms then form nuclei on the surface which grow into larger nanoparticles. Many other variations on this process also exist, such as the use of hydrogen in a chemical reduction of the platinum precursor [86].



(a)



(b)

Figure 44: (a) Platinum nanoparticle synthesis under thermal decomposition in an inert atmosphere. (b) Transmission electron microscopy image from a sc-CO_2 process for Pt nanoparticles on carbon nanotubes (Image by Dr. Galit Levitin, Georgia Institute of Technology). The scale bar is 10 nm.

Figure 44b is a transmission electron microscopy image of platinum nanoparticles deposited on multi-walled carbon nanotubes by a sc-CO_2 process. The typical nanoparticle size is between 5–10 nm, with a visible distribution of sizes. From a practical standpoint, it is not desirable to have a wide nanoparticle size distribution, since many applications demand a specific size for optimal performance. However, in practice there will always be some distribution of sizes, and a key manufacturing

question is to determine what distribution will provide adequate performance while minimizing processing cost.

A coupled deterministic-stochastic model is presented here, motivated by the procedure in Reference [13] and previous modeling studies [35, 49, 121]. A system of ordinary differential equations (ODE) is constructed for the adsorption step and a kinetic Monte Carlo (kMC) simulation is used to model the platinum reduction, nucleation, and nanoparticle growth [57]. Due to the sequential nature of the process, as shown in Figure 44a, the final conditions from the ODE adsorption simulation are used as initial conditions for the kMC growth simulation. The main assumption in the ODE-kMC coupled model is the independence between the process steps, that is, nanoparticles are not formed during the adsorption step and platinum precursor is not adsorbed in the growth step. The kinetic and thermodynamic parameters in this model are listed in Table 11.

The exact nanoscale mechanisms occurring during the process are not fully understood or quantified, and this is generally a challenge for the robust optimal processing of nanomaterials. For example, in this case study the release of the organic ligands might also occur during the adsorption phase, and there could also be nanoparticle nucleation occurring during adsorption. Despite the limitations of the existing process models, their construction and validation in conjunction with experiments encodes current understanding of the process and thus are needed for process engineering [34].

The simulations begin with the specification of the precursor mass (m_{PtOL}) and carbon nanotube mass (m_{CNT}), in the ranges listed in Table 11, while the rest of the parameters in the table are constant. Before the ODE adsorption process is simulated, the solubility of the platinum precursor in sc-CO_2 is evaluated, to determine the initial concentration of platinum in the supercritical fluid phase. Using reported experimental information for the solubility of the precursor in sc-CO_2 [7, 8, 160], a

Table 11: Model parameters for platinum nanoparticles on carbon nanotubes using sc-CO_2.

Operating Conditions		
Description	Variable	Value
Pressure [13]	P	24.2 MPa
Temperature [13]	T	343 K
Reactor volume [163]	V	54 cm^3
Precursor mass	m_{PtOL}	[130–170] mg
Carbon nanotube mass	m_{CNT}	[130–170] mg
Adsorption time	t_{ads}	2 h
Growth time	t_{gr}	2 h
Adsorption Isotherm		
Langmuir adsorption constant [13]	K_{eq}	0.299 $\frac{\text{g sc-CO}_2}{\text{mg PtOL}}$
Adsorption capacity [13]	Q_0	334 $\frac{\text{mg PtOL}}{\text{g CNT}}$
Kinetic Parameters		
Adsorption	k_{ads}	4.11×10^1 $\frac{\text{cm}^3}{\text{mol PtOL·s}}$
Desorption	k_{des}	3×10^{-4} $\frac{1}{\text{s}}$
Reduction	k_{red}	1×10^{-3} $\frac{1}{\text{s}}$
Nucleation	k_{nuc}	1×10^1 $\frac{\text{cm}^3}{\text{mol Pt}_1\text{·s}}$
Growth	k_{gr}	1×10^6 $\frac{\text{cm}^3}{\text{mol Pt}_1\text{·s}}$
Material Constants		
Molecular weight of precursor	MW_{PtOL}	333.34 $\frac{\text{g}}{\text{mol}}$
Molecular weight of active site	MW_{α}	45.0 $\frac{\text{g}}{\text{mol}}$

Chrastil model [20] was fit and then used to compute the initial precursor concentration. The fitted Chrastil model is

$$\ln\ S = 5.13\ \ln\left(\rho_{\text{sc-CO}_2}(T,P)\right) - \frac{9240}{T} + 31.6 \tag{143}$$

where S is the solubility limit at the process conditions in units of $\left(\frac{\text{mg PtOL}}{\text{cm}^3\ \text{sc-CO}_2}\right)$, and $\rho_{\text{sc-CO}_2}$ is the density of the sc-CO_2. The sc-CO_2 density depends on the temperature and pressure, and is computed using an equation of state [133]. If the precursor density is lower than S, all platinum precursor can be solubilized in the fluid phase and the initial concentration of platinum is $\frac{m_{\text{PtOL}}}{V}$, otherwise the maximum platinum precursor is limited by the solubility and the initial concentration is defined according to this limit.

The adsorption-desorption dynamics of the precursor on the carbon nanotubes in

the sc-CO_2 fluid can be modeled as

$$\text{PtOL}(f) + \alpha(s) \rightleftharpoons \text{PtOL-}\alpha(s)$$

where PtOL(f) is the platinum precursor in the fluid phase, $\alpha(s)$ is an active site on the carbon nanotube support, and PtOL-$\alpha(s)$ is the adsorbed platinum precursor on a carbon nanotube active site. The forward reaction is adsorption and the reverse reaction is desorption, with rate constants k_{ads} and k_{des}, respectively. Their ratio determines the Langmuir adsorption constant K_{eq} that is reported in Table 11.

The initial conditions for the platinum precursor concentration and active sites concentration must also be specified. After the solubility verification described previously, the initial concentration of the platinum precursor in the sc-CO_2 fluid is

$$C_{\text{PtOL},0} = \frac{m_{\text{PtOL}}}{\text{MW}_{\text{PtOL}}\ V} \tag{144}$$

where MW_{PtOL} is the molecular weight of the platinum precursor, and V is the reactor volume.

The initial concentration of the available active sites is related to the initial amount of carbon nanotube m_{CNT} in the system. Different functional groups (such as the carboxyl group -COOH) have been incorporated onto the CNTs to enhance their reactivity [157]. We assume here that the adsorption of platinum precursor only occurs at these functionalized sites and not directly on other sites on the carbon nanotube surface—however, this is in fact another open mechanistic question in the literature [157]. The initial concentration of active sites is thus dependent on the extent of functionalization of the CNT, and here we use a value of 4.5 w/w % of the carbon nanotube mass, based on typical commercially available values. Using this information, we can calculate the initial concentration of active sites on the CNT as follows

$$C_{\alpha,0} = \frac{0.045\ m_{\text{CNT}}}{\text{MW}_{\alpha}\ V} \tag{145}$$

where MW_α is the molecular weight of the functional group, which is a carboxyl group here. With the initial conditions $C_{\mathrm{PtOL},0}$, $C_{\alpha,0}$ and the adsorption time t_{ads}, the ODE adsorption model simulates the process dynamics and in the end predicts the final concentration of adsorbed platinum precursor on the carbon nanotube surface, $C_{\mathrm{PtOL}\alpha}\,(t = t_{\mathrm{ads}})$.

This value for the adsorbed precursor is then used as the initial condition for the kMC growth model. The kMC growth model is evaluated using a stochastic simulation—in particular, the Gillespie algorithm [39] is used to simulate individual chemical reactions as probabilistic events that occur at known rates. The kMC simulation includes three surface reactions on the CNT surface: thermal reduction of the adsorbed platinum precursor PtOL to release the organic ligands from the platinum atom, nucleation of a nanoparticle from two individual Pt atoms, and subsequent growth of platinum nanoparticles by the incorporation of additional Pt atoms. Specifically, these reactions take the form

$$\begin{aligned} \mathrm{PtOL\text{-}}\alpha(s) &\overset{k_{\mathrm{red}}}{\rightarrow} \mathrm{Pt}_1(s) + \mathrm{OL}(f) \\ 2\mathrm{Pt}_1(s) &\overset{k_{\mathrm{nuc}}}{\rightarrow} \mathrm{Pt}_2(s) + \alpha(s) \\ \mathrm{Pt}_n(s) + \mathrm{Pt}_1(s) &\overset{k_{\mathrm{gr}}}{\rightarrow} \mathrm{Pt}_{n+1}(s) + \alpha(s), \qquad n = 2, 3, \ldots \end{aligned} \tag{146}$$

where $\mathrm{Pt}_1(s)$ is an isolated platinum atom on the CNT surface, $\mathrm{Pt}_n(s)$ is a nanoparticle with n platinum atoms and $\mathrm{OL}(f)$ is the organic ligand released to the nitrogen fluid phase from the adsorbed platinum precursor. The kMC growth model assumes that platinum is released in its elemental state; two platinum atoms form a stable nucleus; a size-independent growth rate exists for the nanoparticles, and no aggregation between particles contributes to nanoparticle growth [147]. The number of nanoparticles of each size (Pt_n) thus defines the nanoparticle size distribution.

The rate constants for these reactions are listed in Table 11 and they define the

typical rates of reaction for the probabilistic events. Particular features of the distribution like mean and variance are related to the ratio between nucleation and growth rate [35], and their numerical values in Table 11 were estimated to fit the nanoparticle size distribution in Reference [13]. Each kMC simulation proceeds until no single platinum atoms are on the CNT surface or until the growth time t_{gr} is achieved. The kMC system size, defined by the number of platinum atoms used to simulate the overall growth step was 70000 atoms.

6.1.1 Implementation in model reduction of nanoparticle dynamics

Previously in Section 1.1, the problem of reducing the computational demands of the nanoparticle dynamics model was briefly introduced to frame this thesis work. In this chapter, this problem formulation is revisited and presented in more detail and, incorporating some of the definitions used in Chapter 5 for multivariate dynamic systems. As it was the case with the non-adiabatic CSTR case study in Chapter 5, the nanoparticle dynamic model is an initial value problem, where different selections of initial values of precursor m_{PtOL} and carbon nanotubes m_{CNT} mass loaded into the chamber, create different nanoparticle size distributions. Therefore, the initial sample region for this nanoparticle dynamics model is defined from Table 11 as

$$\begin{aligned} 130 \text{ mg PtOL} \leq m_{\text{PtOL}} \leq 170 \text{ mg PtOL} \\ 130 \text{ mg CNT} \leq m_{\text{CNT}} \leq 170 \text{ mg CNT} \end{aligned} \tag{147}$$

After defining the initial sample region in Equation (147), the next step in the description of the nanoparticle dynamics model is to define the dynamic region of it. For this case study, the dynamic region is build with a constant sampling time of $\Delta t = 200$ s and a final simulation time of $t_f = 7200$ s, for both the ODE adsoption model and the kMC growth model. The dynamic region of the nanoparticle dynamic model is made by exploring the initial sample region with multiple intial values selected by a Latin Hypercube. The major difference between the case studies in Chapters 5 and 6

is that the initial sample region and the dynamic region of the nanoparticle dynamic model are not in the same mathematical space, as it was in the non-adiabatic CSTR model. In the non-adiabatic CSTR case study, the variables that define the initial sample region were the same as the state variables that describe the CSTR dynamics. In the nanoparticle dynamic model, due to the coupled ODE-kMC structure of the model and the state reduction of the kMC variables to decrease its computational cost, the variables in the initial sample region and the dynamic region are completely different. The variables that define the initial sample region are the precursor and carbon nanotube mass m_{PtOL}, m_{CNT} whereas the variables used to define the dynamic region are the concentrations of nanoparticles at different sizes in the kMC growth simulation.

In the nanoparticle case study considered here, the coupled ODE-kMC simulation is approximated by an ODE-X simulation, where X represents one of mathematical approximate models evaluated in this chapter and described later on Section 6.2.1. Gaussian process models are among these approximate models for comparison of their approximated dynamic predictions. Figure 45 describes the major steps in the model reduction of the nanoparticle growth dynamics from the kMC simulation to the approximate models. In order to build these approximated models from the simulated data, a dimensional reduction is perform over the higher-order state variables $\mathbf{z}(s)$ of the kMC growth model, to map their information to the low-order and more affordable state variables of the approximate models $\mathbf{x}(s)$. The kMC simulation can be seen as a one-step-ahead recursive function, mapping information from one discrete time s to the next discrete time $s+1$, at a constant Δt. Five state variables are used in $\mathbf{x}(s)$ as a reduced state to describe the kMC growth simulation in Equation (146), that is $d = 5$. The first two states ($\mathbf{x}_1$ and $\mathbf{x}_2$) are the volumetric concentrations of platinum atoms (Pt_1) and adsorbed platinum precursor (PtOL-α) on the CNT surface. The remaining three state variables describe the nanoparticle size distribution Pt_n, using

a reduced state created from a moments approximation [5, 119]. The basic equation for the moments of a distribution is

$$m_q = \sum_{n=2}^{\infty} n^q N_n \tag{148}$$

where m_q is the qth moment of the nanoparticle size distribution, and N_n is the number of nanoparticles of size n in the kMC simulation.

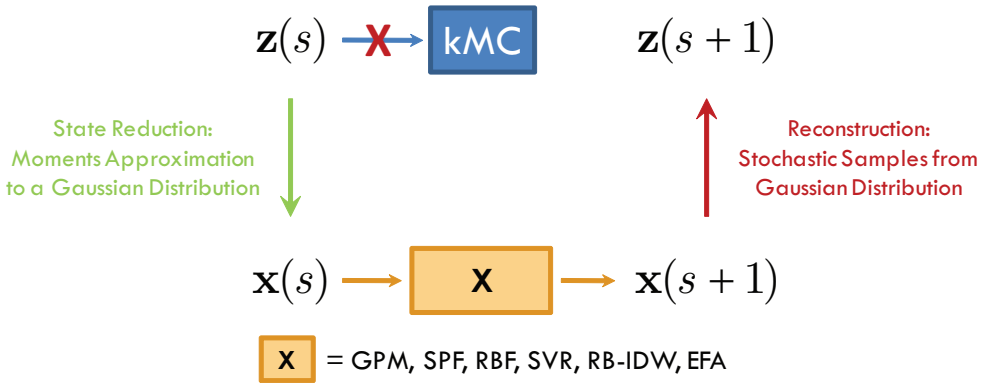


Figure 45: Description of the approximate modeling for the nanoparticle dynamic model as an expensive dynamic simulation. The figure describes the reduction and reconstruction steps to link the expensive dynamic simulations with its approximated dynamic model. X represents any of the evaluated mathematical approximated models, Gaussian process model (GPM), second-order polynomial functions (SPF), radial basis function (RBF), support vector regression (SVR), regression-based inverse distance weighting (RB-IDW), and equation-free approximations (EFA) (See Section 6.2.1).

The dynamics of the population balance defined in Equation (146) are infinite dimensional, while an approximated model must be built for each finite state that is modeled; therefore some approximation of the size distribution is required. If the nanoparticle size distribution followed a normal distribution across the state space at any discrete time s, it would only need the first three moments to completely describe the distribution. Specifically, the moments can be used to compute the mean μ and

variance σ^2 of a normal distribution.

$$\mu = \frac{m_1}{m_0} \tag{149}$$

$$\sigma^2 = \frac{m_2}{m_0} - \mu^2 \tag{150}$$

Here, these three moments are used as a state reduction step to approximate the nanoparticle size distribution $\mathbf{x}_{3-5}(s) = [m_0, m_1, m_2]$ in Equations (148) – (150) as a normal distribution, based on the observations of typical distributions such as the one shown in Figure 47b. Using the same Gaussian distribution assumption, it is possible to obtain a reconstruction of the original higher-dimension $\mathbf{z}(s)$ variables, by sampling from this distribution. A sampled reconstruction is generated from stochastic samples of the Gaussian distribution, until it reaches the kMC system size of 70000 atoms. The reconstruction step in this model reduction is not a one-to-one mathematical operation, contrary to the one-to-one state reduction step, because of the stochastic sampling in the Gaussian distribution. This stochastic characteristic of the reconstruction induces an uncertainty in the attempt of recover the original variables. Clearly the true size distribution cannot be exactly normal since negative nanoparticle sizes are not possible, so some error will be incurred with this reconstruction step. At early discrete times such as that shown in Figure 47a, the Gaussian distribution must be significantly truncated during a sampling ($n \geq 2$), since here the spread of the distribution is large relative to the mean.

6.2 Evaluation of data-driven models for nanoparticle dynamic predictions

The procedure to build approximate models for the nanoparticle dynamic model follows the same steps as in the non-adiabatic CSTR case study in Chapter 5. The initial sample set $\mathcal{D}$ is constructed from dynamic trajectories of the original ODE-kMC simulation, based on the exploration of the initial sample region in Equation (147). A ODE-kMC simulation is defined by an initial precursor mass m_{PtOL} and

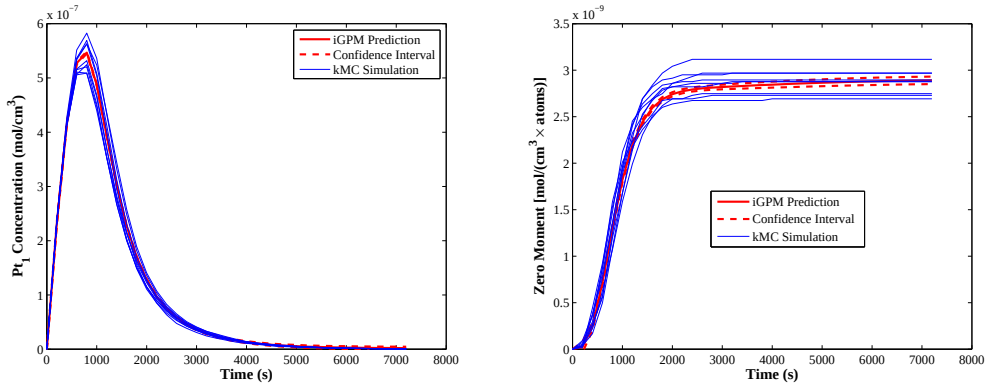
carbon nanotube mass m_{CNT} selected from the rectangular initial sample region. The ODE-kMC simulation is then run for $t_f = 7200$ s, using a constant sampling rate $\Delta t = 200$ s, corresponding to $n_s = 36$ discrete time steps. Twenty precollected dynamic trajectories $n_{dyn} = 20$ are simulated, four of them at the corners of the initial sample region, and the remaining sixteen to be selected using a Latin hypercube experimental design. The set of initial operating conditions maybe well-defined, but in a dynamic context, that does not guarantee that a dynamic region is well-defined, since the trajectories of this nonlinear stochastic system are not known ahead of time.

Once the dynamic trajectories have been collected, the reduced order state is created using the moments approximation, as it was described previously with Equation (148). Then, the reduced-order state variables are scaled in the range 0 - 1 to reduced the possible effects from the different orders of magnitude in the variables. To select the sample points that are used in the approximate model, the scaled reduced-order dynamic trajectories go trough a sparsification procedure as it is described in Section 5.2.2, with a grid spacing of $\Delta g = 0.05$. This sparsification procedure reduces the number of sample points in an approximate model from 720 sample points ($n_{dyn} \times n_s$) to 275 sample points approximately. With this reduced set of sample points, one of the approximate models listed in Figure 45 can be build as a one-step-ahead mapping function, and then used to describe the dynamics of the scaled reduced-order state variables.

An example of a predicted trajectory is shown in Figure 46. This dynamic prediction was made using an iGPM approximate model, by building $d = 5$ independent GPM models. Each of the GPM models used the Gaussian correlation function in Equation (9), a constant regression function and the maximum likelihood approach to estimate their parameters. Two of the five dynamic states are plotted versus time, beginning from an initial sample point defined by a precursor mass of $m_{\mathrm{PtOL}} = 155$ mg and a carbon nanotube mass of $m_{\mathrm{CNT}} = 140$ mg. Since the reactor volume V is fixed,

these two species along with the parameters in Table 11 define the initial state of the system and the dynamics. The trajectories in Figure 46 compare the results of 10 individual stochastic realizations of the ODE-kMC model (blue) to the prediction by the ODE-iGPM model (red). Since each of the GPM computes both the mean prediction and prediction variance, it is possible to plot both the mean value and the uncertainty. The dashed red line denotes $\hat{y} \pm z_{95}\sqrt{\sigma_y^2}$ where z_{95} is the normal score for a 95% confidence interval.

The stochastic realizations shown in Figures 46a and 46b for the reduced-order state variables illustrates the non-constant noise level during the dynamic trajectory in the nanoparticle dynamic model. Along this thesis, the implementation of Gaussian process models for dynamic predictions has always considered a constant noise level in the observations, by estimating a single parameter value for this purpose σ_u^2. The nanoparticle dynamic model is a good case study to discuss how GPM handles this particular noise structure, and its implication for the parameter estimation, dynamic predictions and error estimation properties of the model.

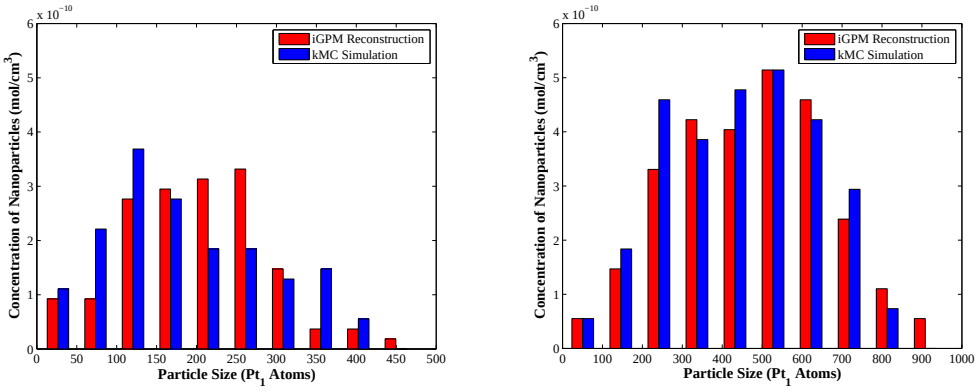


(a) (b)

Figure 46: Approximated nanoparticle dynamic trajectory using a iGPM approximate model. $m_{\text{PtOL}} = 155$ mg, $m_{\text{CNT}} = 155$ mg; (a) concentration of elemental platinum on the CNT surface; (b) zero moment of the nanoparticle size distribution m_0.

The nanoparticle size distribution is plotted in Figure 47, at two representative

times. The blue bars are histograms coming directly from a single stochastic realization of the original simulation, using the same precursor and nanotube masses as in Figure 46. In contrast, the red bars are based on the predictions from the iGPM. Because the iGPM only predicts the first three moments of the nanoparticle size distribution, these moments are then used to construct a Gaussian distribution. The red bars in Figure 47 are histograms resulting from stochastic samples from this Gaussian distribution, until it reaches the kMC system size of 70000 atoms. Figure 47a corresponds to a time early in the process, and neither histogram appears smooth due to the small number of particles in the simulation domain. At the later time in Figure 47b, both histograms appear smoother and consistent with a Gaussian distribution. At both times, the mean and variance of the GPM histogram appears consistent with that of the original simulation.



(a) (b)

Figure 47: A realization of the nanoparticle size distributions at two different process times; (a) $t = 1000$ s (b) $t = 7200$ s. The iGPM realization is drawn from a continuous distribution with mean and variance according to Equations (149) and (150).

The plots in Figures 46 and 47 are illustrative examples of how to use a approximate model to describe the nanoparticle dynamics model. Following the same scheme, other approximate models can be used as iterative mapping functions for dynamic predictions. The nanoparticle dynamic model provides a case study for a systematic comparison of several approximate models for multivariate, stochastic and dynamic

complex simulations. The list of approximate models evaluated in this study includes models with a background in statistics, such as Gaussian process models, and models developed in the areas of artificial intelligence and machine learning, such as support vector regression. This difference in the models will limit the comparison between the approximate models to their mean prediction and not to evaluate their error estimation properties. The next section provides a brief description of the approximate models included in this study, except iGPM, which it was discussed extensively in Chapter 5.

6.2.1 Mathematical description of metamodeling approaches

6.2.1.1 Second-order polynomial regression functions

Polynomial basis functions have been used over the years in many fields in engineering and sciences in general. Their straightforward parameter estimation, simple interpretation and convenient mathematical properties make these models one of the most utilized. Despite several non-linear models that have appeared as alternatives for data mining, linear models based on polynomial basis functions remain as a reference for comparison and analysis. In this case study, a second-order polynomial function (from now on SPF) is used as an iterative mapping function to approximate the dynamics of the reduced state variables in the nanoparticle dynamic model.

Given a dataset of sample points $\mathcal{D}$ of n input/output points $\{\mathbf{x}_i, y_i\}$, $y_i \in \mathbb{R}$, $\mathbf{x}_i \in \mathbb{R}^d$ a second-order polynomial model predicts the outcome $\hat{y}(\mathbf{x})$ as

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_j x_j + \sum_{i=1}^{d} \sum_{j \geq i}^{d} w_{ij} x_i x_j \tag{151}$$

where all w parameters represent coefficients for each of the polynomial basis functions. The most common methodology to estimate the w_i coefficients is by minimizing the sum of the squared errors at each $\mathbf{x}_i$ in $\mathcal{D}$. For the approximation of the nanoparticle dynamic model, $d = 5$ independent second-order polynomial models are estimated, one for each of the reduced state variables.

One second-order polynomial model with $d = 5$ has 21 coefficients to be estimated for the pre-collected dynamic trajectories, giving a total of 105 parameters for the complete reduced state prediction. This number of coefficients largely surpasses most of the approximate models compared in this case study (for example, iGPM has 35 parameters for the same complete reduced state prediction). A stepwise regression procedure [58] has been used to reduce the number of estimated coefficients in each of the second-order polynomial models, while preserving the statistical identification of the available data. Despite the potential extrapolation problems that a polynomial model could suffer (see Section 5.4.1.2), these models are included in the comparison of approximate models to represent a global trend approach for dynamic predictions, in contrast with a localized approach such as iGPM. There are few cases where polynomial basis functions, by themselves, are used in an iterative mapping framework to describe a system's dynamics [51]. Generally, polynomial basis functions are combined with other methodologies to obtain a more robust representation of a continuous spatio-temporal dynamics.

6.2.1.2 Radial basis functions

Radial basis functions (from now on RBF) have been used for over 30 years since their appearance in the 1970s, as a methodology to represent irregular surfaces in topography [54]. Radial basis functions have been employed as empirical models in many research fields including multi-objective optimization [18], metamodeling [130], bioinformatics [136] and classification [104]. A radial basis function approximates a function using a linear combination of radial functions spread across the input space. Given a dataset of sample points $\mathcal{D}$ of n input/output points $\{\mathbf{x}_i, y_i\}$, $y_i \in \mathbb{R}$, $\mathbf{x}_i \in \mathbb{R}^d$, a RBF is constructed as

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{n} w_i r_i(\mathbf{x}) \tag{152}$$

where $r_i(\mathbf{x})$ represents a radial function, and w_i represents the weight in each of the radial functions. The radial function $r_i(\mathbf{x})$ is usually a monotonically decreasing function relative to the distance of a central point [85]. In Equation (152), each of the radial functions has been centered around one of the $\mathbf{x}_i$ sample points of the dataset $\mathcal{D}$, to maximize the coverage across the input space. Many radial functions can be used in RBF, but for the fairest comparison of this model with other approximate models in this study, a Gaussian radial basis function [130] similar to Equation (9) is being used.

$$r_i\left(\mathbf{x}\right) = \exp\left[-\frac{1}{2}\sum_{a=1}^{d}\frac{\left(x_a - x_{i,a}\right)^2}{\ell_a^2}\right] \tag{153}$$

where $\mathbf{x}_i$ is an input point in $\mathcal{D}$

The set of parameters to be estimated in the RBF are the weights w_i for each of the radial functions, as well as the set of range parameters $\boldsymbol{\theta} = [\ell_1, \ldots, \ell_d]$ in Equation (153). In this case study, the estimation of the weights for the RBF is done using the ridge regression approach. This approach minimizes the sum of squared prediction errors of the model, with a regularization term that accounts for potential ill-posed formulated problems. RBF is known for having similar ill-conditioning problems as GPM. The parameter estimation of the weights w_i using the ridge regression approach minimizes the function $\mathcal{C}\left(\mathbf{w}, \boldsymbol{\theta}, \gamma\right)$

$$\mathcal{C}\left(\mathbf{w}, \boldsymbol{\theta}, \gamma\right) = \left(\mathbf{y} - R\mathbf{w}\right)^T\left(\mathbf{y} - R\mathbf{w}\right) + \sqrt{\gamma}\ \mathbf{w}^T\mathbf{w} \tag{154}$$

where $\mathbf{w} \in \mathbb{R}^n, \mathbf{w} = [w_i, \ldots, w_n]$ is the vector of weights in the RBF, $\mathbf{y} \in \mathbb{R}^n, \mathbf{y} = [y_i, \ldots, y_n]$ is the vector of output values in the dataset $\mathcal{D}$, and $R \in \mathbb{R}^{n\times n}, R_{ij} = r_i\left(\mathbf{x}_j\right)$, is a matrix that contains each of the radial functions evaluated at each of the sample points in the dataset $\mathcal{D}$. The regularization parameter γ is included in this cost function to alleviate the ill-conditioning problems in the inversion of the matrix R. The estimated weights for the RBF $\hat{\mathbf{w}}$ as a function of $\boldsymbol{\theta}$ and γ, can be obtained

analytically from the minimization of the cost function in Equation (154).

$$\hat{\mathbf{w}}\left(\boldsymbol{\theta},\gamma\right)=\left(R^{T}R+\gamma I\right)^{-1}R^{T}\mathbf{y} \tag{155}$$

where $I\in\mathbb{R}^{n\times n}$ is the identity matrix.

Most of the RBF implementations leave the values of $\boldsymbol{\theta}$ and γ to be selected by the user. In this case study these values are selected via a nonlinear constrained optimization that minimize the mean square cross-validation error ($MSCV$) [65, 102]. A leave-one-out cross-validation error computes the prediction error between the output value y_i and the predicted value $\hat{y}\left(\mathbf{x}_i\right)$ of a RBF where the information of the sample point $\{\mathbf{x}_i,y_i\}$ has been left out on purpose from the estimated $\hat{\mathbf{w}}$. The mean square cross-validation error of a RBF is the mean of the leave-one-out cross-validation errors at each of the n sample points of the dataset $\mathcal{D}$. According to this error metric, the nonlinear constrained optimization used to estimate $\boldsymbol{\theta}$ and γ is

$$\begin{aligned}\min_{\boldsymbol{\theta},\gamma}\ & MSCV\left(\boldsymbol{\theta},\gamma\right)=\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\hat{y}^{RBF}\left(\mathbf{x}_i\right)\right)^2\\ &\text{subject to}\\ &1\times10^{-3}\leq\ell_a\leq1\quad i=1,\ldots,d\\ &1\times10^{-9}\leq\gamma\leq1\times10^{9}\end{aligned} \tag{156}$$

where $\hat{y}^{RBF}\left(\mathbf{x}_i\right)$ represents the RBF prediction in Equation (152) without the information of the i^{th} sample point.

Radial basis functions have been used extensively to represent system dynamics. Early implementations of RBF in system identification were made by Elanayar and Shin [32]. More recently, RBF have been used as a meshless method to solve partial differential equations [162], as part of a model predictive control schemes for unstable nonlinear chemical processes [148] and distributed parameter systems [4]. In contrast with the previous deterministic examples, this case study build RBF using data from a stochastic dynamic model.

6.2.1.3 Support vector regression

Support vector machine is an classification model developed in the machine learning community by Vapnik and cooworkers [15] while they were working on a separable bipartition problem at AT&T Bell Laboratories. The original formulation of support vector machines creates a mathematical hyperplane boundary that classifies data points in two separate and different groups of data. Later, Vapnik extended this classification model to create support vector regression (from now on SVR) [31], by understanding that a regression model is also a hyperplane that divides the set of outcomes whose distance margin to this mathematical boundary is minimized. Given a dataset of sample points $\mathcal{D}$ of n input/output points $\{\mathbf{x}_i, y_i\}$, $y_i \in \mathbb{R}$, $\mathbf{x}_i \in \mathbb{R}^d$, a SVR can be constructed as

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{n} w_i r_i(\mathbf{x}) + b \tag{157}$$

where $\mathbf{w} \in \mathbb{R}^n, \mathbf{w} = [w_i, \ldots, w_n]$ and $b \in \mathbb{R}$ are coefficients to be estimated. In SVR, the radial function $r_i(\mathbf{x})$ in Equation (153) has a completely different meaning as in the RBF model. Here, the radial functions work as mapping functions from the original input space $\mathbb{R}^d$ to a higher dimensional feature space $\mathbb{R}^n$. The rationale for doing this is that it may be easier to create a linear mathematical hyperplane to separate the output information in a high dimensional feature space, than creating a non-linear mathematical hyperplane in the original input space. This type of mathematical mapping procedure is quite common in the machine learning community and it is known as *kernalization*. Because the usage of the radial basis functions as kernel functions in SVR, the range parameters $\boldsymbol{\theta} = [\ell_1, \ldots, \ell_d]$ in the radial functions must also be estimated.

The mathematical theory behind the estimation of the $\mathbf{w}$ vector coefficients and the b bias term in SVR is well documented [12, 50, 132]. In summary, the estimation of these coefficients is made by solving a quadratic programming optimization

problem, after including slack variables to account for the misclassification of the output information, creating a dual optimization problem with Lagrange multipliers and evaluating the Karush-Kuhn–Tucker (KKT) conditions that must be satisfied by the optimal solution. Instead of using the traditional ϵ-insensitive loss function developed by Vapnik in its original SVR formulation [31], the SVR parameter estimation in this study uses a quadratic loss function [50], to keep consistency with all the loss functions in the different approximate models of this chapter. The quadratic programming problem used to estimate the w_i coefficients as a function of $\boldsymbol{\theta}$ and γ in SVR is

$$\begin{aligned} \min_{\mathbf{w}(\boldsymbol{\theta},\gamma)} \mathcal{C}\left(\mathbf{w},\boldsymbol{\theta},\gamma\right) &= \frac{1}{2}\mathbf{w}^T R\mathbf{w} - \mathbf{w}^T\mathbf{y} + \gamma\ \mathbf{w}^T\mathbf{w} \\ &\text{subject to} \\ \sum_{i=1}^{n} w_i &= 0 \end{aligned} \tag{158}$$

where the definitions of $R, \mathbf{y}$ and γ were presented previously for the RBF. From Equation (158), the estimated weight vector $\hat{\mathbf{w}}$ and estimated bias $\hat{b}$ to be used in the SVR prediction in Equation (157) are

$$\hat{\mathbf{w}}\left(\boldsymbol{\theta},\gamma\right) = \arg\min_{\mathbf{w}(\boldsymbol{\theta},\gamma)} \mathcal{C}\left(\mathbf{w},\boldsymbol{\theta},\gamma\right) \tag{159}$$

$$\hat{b} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{n}\hat{w}_j r_j\left(\mathbf{x}_i\right)\right) \tag{160}$$

Like in the case of the RBF model, the additional parameters $\boldsymbol{\theta}$ and γ for SVR are estimated using the mean square cross-validation error $MSCV$, leaving one sample point $\{\mathbf{x}_i, y_i\}$ at a time. The nonlinear constrained optimization used to estimate $\boldsymbol{\theta}$

and γ in SVR is

$$\min_{\boldsymbol{\theta},\gamma} MSCV(\boldsymbol{\theta},\gamma) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}^{SVR}(\mathbf{x}_i)\right)^2$$
$$\text{subject to} \tag{161}$$
$$1\times 10^{-3} \leq \ell_a \leq 1 \quad i = 1,\ldots,d$$
$$1\times 10^{-9} \leq \gamma \leq 1\times 10^{9}$$

where $\hat{y}^{SVR}(\mathbf{x}_i)$ represents the SVR prediction in Equation (157) without the information of the i^{th} sample point.

Despite that SVR is a fairly recent empirical model, its success has already reached other research fields outside the machine learning community. SVR has already been used as a state estimator in a model predictive flight control [130], as a approximate model for the dynamic description of financial time series [141], as a system identification methodology for Wiener models [140], and as an approximated dynamic model for a tubular reactor [116]. In the machine learning community, SVR and GPM are considered the current state-of-the-art for artificial intelligence and data mining. Despite that both empirical models are highly adaptive thanks to their non-parametric characteristics, most authors will argue that SVR has significant advantages over GPM.

The current algorithms for solving quadratic programming problems are an advantage for SVR, against the potential local optimal problems during the MLE methodology in GPM. Additionally, SVR could use only a subset of the sample points in $\mathcal{D}$ when the ϵ-insensitive loss function is employed, compare to the GPM implementation where all sample points are always used. However, the robust statistical framework of GPM provides error estimations that SVR cannot, a trait that has been explored and exploited in this thesis work. To the best of my knowledge, there has not been a comparison between these two empirical models in the context of system dynamics.

6.2.1.4 Regression-based inverse distance weighting

Inverse distance weighting is an empirical model for interpolation proposed in the late 1960s [129], which uses a weighted average of the observations y_i to make its prediction. The original version of the inverse distance weighting uses the Euclidean distance as a inverse metric to define the weights of each observation in the model, a much simple formulation compare with some of the most recent empirical models like GPM, RBF or SVR. The attractive feature of the inverse distance weighting is that it does not require the inversion of a —sometimes ill-conditioned— matrix. As was mentioned previously in this thesis, the inversion of covariance matrix V in GPM or the inversion of the R matrix in RBF and SVR, is the common bottleneck of these models. Recently, Joseph and Kang [65] has proposed an alternative version of the inverse distance weighting, by including a linear regression model as part of its prediction (from now on RB-IDW), with the idea that many complex systems in spatial statistics and computer experiments have global trends.

Given a dataset of sample points $\mathcal{D}$ of n input/output points $\{\mathbf{x}_i, y_i\}$, $y_i \in \mathbb{R}$, $\mathbf{x}_i \in \mathbb{R}^d$, a RB-IDW can be constructed as

$$\hat{y}(\mathbf{x}) = \mu\left(\mathbf{x}, \boldsymbol{\beta}\right) + \frac{\sum_{i=1}^{n} w_i(\mathbf{x}) e_i}{\sum_{i=1}^{n} w_i(\mathbf{x})} \tag{162}$$

where $\mu\left(\mathbf{x}, \boldsymbol{\beta}\right)$ is a linear regression model that capture the global trend of the observations, $e_i \in \mathbb{R}$ is the error between the regression model at the sample points $\mathbf{x}_i$ and the observation y_i, $e_i = y_i - \mu\left(\mathbf{x}_i, \boldsymbol{\beta}\right)$. Last, the weight coefficients $w_i(\mathbf{x}) \in \mathbb{R}$ represent the contribution from the inverse distance weighting model to account for possible local effects in the prediction. Joseph and Kang [65] also proposed to use a new weighting function for their RB-IDW model, which has some similarities with the radial basis functions used in GPM, RBF, and SVR

$$w_i\left(\mathbf{x}, \boldsymbol{\theta}\right) = \frac{\exp\left[-\sum_{a=1}^{d} \frac{1}{2} \frac{\left(x_a - x_{i,a}\right)^2}{\ell_a^2}\right]}{\sum_{a=1}^{d} \frac{1}{2} \frac{\left(x_a - x_{i,a}\right)^2}{\ell_a^2}} \tag{163}$$

where $\boldsymbol{\theta} = [\ell_1, \ldots, \ell_d]$ are the range parameters that control the distance effects in each of the input space dimensions.

According to this mathematical description, a RB-IDW requires a linear regression for $\mu(\mathbf{x}, \boldsymbol{\beta})$ and an estimate of $\boldsymbol{\theta}$ for the weighting function. The selection of a regression model for RB-IDW could be done with any variable selection technique in statistics. For example, the second-order polynomial function described previously could be used in the RB-IDW to capture global trends. However, in this case study, the main interest is in evaluating the local behavior in the prediction of the RB-IDW that makes it comparable to GPM, RBF, and SVR. Therefore, the RB-IDW will use a constant regression function, $\mu(\mathbf{x}, \boldsymbol{\beta}) = \boldsymbol{\beta}$, and the estimated value of the coefficient $\boldsymbol{\beta}$ is equal to the mean of the output observations y_i. Lastly, the estimation of the range parameters $\boldsymbol{\theta}$ in the weighting function $w_i(\mathbf{x})$ is made with a nonlinear constrained optimization using the $MSCV$ error of the residuals between the y_i observations and the estimated $\hat{\boldsymbol{\beta}}$. Such constrained optimization to estimate $\boldsymbol{\theta}$ in RB-IDW is

$$\begin{aligned} \min_{\boldsymbol{\theta}} \; & MSCV(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(e_i - \hat{e}^{IDW}(\mathbf{x}_i)\right)^2 \\ & \text{subject to} \\ & 1 \times 10^{-3} \leq \ell_a \leq 1 \quad i = 1, \ldots, d \end{aligned} \tag{164}$$

where $\hat{e}^{IDW}(\mathbf{x}_i)$ represents the IDW prediction of the residual e_i without the information of the i^{th} sample point.

6.2.1.5 Equation-free modeling

Equation-free modeling is not a metamodel like any of the previous empirical models presented in this chapter. Equation-free modeling is often presented as a modeling paradigm in multiscale computation, where the time evolution of a macroscopic, coarse-level state variable is described as a function of the time evolution of a microscopic, fine-level state variable [37, 137]. The reason to take such approach is that

"the methodology bypasses the derivation of macroscopic evolution equations when these equations conceptually exist but are not available in closed form, hence the term equation-free" [68].

According to this description, an equation-free approximation (from now on EFA) can be implemented to model the dynamics of the low-order, macroscopic, coarse-level variables $\mathbf{x}(s)$ using the high-order, microscopic, fine-level variables $\mathbf{z}(s)$ in the nanoparticle dynamic model. Figure 48 is a representation of EFA for the case study in this chapter. The coarse time-stepper, which is the essential tool in the EFA methodology, involves three steps: lifting, fine-level simulation and restriction. Lifting $\mu(\mathbf{x}(s))$ and restriction $\mathcal{M}(\mathbf{z}(s))$ are two functions that map the information between the multiscale levels of the simulation. In practice, there are not generic lifting/restriction functions that will work for all systems where EFA is used, these functions heavily depends on the mathematical understanding of the system to be simulated. In this case study, the sampling procedures for reconstruction and the moments approximation for reduction explained in Figure 45 will work as the lifting/restriction functions of the EFA. Due to the stochastic nature of the reconstruction step and the kMC dynamic simulation, it is necessary to perform repetitions of the EFA methodology at each evaluated $\mathbf{x}(s)$. The EFA dynamic prediction at $\mathbf{x}(s+1)$ is created by averaging the results from 5 repetitions of the coarse time-stepper (a repetition means reconstruction of the nanoparticle size distribution, moving the distribution in time with the kMC simulation, and reducing the state with the moments approximation).

In this case study, equation-free modeling is not used as a methodology to improve the computational time of expensive dynamic simulations. However, researchers have explored the possibility of using EFA to accelerate the macroscopic dynamic model using a coarse projective integration [37, 68]. Figure 48 also reveals why EFA can be compared with the approximate models previously described in this chapter. While

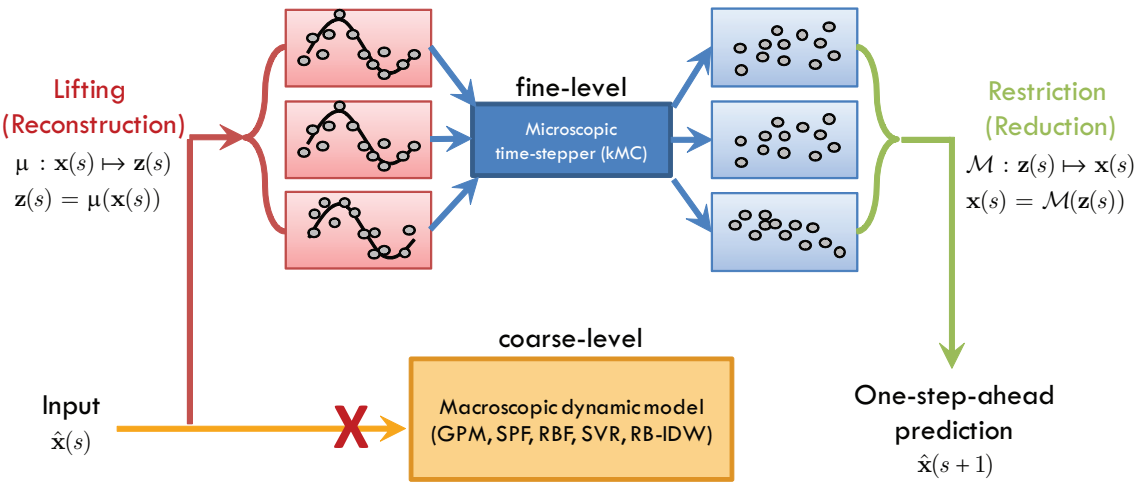


Figure 48: Description of the equation-free modeling. The figure shows the relationship between the lifting / reconstruction and the restriction / reduction steps of the equation-free modeling, applied to the nanoparticle dynamic model.

EFA assumes a *equation-free* modeling for the dynamics of the state variables $\mathbf{x}(s)$, the remaining approximate models propose a *close-form equation* to represent these dynamics, at the cost of potentially using the erroneous mathematical structure. Since EFA assumes that the fine-level dynamic model is an accurate representation of the system, it reasonable to think that the prediction error in EFA is because *the errors in the reconstruction and reduction steps*. Based on this, EFA allows this case study to compare the state reduction errors in an expensive dynamic simulation with the prediction errors of the metamodels.

6.2.2 Research Analysis

The analysis in this chapter is focused on the comparison between different metamodels as potential approximate models for multivariate stochastic dynamic predictions under the iterative mapping framework.

As it has been done throughout this thesis, the research analysis for dynamic models is based on the one-step-ahead prediction error LEM, Equation (110) and the global prediction error of a dynamic trajectory DEM, Equation (111). For clarification, all LEM and DEM values are calculated for the prediction of the scaled

reduced state variables of the kMC simulation, and not for the full-state variables $\mathbf{z}(s)$ or even the reduced state varianles $\mathbf{x}(s)$. Therefore, the LEM and DEM values of the metamodels are not directly influenced by the state reduction procedure, except for the equation-free approximations, where most of its prediction error is related to this mathematical transformation. A test dataset of 200 dynamic trajectories is constructed as reference for the DEM calculations. The initial sample values of these 200 trajectories are located in the initial sample region, Equation (147), and selected using a Latin hypercube approach. The sampling rate used in these trajectories is $\Delta t = 20$ s and the final time for the kMC growth simulation is $t_{gr} = 7200$ s as it appears in Table 11.

Each of the 200 dynamic trajectories is simulated 100 times to capture the mean behavior of the stochastic kMC simulation. Each of these kMC repetitions is first reduced using the moments approximation, and then averaged at each discrete time step to derive the reference mean dynamic trajectories in the DEM prediction error. The selected test sample points for LEM are chosen by a sparsification procedure using all 200 mean dynamic trajectories in the DEM test dataset, with a grid spacing of $\Delta g = 0.01$. Approximately, 2000 sample points are used in the LEM calculations, spread across the five-dimension dynamic region. For any finite sample size, the sample mean dynamic trajectory will fluctuate from the true mean kMC behavior, and this will of course limit the minimum value of the DEM and LEM that can be achieved. However, by using a fixed value, this case study is able to use the LEM and DEM to compare the performance of various approximate models in this chapter.

The comparison of approximate models considers 6 different mathematical models: iGPM, SPF, RBF, SVR, RB-IDW, and EFA. From all these models, the first five will correspond to a metamodeling approach where pre-collected information is used to generate a recursive mapping function, while the last one bypasses the construction of a close-form equation. For the case of the metamodels, the idea is to create d

independent metamodels to predict each of the scaled reduced state variables $\mathbf{x}(s)$. Each of the metamodels is build using a set of $n_{dyn} = 20$ dynamic trajectories, and a sparsification with $\Delta g = 0.05$. The initial sample points for the dynamic trajectories are selected using a Latin hypercube approach. There are some specific considerations in this comparison study:

1. There is not repeated information in the datasets used to build each of the different metamodels. This is in contrast to some previous work also focused in compared metamodeling approaches using stochastic observations [85], in which the average value of the repetitions is used as the training target. This study does not aid the predictions of the metamodels by using average values of the stochastic dynamic information.

2. No additional sample points can be added to the database $\mathcal{D}$ for any of the metamodels. This condition is explicitly stated here, to justify why a common methodology in the area of fast function evaluations like *in-situ* adaptive tabulation (ISAT) [114] is not used in this study. The adaptive component of ISAT will require additional function evaluations of the nanoparticle dynamic model to enrich its database $\mathcal{D}$ for tabulation. While there are implementations of sequential procedures that add information for a dynamic GPM framework [56], such mathematical implementations have not been found for the remaining metamodels.

3. The parameter methodology to estimate the GPM has been changed from the recurrent MLE methodology to the cross-validation approach. The reason for this change is the substantial difference in the computational cost of parameter estimation that could favor the GPM over the rest of the metamodels.

4. The comparison between the different metamodels is based on LEM prediction errors, DEM prediction errors, CPU Time for the parameter estimation

and CPU Time for a one-step-ahead prediction in the five-dimensional dynamic region. Other performance metrics for the evaluation of computationally-expensive black-box functions can be found in [126].

6.3 Results

6.3.1 Comparison of metamodeling approaches for mean dynamic predictions in nanoparticle synthesis

The last part of this results section is the comparison of data-driven models for approximate dynamic predictions of the nanoparticle dynamic model. To the best of my knowledge, this is the first systematic comparison between metamodels to describe system dynamics as an iterative mapping function. Therefore, the comparison will evaluate their one-step-ahead prediction error (LEM) and their global prediction error (DEM). To provide robustness to the study, 100 different experimental designs have been using in each of the selected metamodels for dynamic prediction. The one-step-ahead prediction error considers nearly 2000×100 prediction errors in each of the metamodels. Then, average values of the prediction errors can be computed. Table 12 summarizes the results from these computations. The first element to noticed in Table 12 is that equation-free approximations have major difficulties for the one-step-ahead prediction error with only a percentage error of 0.57 %, which still a small value. More important is to see that EFA has a poor performance because its large prediction errors on the three moments derived from the nanoparticle size distribution $[\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5]$. These are the three reduced state variables that are directly involved in the reconstruction / reduction mechanism of the EFA. Potentially, by increasing the number of repetitions required in the EFA for its prediction, those prediction errors could decrease significantly.

The performance of RB-IDW in the one-step-ahead prediction error is similar to the EFA results. In principle, RB-IDW should not be used to model the stochastic nanoparticle dynamic model because its characteristics as an interpolator. But, the

Table 12: Mean values of $\log_{10}(LEM)$ over the test samples for the nanoparticle dynamics model. The table summarizes the LEM results in each of the five reduced state variables as well as the overall LEM prediction of each approximate model. The LEM values were calculated over 100 different experimental designs, each of them with a $n_{dyn} = 20$ dynamic trajectories and a grid spacing $\Delta g = 0.05$.

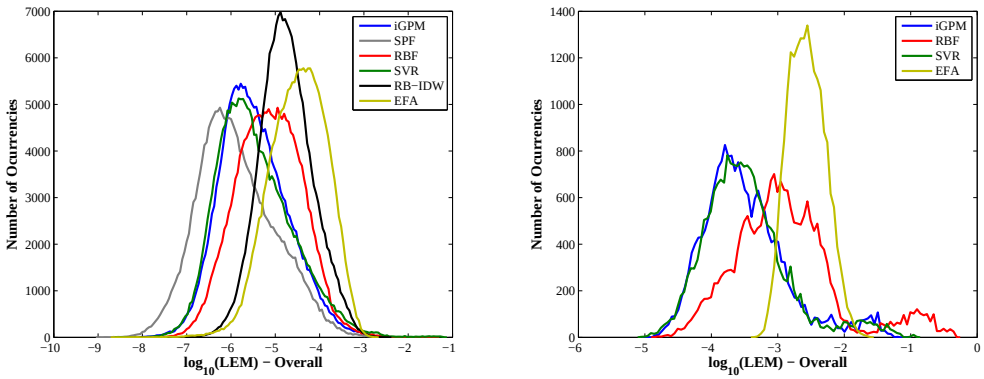
Model	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$	$\mathbf{x_5}$	**Overall**
iGPM	-6.0172	-6.8450	-5.3611	-6.5489	-6.7201	-5.4743
SPF	-6.1269	-8.9813	-5.9249	-7.1274	-7.2014	-5.8747
RBF	-6.1890	-6.5998	-5.1638	-5.8573	-6.1690	-5.1383
SVR	-6.1037	-6.7589	-5.4575	-6.4467	-6.5558	-5.4585
RB-IDW	-5.5256	-6.5984	-4.7884	-5.3845	-5.3486	-4.7532
EFA	-6.5194	-8.2556	-4.1748	-5.3958	-5.1870	-4.4854

results in this table indicate that RB-IDW has similar prediction error performance as the iGPM, SVR and RBF for the prediction of $\mathbf{x}_2$, the scaled concentration of active sites on the carbon nanotube surface. The reason for this is that this particular reduced state variable has the smallest noise level across the dynamic region. The best performance of all the metamodels in this one-step-ahead prediction error is for the SPF, closely followed by the iGPM and SVR, with a percentage error of 0.12 %. Perhaps one of the reason for this outstanding behavior is the large number of available regression functions that are screened during its construction.

Table 13 summarizes the results regarding the dynamic predictions of each of the metamodels. First, as in Chapter 5, the presence of global trend functions like SPF lead to a potential extrapolation problem during its iterative mapping. Also, the interpolation characteristics of the RB-IDW, where the model begins to interpolate noise observations in regions with high-noise levels, leads the RB-IDW to generate extrapolation problems. In the end, from the four remainder metamodels, iGPM is exhibits the best performance in terms of DEM, closely followed by the SVR metamodel. To provide a more robust representation of the results in these tables, Figure 49 shows the LEM and DEM distributions over the different test sample points and dynamic trajectories for the different metamodels.

Table 13: Mean values of $\log_{10}(DEM)$ over the test samples for the nanoparticle dynamics model. The table summarizes the DEM results in each of the five reduced state variables as well as the overall DEM prediction of each approximate model. The LEM values were calculated over 100 different experimental designs, each of them with a $n_{dyn} = 20$ dynamic trajectories and a grid spacing $\Delta g = 0.05$. The asterisk represents DEM prediction errors where extrapolation problems have occurred.

Model	x_1	x_2	x_3	x_4	x_5	Overall
iGPM	-4.1668	-5.1500	-3.2664	-3.7326	-3.4900	-3.4939
SPF	-4.3967*	-7.1003*	3.5441*	-4.4567*	3.8685*	-3.8369*
RBF	-4.0692	-4.6837	-2.7381	-2.9337	-2.7476	-2.8949
SVR	-4.2276	-5.0854	-3.3083	-3.7230	-3.4447	-3.4857
RB-IDW	-4.1014*	-5.0454*	-3.2062*	-3.5689*	-3.3367*	-3.3947*
EFA	-3.7133	-6.9880	-2.5825	-2.5572	-2.1922	-2.5668



(a) (b)

Figure 49: LEM and DEM prediction error distributions in the prediction of different metamodels for the nanoparticle dynamic model. Both figures were constructed from the results of 100 different experimental designs, each of them with $n_{dyn} = 20$ dynamic trajectories and $\Delta g = 0.05$.

The ultimate goal in building approximate models for expensive dynamic simulations, is to reduce the computational cost of these function evaluations. So far, the best prediction performance has been made by the non-parametric models, iGPM and SVR. Table 14 summarizes the computational costs of building each of these mathematical models. The implementation of a cross-validation approach increases the computational cost for parameter estimation of iGPM, RBF and SVR, compared to SPF. To give an idea in the increase of the computational cost for parameter

estimation due to cross-validation, an iGPM using the MLE parameter estimation methodology takes approximately 88 seconds to be completed, or 0.8 % of the computational time employed in the cross-validation approach. The major reason for this significant increase in the computational time is in the inversion of the matrices that locally correlates the information in each of these models. Once the parameter estimation is completed, the speed-up obtained by the metamodeling approaches is significant compared to the original kMC simulation. A kMC simulation for one of the dynamic trajectories takes approximately 55 seconds to be completed, compared to 0.03 seconds that will take an iGPM to perform the prediction of a mean dynamic trajectory, a computational reduction of nearly 99.9 %, with a percentage error in the dynamic trajectory of 1.79 %.

Table 14: Computational cost of different approximate models for the nanoparticle dynamics model. The tables shows the average CPU time in seconds for the parameter estimation and prediction of the approximate models, as well as the different parameter estimation methodologies in each of them. The CPU time calculations were computed with a Intel®Core^{TM} 2 @ 2.4 GHz, Matlab Version R2009b.

Model	**Parameter Estimation**	**CPU Time (s)**	
		Estimation	**Prediction**
iGPM	Cross-Validation (CV)	11193	7.5719×10^{-4}
SPF	Least-Squares Estimator	4.4684	0.2976×10^{-4}
RBF	Ridge Regression + CV	18963	2.3216×10^{-4}
SVR	Quadratic Programming + CV	18897	2.5425×10^{-4}
RB-IDW	Least-Squares Estimator + CV	31.8472	3.0913×10^{-4}
EFA	N/A	—-	1.1797

6.4 Discussion

The nanoparticle dynamics model is a more realistic case study for the dynamic implementation of Gaussian process models. Although the case study explored in this chapter is not truly an "expensive" dynamic simulations (i.e. it only takes 55 seconds to describe a dynamic trajectory), the main result in this chapter shows that metamodeling approaches can be used to significantly decrease the computational

time of expensive dynamic simulations. Despite that in this thesis Gaussian process models have been used for dynamic predictions, it is necessary to recognized support vector regression as a viable and efficient approximate model for dynamic predictions.

This case study has provided additional challenges for the implementation of the GPM error estimation properties in expensive dynamic simulations. The kMC simulation generates a non-constant noise level in the stochastic observations used in the model. This non-constant noise level has shown significant effects in the performance of the proposed error estimation analysis for multivariate dynamic systems. The results in this chapter suggests that it is necessary to implement solutions within the framework of GPM, that aids in the identification of the noise structure. The stochastic kriging model proposed by Ankenman and cooworkers [6] contains an additional set of variables $\sigma^2_{u,ii}$ that represent the characteristic variance of a sample point in the model. Other suggestions include incorporating additional correlation functions in the regression covariance matrix V. With this idea, the non-constant noise level will be seen as an additional local correlation process that is simultaneously present in the stochastic observations.

One of the arguments in these comparisons of several mathematical models is that multiple dynamic problems must be evaluated in order to provide a more solid conclusion regarding these models. While this is a valid argument, the nanoparticle dynamic model is a fairly complex simulation in terms of the dimensional size, the issue with the state reduction and the non-constant noise level. Regardless of this, a deterministic dynamic model with the same dimension, and a even simpler dynamic model, with a highly nonlinear behavior could be a good addition to complete the metamodeling comparison.

In summary, based on their accuracy of mean prediction and the computational requirements of the mathematical models, support vector regression is a recommended choice for approximate models of expensive dynamic trajectories. Support vector

regression has similar LEM and DEM accuracy as the iGPM metamodel, but it generates dynamic predictions three times faster than the iGPM metamodel. In favor of the Gaussian process models, the implementation of the MLE as a parameter estimation will be significantly lower compared to the SVR metamodel. Moreoever, the statistical interpretation of the Gaussian process model allows the estimation of the prediction error, as it has been shown in this thesis. This type of results about the prediction error cannot be obtained with the support vector regression.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

This thesis shows the implementation of metamodeling approaches as a solution to approximate complex, and also expensive, dynamic simulations. The metamodel is implemented by constructing a iterative mapping function with an empirical model. For that purpose, the selected metamodel was a Gaussian process model. The iterative use of this mapping function approximates the dynamic evolution of a system. The major source of error in this type of metamodeling implementation is the model itself. By using recursively its own prediction, the model propagates the prediction errors at early times. Therefore, understanding the propagation of error in these iterative mapping functions is an essential element in the successful implementation of metamodeling for dynamic simulations.

The results in this thesis provide evidence that metamodeling is a viable option for approximating dynamic simulations. This thesis concludes that metamodeling approaches are suitable for approximations when the user has a good control over the data acquisition from the complex dynamic simulations. Ideally, the user should have enough control to obtain sample information at a specific state and a specific time during a dynamic prediction. This level of sampling control allows the user to incorporate dynamic information about unexplored areas in the dynamic region, or about regions where additional data points are required for precise predictions. This thesis recommends that metamodeling approaches can be used to approximate low-dimensional dynamic simulations (up to 8 state variables). Additionally, this thesis shows that metamodeling techniques can be combined with state reduction techniques

to approximate high-dimensional dynamic simulations, as it is the nanoparticle dynamic model. Last, this thesis uses metamodeling techniques for approximation of dynamic simulations over a continuous state space, further studies are required to generate dynamic predictions of a discrete state space.

Gaussian process models are one of the most popular empirical models used in engineering. Its convenient mathematical properties, its flexibility and its error estimation properties makes this model a recurrent candidate to approximate non-linear functions. This thesis analyzes the implications of the error estimation properties of the model when stochastic observations are used. The conclusion is that these properties depend on the identification of local correlations in the stochastic output information. This thesis defines a lower limit of the signal-to-noise ratio $\frac{\mu}{\sigma} = 10$, to guarantee those error estimation properties, despite the presence of noise in the observations. This thesis reveals two scenarios where the identification of local correlation is significantly reduced: when the noise in the observations is sufficiently large (i.e. when the signal-to-noise ratio is below the recommended noise limit) or when highly descriptive global trend functions are included in the model.

Traditionally the Gaussian process model uses a maximum likelihood estimator as a methodology to estimate its parameters. While the maximum likelihood estimator showed a good performance in the identification of the noise in the stochastic observations, a more robust approach to estimate this particular parameter is using repetitions and a sample variance calculation. This isolated estimation of the noise level effectively separates the data variability in the stochastic observations. At the same time, this thesis shows that experimental designs for GPM should favor a space-filling approach, even in the cases of stochastic observations. In this way, the identification of the local correlation in the model is enhanced and the error estimation properties of GPM are preserved.

One of the major questions in this thesis is: can Gaussian process models estimate

the propagated error in dynamics? This thesis evaluates two dynamic implementations where the input uncertainty was explicitly considered in the GPM predictive distribution. The result indicates that it is possible to track the propagated error using the Gaussian approximation for input uncertainty. This result requires additional evaluations, including the effects of the number of sample points and the evaluation with another one-dimensional dynamic model.

The implementation of Gaussian process models for the prediction of the non-adiabatic reactor in Chapter 5 provided a large amount of information to be considered in the entire dynamic framework. This thesis is the first case where a multivariate Gaussian process model (known also as cokriging in spatial statistics), is implemented as an iterative mapping function. This thesis expands the error estimation analysis to multivariate systems and shows that it is possible to provide information about the prediction error distribution. This chapter opens the discussion about sparsification and the appropriate methods to select sample points for system dynamics. This chapter shows the difficulties that occur with global trend functions, thanks to the extrapolation problem. The presence of the two steady states suggests the importance of including gradient and Hessian output information as part of the selection of sample points in the GPM dynamic framework. This case study shows that Gaussian process models are suitable for representing the dynamics of stable steady states.

At last, this thesis returned to the beginning of this work, by implementing all the knowledge that has been created around Gaussian process models, and comparing it with other metamodeling approaches. As a result, support vector regression is a recommended choice, over the Gaussian process model, if the only interest is is obtaining accurate mean dynamic predictions. To conclude with a final remark, the best GPM implementation for the task of approximating complex and expensive dynamic trajectories is the stochastic kriging model [6], using a constant regression

function. This particular version of Gaussian process models allows features to capture constant and non-constant noise levels, as well as the local correlation necessary to preserve the error estimation properties of the model.

7.2 Future Work

The most important element in the construction of a Gaussian process model dynamic framework is the generation of the database $\mathcal{D}$ for the model. In metamodeling, the selection of sample points for an empirical model is based on certain geometrical and/or spatial properties in the input space, or tailored to improve certain characteristics of the empirical models to be used. The research area of design and analysis computer experiments (DACE) explores the implementation of novel techniques to improve the construction of metamodels. Then the idea is to bring some of the concepts from the DACE research area and apply them for the dynamic implementation presented in this thesis.

Most of the current implementations in DACE are related to a sequential selection of the sample points in the input space, such that at every iteration, the metamodel gains information about the approximated function. Such concept could be implemented in this dynamic implementation of GPM to improve the dynamic prediction of the metamodel. Hernandez and Grover implemented a preliminary version of a sequential DACE for the identification of system dynamics using Gaussian process models [56]. Usually these sequential DACE approaches begins with an initial set of sample points in the input space. Then the selection of the next sample points is made via an optimization problem, where the new sample point for the dataset is selected from the predetermined input space.

In the case of this thesis the preliminary database for this sequential DACE can be generated following the exploration of the initial sample region and the sparsification procedure. The major challenges in the implementation of these ideas appear

at the moment of performing the selection of a new sample point. The dynamic region, which is the input space where the metamodel is built, does not have a clear mathematical boundary that describes it. Hernandez and Grover [56] solved this situation by constructing a convex hull that envelops all sample points in the dynamic region, and then used those hyperplanes as constraints for the optimization in a five-dimensional state space. The drawback of this solution is to assume that the dynamic region is indeed a convex region. If the dynamic region is non-convex, then is possible that the solution of this optimization problem is a sample point with no significant value to describe the dynamics of the system.

Another element that was not incorporated in this thesis is the model selection of the correlation functions in Gaussian process models. During the thesis, only one correlation function was used, the Gaussian correlation function. This thesis explained in Chapter 2 the mathematical reasons why this correlation function has been used in most metamodeling research. There are a few references where the problem of model selection for correlation functions is addressed [44, 61, 124]. The current emphasis in the implementation of correlation functions is to use *non-stationary* correlation functions. The Gaussian correlation function in Equation (9) is a *stationary* correlation function, because the correlation between two residuals does not depend of the location of those points in the input space, it only depends on the distance between them. In a non-stationary correlation function, the specific location of the sample points in the input space is used as part of the local correlation. In the context of the dynamic implementation of GPM, a non-stationary correlation function is capable of capturing nonlinear behaviors of the output information that occurs in specific locations of the dynamic region. The non-stationary correlation function allows a more descriptive local correlation, but it requires a larger number of sample points in the dataset $\mathcal{D}$ to estimate the larger number of parameters in the function. Although these types of correlations are not so common in engineering, there are few cases where these

functions have been used in GPM for computer experiments [155] and real climate data [110].

REFERENCES

[1] Ababou, R., Bagtzoglou, A. C., and Wood, E. F., "On the condition number of covariance matrices in kriging, estimation and simulation of random fields," *Mathematical Geology*, vol. 26, pp. 99–133, 1994.

[2] Abrol, S., Lu, M., Hill, D., Herrick, A., and Edgar, T. F., "Faster dynamic process simulation using in situ adaptive tabulation," *Industrial & Chemistry Engineering Research*, vol. 49, pp. 7814–7823, 2010.

[3] Abt, M. and Walsh, W., "Fisher information and maximum likelihood estimation of covariance parameters in Gaussian stochastic processes," *Canad. J. Statist.*, vol. 26, pp. 127–137, 1998.

[4] Aggelogiannaki, E. and Sarimveis, H., "Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models," *Computers and Chemical Engineering*, vol. 32, pp. 1225–1237, 2008.

[5] Alexander, F. J., Johnson, G., Eyink, G. L., and Kevrekidis, I. G., "Equation-free implementation of statistical moment closures," *Physical Review E*, vol. 77, p. 026701, 2008.

[6] Ankenman, B., Nelson, B. L., and Staum, J., "Stochastic kriging for simulation metamodeling," *Operations Research*, vol. 58, no. 2, pp. 371–382, 2010.

[7] Aschenbrenner, O., Dahmen, N., Schaber, K., and Dinjus, E., "Adsorption of dimethyl(1,5-cyclooctadiene)platinum on porous supports in supercritical carbon dioxide," *Ind. Eng. Chem. Res.*, vol. 47, no. 9, pp. 3150–3155, 2008.

[8] Aschenbrenner, O., Kemper, S., Dahmen, N., Schaber, K., and Dinjus, E., "Solubility of β-diketonates, cyclopentadienyls, and cyclooctadiene complexes with various metals in supercritical carbon dioxide," *Journal of Supercritical Fluids*, vol. 41, pp. 179–186, 2007.

[9] Aumi, S. and Mhaskar, P., "Integrating data-based modeling and nonlinear control tools for batch process control," *AICHE Journal*, 2011. In Press, DOI: 10.1002/aic.12720.

[10] Azman, K. and Kocijan, J., "Application of Gaussian processes for black-box modelling of biosystems," *ISA Transactions*, vol. 46, pp. 443–457, 2007.

[11] Baggett, L. S., *A comprenhensive approach to spatial and spatiotemporal dependence modeling*. PhD thesis, Rice University, 2000.

[12] Basak, D., Pal, S., and Patranabis, D. C., "Support vector regression," *Neural Information Processing Letters and Reviews*, vol. 11, pp. 203–224, 2007.

[13] Bayrakceken, A., Kitkamthorn, U., Aindow, M., and Erkey, C., "Decoration of multi-wall carbon nanotubes with platinum nanoparticles using supercritical deposition with thermodynamic control of metal loading," *Scripta Materialia*, vol. 56, pp. 101–103, 2007.

[14] Blasco, J. A., Fueyo, N., Dopazo, C., and Chen, J.-Y., "A self-organizing-map approach to chemistry representation in combustion applications," *Combustion Theory and Modelling*, vol. 4, pp. 61–76, 2000.

[15] Boser, B. E., Guyon, I. M., and Vapnik, V. N., "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152, ACM Press, 1992.

[16] Boyle, P. and Frean, M., "Dependent Gaussian process," in *Advances in Neural Information Processing Systems 17* (Saul, L. K., Weiss, Y., and Bottou, L., eds.), pp. 217–224, Cambridge, MA: The MIT Press, 2005.

[17] Chapman, W. L., Welch, W. J., Bowman, K. P., Sacks, J., and Walsh, J. E., "Arctic sea ice variability: Model sensitivities and a multidecadal simulation," *Journal of Geophysical Research*, vol. 99, pp. 919–935, 1994.

[18] Chen, G., Han, X., Liu, G., Jiang, C., and Zhao, Z., "An efficient multi-objective optimization method for black-box functions using sequential approximate technique," *Applied Soft Computing*, vol. 12, pp. 14–27, 2012.

[19] Chen, V. C. P., Tsui, K.-L., Barton, R. R., and Meckesheimer, M., "A review on design, modeling and applications of computer experiments," *IIE Transactions*, vol. 38, no. 4, pp. 273–291, 2006.

[20] Chrastil, J., "Solubility of solids and liquids in supercritical gases," *J. Phys. Chem*, vol. 86, pp. 3016–3021, 1982.

[21] Crary, S. B., "Design of computer experiments for metamodel generation," *Analog Integrated Circuits and Signal Processing*, vol. 32, pp. 7–16, 2002.

[22] Crespo, J. L., Zorrilla, M., Bernardos, P., and Mora, E., "A new image prediction model based on spatio-temporal techniques," *Visual Comput.*, vol. 23, pp. 419–431, 2007.

[23] Crespo, L. G. and Sun, J. Q., "Fixed final time optimal control via simple cell mapping," *Nonlinear Dynamics*, vol. 31, pp. 119–131, 2003.

[24] Cressie, N., *Statistics for Spatial Data*. Wiley Interscience, 3^{rd} ed., 1993.

[25] Crombecq, K., Laermans, E., and Dhaene, T., "Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling," *European Journal of Operational Research*, vol. 214, pp. 683–696, 2011.

[26] Currin, C., Mitchell, M., Morris, M., and Ylvisaker, D., "Bayesian prediction of deterministic function, with applications to the design and analysis of computer experiments," *Journal of the American Statistical Association*, vol. 86, pp. 953–963, 1991.

[27] Davis, G. J. and Morris, M. D., "Six factors which affect the condition number of matrices associated with kriging," *Mathematical Geology*, vol. 29, pp. 669–683, 1997.

[28] Deisenroth, M. P., Rasmussen, C. E., and Peters, J., "Gaussian process dynamic programming," *Neurocomputing*, vol. 72, pp. 1508–1524, 2009.

[29] den Hertog, D., Kleijnen, J. P. C., and Siem, A. Y. D., "The correct kriging variance estimated by bootstrapping," *Journal of the Operational Research Society*, vol. 57, pp. 400–409, 2006.

[30] Dixon, L. C. W. and Szeg, G. P., "The global optimization problem: An introduction," in *Towards Global Optimisation 2* (Dixon, L. C. W. and Szeg, G. P., eds.), pp. 1–15, Amsterdam: Elsevier Service, 1978.

[31] Drucker, H., Burges, C., Kaufman, L., Smola, A., and Vapnik, V., "Support vector regression machines," in *Proceedings of the 1996 Conference: Advances in Neural Information Processing Systems*, vol. 9, pp. 155–161, 1996.

[32] Elanayar, S. and Shin, Y. C., "Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems," *IEEE Transactions on Neural Networks*, vol. 5, pp. 594–603, 1994.

[33] Eldar, Y. C., "Minimum variance in biased estimation: Bounds and asymptotically optimal estimators," *IEEE Transactions on Signal Processing*, vol. 52, pp. 1915–1929, 2004.

[34] Erkey, C., "Preparation of metallic supported nanoparticles and films using supercritical fluid deposition," *Journal of Supercritical Fluids*, vol. 47, pp. 517–522, 2007.

[35] Finney, E. E. and Finke, R. G., "Nanocluster nucleation and growth kinetic and mechanistic studies: A review emphasizing transition-metal nanoclusters," *Journal of Colloid and Interface Science*, vol. 317, pp. 351–374, 2008.

[36] Fischer, J. and Kreuzer, E., "Generalized cell mapping for randomly perturbed dynamical systems," *Z. Angew. Math. Mech.*, vol. 81, no. 11, pp. 769–777, 2001.

[37] Gear, C. W., Kevrekidis, I. G., and Theodoropoulos, C., "Coarse integration/bifurcation analysis via microscopic simulators: micro-Galerkin methods," *Computers and Chemical Engineering*, vol. 26, pp. 941–963, 2002.

[38] Gelfand, A. E., Ghosh, S. K., Knight, J. R., and Sirmans, C. F., "Spatio-temporal modeling of residential sales data," *Journal of Business & Economic Statistics*, vol. 16, no. 3, pp. 312–321, 1998.

[39] Gillespie, D. T., "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *Journal of Computational Physics*, vol. 22, pp. 403–434, 1976.

[40] Girard, A. and Murray-Smith, R., "Gaussian processes: Prediction at a noisy input and application to iterative multiple-step ahead forecasting of time series," in *Switching and Learning in Feedback Systems (Lecture Notes in Computer Science 3355)* (Murray-Smith, R. and Girard, A., eds.), pp. 158–184, Berlin: Springer, 2005.

[41] Gneiting, T., Kleiber, W., and Schlather, M., "Matern cross-covariance functions for multivariate random fields," *Journal of the American Statistical Association*, vol. 105, no. 491, pp. 1167–1177, 2010.

[42] Goel, T., Hafkta, R. T., and Shyy, W., "Comparing error estimation measures for polynomial and kriging approximation of noise-free functions," *Struct. Multidisc. Optim.*, vol. 38, pp. 429–442, 2009.

[43] Goldberger, A. S., "Best linear unbiased prediction in the generalized linear regression model," *Journal of the American Statistical Association*, vol. 57, no. 298, pp. 369–375, 1962.

[44] Gorsich, D. J. and Genton, M. G., "Variogram model selection via nonparametric derivative estimation," *Mathematical Geology*, vol. 32, pp. 249–270, 2000.

[45] Goulard, M. and Voltz, M., "Linear coregionalization model: Tools for estimation and choice of cross-variogram matrix," *Mathematical Geology*, vol. 24, pp. 269–286, 1992.

[46] Grancharova, A., Kocijan, J., and Johansen, T. A., "Explicit stochastic predictive control of combustion plants based on Gaussian process models," *Automatica*, vol. 44, pp. 1621–1631, 2008.

[47] Gregorcic, G. and Lightbody, G., "Gaussian process approach for modeling of nonlinear systems," *Engineering Applications of Artificial Intelligence*, vol. 22, pp. 522–533, 2009.

[48] Guder, R. and Kreuzer, E., "Control of an adaptive refinement technique of generalized cell mapping by system dynamics," *Nonlinear Dynamics*, vol. 20, pp. 21–32, 1999.

[49] Gummalla, M., Tsapatsis, M., Watkins, J. J., and Vlachos, D. G., "Multiscale hybrid modeling of film deposition within porous substrates," *AIChE Journal*, vol. 50, no. 3, pp. 684–695, 2004.

[50] Gunn, S. R., "Support vector machines for classification and regression," tech. rep., Image Speech and Intelligent Systems Research Group, University of Southampton, 1997.

[51] Guo, L., Billings, S., and Coca, D., "Identification of partial differential equation models for a class of multiscale spatio-temporal dynamical systems," *International Journal of Control*, vol. 83, pp. 40–48, 2010.

[52] Habbi, H., Kidouche, M., and Zelmat, M., "Data-driven fuzzy models for nonlinear indentification of a complex heat exchanger," *Applied Mathematical Modeling*, vol. 35, pp. 1470–1482, 2011.

[53] Handcock, M. S. and Wallis, J. R., "An approach to statistical spatial-temporal modeling of meteorological fields," *Journal of the American Statistical Association*, vol. 89, no. 426, pp. 368–378, 1994.

[54] Hardy, R. L., "Multiquadric equations of topography and other irregular surfaces," *Journal of Geophysical Reseach*, vol. 76, pp. 1905–1915, 1971.

[55] Hedengren, J. D. and Edgar, T. F., "Approximate nonlinear model predictive control with in situ adaptive tabulation," *Computers and Chemical Engineering*, vol. 32, pp. 706–714, 2008.

[56] Hernandez, A. F. and Grover, M. A., "Stochastic dynamic predictions using Gaussian process models for nanoparticle synthesis," *Computers & Chemical Engineering*, vol. 34, no. 12, pp. 1953–1961, 2010.

[57] Hernandez, A. F. and Grover, M. A., "Comparison of sampling strategies for Gaussian process models, with application to nanoparticle dynamics," *Industrial & Chemistry Engineering Research*, vol. 50, pp. 1379–1388, 2011.

[58] Hocking, R., "The analysis and selection of variables in linear regression," *Biometrics*, vol. 32, pp. 1–49, 1976.

[59] Hsu, C. S., "A theory of cell-to-cell mapping dynamical systems," *Journal of Applied Mechanics - Transactions of the ASME*, vol. 47, no. 4, pp. 931–939, 1980.

[60] Huang, C., Yao, Y., Cressie, N., and Hsing, T., "Multivariate intrinsic random functions for cokriging," *Mathematical Geosciences*, vol. 41, pp. 887–904, 2009.

[61] Huang, H.-C., Martinez, F., Mateu, J., and Montes, F., "Model comparison and selection for stationary space-time models," *Computational Statistics & Data Analysis*, vol. 51, pp. 4577–4596, 2007.

[62] Jones, D., Schonlau, M., and Welch, W., "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[63] Jones, R. H. and Zhang, Y., "Models for continuous stationary space-time processes," in *Modelling Longitudinal and Spatially Correlated Data (Lecture Notes in Statistics 122)* (Gregoire, T. G., Brillinger, D. R., Diggle, P. J., Russek-Cohen, E., Warren, W. G., and Wolfinger, R. D., eds.), pp. 289–298, New York: Springer, 1997.

[64] Joseph, R., Hung, Y., and Sudjanto, A., "Blind kriging: A new method for developing metamodels," *Journal of Mechanical Design*, vol. 130, no. 3, 2008.

[65] Joseph, V. R. and Kang, L., "Regression-based inverse distance weighting with applications to computer experiments," *Technometrics*, vol. 53, pp. 254–265, 2011.

[66] Jost, G., Heuvelink, G. B. M., and Papritz, A., "Analysing the space-time distribution of soil water storage of a forest ecosystem using spatio-temporal kriging," *Geoderma*, vol. 128, pp. 258–273, 2005.

[67] Kevrekidis, I. G., Gear, C. W., and Hummer, G., "Equation-free: The computer-aided analysis of complex multiscale systems," *AIChE Journal*, vol. 50, no. 7, pp. 1346–1355, 2004.

[68] Kevrekidis, I. G. and Samaey, G., "Equation-free multiscale computation: algorithms and applications," *Annual Review of Physical Chemistry*, vol. 60, pp. 321–344, 2009.

[69] Kim, H.-C. and Ghahramani, Z., "The EM-EP algorithm for Gaussian process classification," in *Proceedings of the International Conference in Computer Vision*, 2003.

[70] Kirk, P. D. W. and Stumpt, M. P. H., "Gaussian process regression bootstrapping: exploring the effects of uncertainty in time course data," *Bioinformatics*, vol. 9, no. 10, pp. 1300–1306, 2009.

[71] Kitanidis, P. K., "Parametric estimation of covariances of regionalized variables," *Water Resources Bulletin*, vol. 23, no. 4, pp. 557–567, 1987.

[72] Kitanidis, P. K. and Shen, K.-F., "Geostatistical interpolation of chemical concentration," *Advances in Water Resources*, vol. 19, no. 6, pp. 369–378, 1996.

[73] Kitanidis, P. K., "Statistical estimation of polynomial generalized covariance functions and hydrologic applications," *Water Resources Research*, vol. 19, pp. 909–921, 1983.

[74] Kitanidis, P. K. and Lane, R. W., "Maximum likelihood parameter estimation of hydrologic spatial process by the Gauss-Newton method," *Journal of Hydrology*, vol. 79, pp. 53–71, 1985.

[75] Kleijnen, J. P. C., van Beers, W., and van Nieuwenhuyse, I., "Constrained optimization in expensive simulation: Novel approach," *European Journal of Operational Research*, vol. 202, pp. 164–174, 2010.

[76] Kleijnen, J. P. C. and van Beers, W. C. M., "Robustness of kriging when interpolating in random simulation with heterogeneous variances: Some experiments," *European Journal of Operational Research*, vol. 165, pp. 826–834, 2005.

[77] Kleijnen, J. P., "An overview of the design and analysis of simulation experiments for sensitivity analysis," *European Journal of Operational Research*, vol. 164, pp. 287–300, 2005.

[78] Kocijan, J. and Likar, B., "Gas-liquid separator modeling and simulation with Gaussian-process models," *Simulation Modeling Practice and Theory*, vol. 16, pp. 910–922, 2008.

[79] Koehler, J. R. and Owen, A. B., "Computer experiments," in *Handbook of Statistics* (Ghosh, S. and Rao, C. R., eds.), pp. 261–308, New York: Elsevier Science, 1996.

[80] Koutsourelakis, P.-S. and Bilionis, E., "Scalable Bayesian reduced-order models for simulating high-dimensional multiscale dynamical systems," *Multiscale Modeling and Simulation*, vol. 9, pp. 449–485, 2011.

[81] Krige, D. G., "A statistical approach to some basic mine valuation problems on the witwatersrand," *J. of the Chem., Metal. and Mining Soc. of South Africa*, vol. 52, pp. 119–139, 1951.

[82] Kunsch, H., Papritz, A., and Bassi, F., "Generalized cross-covariances and their estimation," *Mathematical Geology*, vol. 29, pp. 779–799, 1997.

[83] Kyriakidis, P. C. and Journel, A. G., "Geostatistical space-time models: A review," *Mathematical Geology*, vol. 31, no. 6, pp. 651–684, 1999.

[84] Li, L. and Reverz, P., "Interpolation methods for spatio-temporal geographic data," *Computers, Environment and Urban Systems*, vol. 28, pp. 201–227, 2004.

[85] Li, Y., Ng, S., Xie, M., and Goh, T., "A systematic comparison of metamodeling techniques for simulation optimization in decision support systems," *Applied Soft Computing*, vol. 10, pp. 1257–1273, 2010.

[86] Lin, Y., Cui, X., Yen, C., and Wai, C. M., "Platinum/carbon nanotube nanocomposite synthesized in supercritical fluid as electrocatalysts for low-temperature fuel cells," *J. Phys. Chem. B*, vol. 109, pp. 14410–14415, 2005.

[87] Lindenbergh, R., Keshin, M., der Marel, H. V., and Hanssen, R., "High resolution spatio-temporal water vapour mapping using gps and meris observations," *International Journal of Remote Sensing*, vol. 29, no. 8, pp. 2393–2409, 2008.

[88] Liu, B. J. D. and Pope, S. B., "The performance on *in situ* adaptive tabulation in computations of turbulent flames," *Combustion Theory and Modeling*, vol. 9, no. 4, pp. 549–568, 2005.

[89] Liu, X. and Zhang, Y., "Numerical dynamic modeling and data driven control via least square techniques and Hebbian learning algorithm," *International Journal of Numerical Analysis and Modeling*, vol. 7, pp. 66–86, 2010.

[90] Loeppky, J. L., Sacks, J., and Welch, W. J., "Choosing the sample size of a computer experiment: A practical guide," *Technometrics*, vol. 51, pp. 366–376, 2009.

[91] Lophaven, S. N., Nielsen, H. B., and Sondergaard, J., *DACE: A Matlab Kriging toolbox, v. 2.0*. IMM Technical University of Denmark, Lyngby, 2002.

[92] Luo, X., *Spatiotemporal Stochastic Models for Earth Science and Engineering Applications*. PhD thesis, McGill University, 1998.

[93] Ma, C., "Spatio-temporal variograms and covariance models," *Adv. Appl. Prob.*, vol. 37, pp. 706–725, 2005.

[94] Marchant, B. and Lark, R., "Optimized sample schemes for geostatistical surveys," *Mathematical Geology*, vol. 39, no. 1, pp. 113–134, 2007.

[95] Mardia, K. V. and Goodall, C. R., "Spatial-temporal analysis of multivariate environmental monitoring data," in *Multivariate Environmental Statistics* (Patil, G. and Rao, C., eds.), pp. 347–386, Amsterdam: Elsevier Science, 1993.

[96] Mardia, K. V. and Marshall, R. J., "Maximum likelihood estimation of models for residual covariance in spatial regression," *Biometrika*, vol. 71, pp. 135–146, 1984.

[97] Marrel, A., Iooss, B., Dorpe, F. V., and Volkova, E., "An efficient methodology for modeling complex computer codes with Gaussian processes," *Computational Statistics and Data Analysis*, vol. 52, pp. 4731–4744, 2008.

[98] Martin, J. D. and Simpson, T. W., "Use of kriging models to approximate deterministic computer models," *AIAA Journal*, vol. 12, no. 4, pp. 115–125, 2005.

[99] Martin, J. D., "Computational improvements to estimating kriging metamodel parameters," *Journal of Mechanical Design*, vol. 131, p. 084501, 2009.

[100] Matheron, G., *Traite de Geostatistique Appliquee, Tome II: Le Krigeage*. Editions Bureau de Recherche Geologiques et Minieres, Paris, 1963.

[101] McKay, M., Beckman, R., and Conover, W., "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, pp. 239–245, 1979.

[102] Meckesheimer, M., Booker, A. J., Barton, R. R., and Simpson, T. W., "Computationally inexpensive metamodel assessment strategies," *AIAA Journal*, vol. 40, no. 10, pp. 2053–2060, 2002.

[103] Morris, M. and Mitchell, T., "Exploratory designs for computational experiments," *Journal of Statistical Planning and Inference*, vol. 43, pp. 381–402, 1995.

[104] Musavi, M., Ahmed, W., Chan, K., Faris, K., and Hummels, D., "On the training of radial basis function classifiers," *Neural Networks*, vol. 5, pp. 595–603, 1992.

[105] O'Dowd, R. J., "Conditioning of coefficient matrices of ordinary kriging," *Mathematical Geology*, vol. 23, pp. 721–739, 1991.

[106] Oguz, C. and Gallivan, M. A., "A data-driven approach for reduction of molecular simulations," *International Journal of Robust and Nonlinear Control*, vol. 15, no. 15, pp. 727–743, 2005.

[107] Oguz, C. and Gallivan, M. A., "Identification of a dynamic model for a thin film deposition process using a self-organizing map," in *IEEE International Joint Conference on Neural Networks*, (Vancouver), pp. 973–980, 2006.

[108] Oguz, C. and Gallivan, M. A., "Optimization of a thin film deposition process using a dynamic model extracted from molecular simulations," *Automatica*, vol. 44, no. 8, pp. 1958–1969, 2008.

[109] Oliver, D. S., "Gaussian cosimulation: Modeling of the cross-covariance," *Mathematical Geology*, vol. 35, pp. 681–698, 2003.

[110] Paciorek, C. J. and Schervish, M. J., "Spatial modelling using a new class of nonstationary covariance functions," *Environmetrics*, vol. 17, pp. 483–506, 2006.

[111] Pardo-Iguzquiza, E. and Dowd, P. A., "The second-order stationary universal kriging model revisited," *Mathematical Geology*, vol. 30, pp. 347–378, 1998.

[112] Pardo-Iguzquiza, E., Mardia, K. V., and Chica-Olmo, M., "ML-MATERN: A computer program for maximum likelihood inference with the spatial matern covariance model," *Computers & Geosciences*, vol. 35, pp. 1139–1150, 2009.

[113] Patterson, H. and Thompson, R., "Recovery of inter-block information when block sizes are unequal," *Biometrika*, vol. 58, pp. 545–554, 1971.

[114] Pope, S. B., "Computationally efficient implementation of combustion chemistry using in-situ adaptive tabulation," *Combustion Theory and Modeling*, vol. 1, no. 1, pp. 41–63, 1997.

[115] Porcu, E., Mateu, J., and Saura, F., "New class of covariance and spectral density functions for spatio-temporal modeling," *Stoch. Environ. Res. Risk Assess.*, vol. 22, pp. S65–S79, 2008. Suppl 1.

[116] Qi, C., Li, H.-X., Zhang, X., Zhao, X., Li, S., and Gao, F., "Time/space-separation-based svm modeling for nonlinear distributed parameter processes," *Industrial & Chemistry Engineering Research*, vol. 50, pp. 332–341, 2011.

[117] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Tucker, P. K., "Surrogate-based analysis and optimization," *Progress in Aerospace Sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[118] Quiñonero-Candela, J. and Rasmussen, C. E., "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[119] Ramkrishna, D., *Population Balances: Theory and Applications to Particulate Systems in Engineering*. Academic Press, 2000.

[120] Rasmussen, C. E. and Williams, C. K. I., *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[121] Ratsch, C. and Venables, J. A., "Nucleation theory and the early stages of thin film growth," *Journal of Vacuum Science & Technology. A.*, vol. 21, no. 5, pp. S96–S109, 2003.

[122] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., "Design and analysis of computer experiments," *Statistical Science*, vol. 4, no. 4, pp. 409–435, 1989.

[123] Santner, T. J., Williams, B. J., and Notz, W., *The Design and Analysis of Computer Experiments*. Springer, 1^{st} ed., 2003.

[124] Seeger, M., "Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers," in *Advances in Neural Information Processing Systems 12* (Solla, S. A., Leen, T. K., and Muller, K.-R., eds.), pp. 603–609, Cambridge, MA: The MIT Press, 2000.

[125] Shallow, W. H. and Monahan, J. F., "Monte carlo comparison of ANOVA, MIVQUE, REML, and ML estimators of variance components," *Technometrics*, vol. 26, pp. 47–57, 1984.

[126] Shan, S. and Wang, G. G., "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions," *Struct. Multidisc. Optim.*, vol. 41, pp. 219–241, 2010.

[127] Shao, T. and Krishnamurty, S., "A clustering-based surrogate model updating approach to simulation-based engineering design," *Journal of Mechanical Design*, vol. 130, p. 041101, 2008.

[128] Shen, Y., Ng, A. Y., and Seeger, M., "Fast Gaussian process regression using kd-trees," in *Advances in Neural Information Processing Systems 18* (Weiss, Y., Schlkopf, B., and Platt, J., eds.), Cambridge, MA: The MIT Press, 2006.

[129] Shepard, D., "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 ACM National Conference*, (New York), pp. 517–524, 1968.

[130] Shin, M., Sargent, R. G., and Goel, A. L., "Gaussian radial basis functions for simulation metamodeling," in *Proceedings of the 2002 Winter Simulation Conference*, pp. 483–488, 2002.

[131] Simpson, T. W., Poplinski, J., Koch, P. N., and Allen, J. K., "Metamodels for computer-based engineering design: Survey and recommendations," *Engineering with Computers*, vol. 17, pp. 129–150, 2001.

[132] Smola, A. J. and Scholkopf, B., "A tutorial on support vector regression," Tech. Rep. NC-TR-98-030, Royal Holloway College, University of London, UK, 1998.

[133] Span, R. and Wagner, W., "A new equation of state for carbon dioxide covering the fluid region from the triple-point temperature to 1100 K at pressures up to 800 MPa," *J. Phys. Chem. Ref. Data*, vol. 25, no. 6, pp. 1509–1596, 1996.

[134] Stein, M. L., "Space-time covariance functions," *Journal of the American Statistical Association;*, vol. 100, no. 469, pp. 310–321, 2005.

[135] Stulajter, F., "Predictions in time series using multivariable regression models," *Journal of Time Series Analysis*, vol. 22, no. 3, pp. 365–373, 2001.

[136] Takasaki, S., Kawamura, Y., and Konagaya, A., "Selecting effective sirna sequences by using radial basis function network and decision tree learning," *BMC Bioinformatics*, vol. 7, p. S22, 2006.

[137] Theodoropoulos, C., Qian, Y.-H., and Kevrekidis, I. G., "Coarse stability and bifurcation analysis using time-steppers: A reaction-diffusion example," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 97, pp. 9840–9843, 2000.

[138] Todini, E. and Ferraresi, M., "Influence of parameter estimation uncertainty in kriging," *Journal of Hydrology*, vol. 175, pp. 555–566, 1996.

[139] Tonse, S. R., Moriarty, N. W., Frenklach, M., and Brown, N. J., "Computational economy improvements in prism," *International Journal of Chemical Kinetics*, vol. 35, no. 9, pp. 438–452, 2003.

[140] Ttterman, S. and Toivonen, H. T., "Support vector method for identification of Wiener models," *Journal of Process Control*, vol. 19, pp. 1174–1181, 2009.

[141] Tung, H. and Wong, M., "Financial risk forecasting with nonlinear dynamics and support vector regression," *Journal of the Operational Research Society*, vol. 60, pp. 685–695, 2009.

[142] van Beers, W. C. M. and Kleijnen, J. P. C., "Kriging for interpolation in random simulation," *Journal of the Operational Research Society*, vol. 54, pp. 255–262, 2003.

[143] van Beers, W. C. M. and Kleijnen, J. P. C., "Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping," *European Journal of Operational Research*, vol. 186, pp. 1099–1113, 2008.

[144] Varma, A. and Morbidelli, M., *Mathematical Methods in Chemical Engineering*. Oxford University Press, USA, 1997.

[145] Varshney, A. and Armaou, A., "Multiscale optimization using hybrid pde/kmc process systems with application to thin film growth," *Chemical Engineering Science*, vol. 60, pp. 6780–6794, 2005.

[146] Veljkovic, I., Plassmann, P. E., and Haworth, D. C., "A scientific online database for efficient function approximation," in *Computational Science and its Applications (Lecture Notes in Computer Science 2667)* (Kumar, V., Gavrilova, M. L., Tan, C. J. K., and L'Ecuyer, P., eds.), pp. 643–653, New York: Springer, 2003.

[147] Venables, J. A., "Rate equation approaches to thin-film nucleation kinetics," *Philosophical Magazine*, vol. 27, no. 3, pp. 697–738, 1973.

[148] Venkateswarlu, C. and Rao, K. V., "Dynamic recurrent radial basis function network model predictive control of unstable nonlinear processes," *Chemical Engineering Science*, vol. 60, pp. 6718–6732, 2005.

[149] ver Hoef, J. M. and Barry, R. P., "Constructing and fitting models for cokriging and multivariable spatial prediction," *Journal of Statistical Planning and Inference*, vol. 69, pp. 275–294, 1998.

[150] ver Hoef, J. M. and Cressie, N., "Multivariable spatial prediction," *Mathematical Geology*, vol. 25, pp. 219–240, 1993.

[151] Viana, F. A. C. and Haftka, R. T., "Cross validation can estimate how well prediction variance correlates with error," *AIAA Journal*, vol. 47, no. 9, pp. 2266–2270, 2009.

[152] Vivarelli, F. and Williams, C. K. I., "Discovering hidden features with Gaussian processes regression," in *Advances in Neural Information Processing Systems 11* (Kearns, M. S., Solla, S. A., and Cohn, D. A., eds.), pp. 613–619, Cambridge, MA: The MIT Press, 1999.

[153] Wackernagel, H., *Multivariate Geostatistics*. Springer, 3 ed., 2003.

[154] Wang, J. M., Fleet, D. J., and Hertzmann, A., "Gaussian process dynamical models for human motion," *IEEE Transactions of pattern analysis and machine intelligence*, vol. 30, pp. 283–298, 2008.

[155] Xiong, Y., Chen, W., Apley, D., and Ding, X., "A non-stationary covariance-based kriging method for metamodelling in engineering design," *International Journal for Numerical Methods in Engineering*, vol. 71, pp. 733–756, 2007.

[156] Yao, T., "Nonparametric cross-covariance modeling as exemplified by soil heavy metal concentrations from the swiss jura," *Geoderma*, vol. 88, pp. 13–38, 1999.

[157] Ye, X.-R., Lin, Y., Wang, C., Engelhard, M. H., Wang, Y., and Wai, C. M., "Supercritical fluid synthesis and characterization of catalytic metal nanoparticles on carbon nanotubes," *Journal of Materials Chemistry*, vol. 14, pp. 908–913, 2004.

[158] Yin, J., Ng, S. H., and Ng, K. M., "A study on the effects of parameter estimation on kriging model's prediction error in stochastic simulations," in *Proceedings of the Winter Simulation Conference*, pp. 674–685, 2009. doi: 10.1109/WSC.2009.5429703.

[159] Ying, Z., "Maximum likelihood estimation of parameters under a spatial sampling scheme," *Ann. Statist.*, vol. 21, pp. 1567–1590, 1993.

[160] Yoda, S., Mizuno, Y., Furuya, T., Takebayashi, Y., Otake, K., Tsuji, T., and Hiaki, T., "Solubility measurements of noble metal acetylacetonates in supercritical carbon dioxide by high performance liquid chromatography (HPLC)," *Journal of Supercritical Fluids*, vol. 44, pp. 139–147, 2008.

[161] Zavala, V. M., Constantinescu, E. M., Krause, T., and Anitescu, M., "On-line economic optimization of energy systems using weather forecast information," *Journal of Process Control*, vol. 19, pp. 1725–1736, 2009.

[162] Zhang, X., Song, K. Z., Lu, M. W., and Liu, X., "Meshless methods based on collocation with radial basis functions," *Computational Mechanics*, vol. 26, pp. 333–343, 2000.

[163] Zhang, Y., Kang, D., Saquing, C., Aindow, M., and Erkey, C., "Supported platinum nanoparticles by supercritical deposition," *Ind. Eng. Chem. Res.*, vol. 44, no. 11, pp. 4161–4164, 2005.

[164] Zhu, Z. and Stein, M. L., "Spatial sampling design for parameter estimation of the covariance function," *Journal of Statistical Planning and Inference*, vol. 134, pp. 583–603, 2005.

[165] Zimmerman, D. L. and Zimmerman, M. B., "A comparison of spatial semivariogram estimators and corresponding ordinary kriging predictors," *Technometrics*, vol. 33, pp. 77–91, 1991.

[166] Zufiria, P. J. and Martinez-Marin, T., "Improved optimal control methods based upon the adjoining cell mapping technique," *Journal of Optimization Theory and Applications*, vol. 118, no. 3, pp. 657–680, 2003.