NORTHWESTERN UNIVERSITY

High-dimensional Gaussian Process Methods for Uncertainty Quantification

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

in the field of

Industrial Engineering and Management Sciences

By

Moses Y-H. Chan

EVANSTON, ILLINOIS

December 2023

© Copyright by Moses Y-H. Chan 2023

All Rights Reserved

ABSTRACT

High-dimensional Gaussian Process Methods for Uncertainty Quantification

Moses Y-H. Chan

This thesis concerns the construction and analysis of emulators for high-dimensional computer simulation outputs. The construction of emulators is beneficial when computer simulations are time-consuming. Some common features in the case of high-dimensional computer simulations pose a significant challenge in constructing emulators, including simulation failures and non-homogeneous stochastic error structures.

The first contribution handles the scenario when simulation outputs contain missing responses due to simulation failures. We have developed an emulator method, Principal Component Gaussian Process with Missingness (PCGPwM), that seamlessly incorporates missing responses and scales favorably with the output dimension. The PCGPwM method is applied to facilitate Bayesian calibration of the nuclear Fayans density functional that fails occasionally.

The next contribution tackles multivariate outputs from stochastic simulation models that exhibit non-homogeneous error structure across output dimensions. We have developed

the method of Latent Component Gaussian Process (LCGP) to model the possibly heterogeneous error structure in simulation outputs. The LCGP method represents the simulation outputs with a basis representation formulation, while incorporating the output error structure, resulting in a flexible emulator that is able to capture the dynamics of the simulation model and its error structure. LCGP is applied to emulate a viscous anisotropic hydrodynamics model, a nuclear physics simulation model for understanding high-energy heavy-ion collisions.

The third contribution revisits the nuclear Fayans density functional and considers its failure patterns. We have developed Logistic Principal Component Analysis Gaussian Process (LPCAGP) to emulate failures, or broadly, multivariate binary outputs. By jointly estimating the logistic principal components for reducing the output dimensions and the hyperparameters for a multivariate GP, LPCAGP is able to capture the underlying failure patterns. LPCAGP excels in cases where the underlying failure probabilities are smooth with respect to the input space. The LPCAGP method is used to predict failures encountered by the Fayans density functional, providing a guide to future Bayesian calibration of the functional.

Acknowledgements

I wish to express my deepest appreciation for my advisor, Matt Plumlee. Not only does he provide research guidance, he is also a consistent encouragement to pursuing excellence in research and beyond. I am indebted to Barry Nelson who has served as a mentor since day zero when I visited the Northwestern IEMS department. The clarity of communication, passion in teaching, and *timeliness*, are several things I hope not to disappoint him with. I am grateful to Stefan Wild for his guidance and support in and outside of research. Conversations with him are always on point, and his effort in raising up younger researchers is remarkable. I also thank the Northwestern faculty who sharpened my thinking and shone the light ahead.

I had the fortune of receiving incredible instructions and supervisions at University of Michigan that shaped who I am as a researcher. I would like to thank Richard Hughes, who opened the door to research; Amy Cohn, who kept the door open and showed me many possible paths; Jim Bagian, who taught me effective and persuasive communications; and Joe DeRosier, who ensured any recommended solution is actionable.

This path would have been more difficult without knowing fellow researchers and students in IOE (Michigan), IEMS (Northwestern), and BAND (Bayesian Analysis of Nuclear Dynamics collaboration). Of the innumerable I would mention Mariana Escallon Barrios, Pablo Giuliani, Kyle Godbey, Dan Liyanage, Zhonghao Liu, Daniel Odell, Alexandra

Semposki, Sinan Seymen, Mookyong Son, Zhe (Robert) Su, Özge Sürer, Adam VanDeusen, Eugene Wickett, Tim Williams, and John Yannotty.

This series of work would not have been possible without fellow researchers who kindly provide the simulation data and assist with understanding the data sets correctly. I wish to thank Jared O'Neal and Paul-Gerhard Reinhard for developing the Fayans energy density functional (Chapters 2 and 4.) I wish to also thank Ulrich Heinz and Dan Liyanage for providing the data set from the viscous anisotropic hydrodynamics simulation model (Chapter 3.)

I wish to also thank Wessel Bruinsma for his helpful and extensive discussion over research topics and the use of software he has developed.

I wish to thank the Ann Arbor, Evanston, and Chicago communities who have surrounded me throughout my studies in the United States. Specifically, I thank the community orchestra members at Lakeview Orchestra have granted me a chance to make both music and friends amid research work.

My family has been my support throughout most if not all decisions I have made. Thank you for your patience and love.

Finally, to the feline gremlins in the house, Pine and Linden, thank you.

Dedication

To my family and to those who receive me as family.

Table of Contents

ABSTRACT 3

Acknowledgements 5

Dedication 7

List of Tables 10

List of Figures 12

Chapter 1. INTRODUCTION 17

1.1. Gaussian process background 18

1.2. High-dimensional simulation outputs 19

Chapter 2. CONSTRUCTING A SIMULATION SURROGATE WITH PARTIALLY OBSERVED OUTPUTS 22

2.1. Fayans energy density functional 27

2.2. High-dimensional surrogates 30

2.3. Fast surrogates with missing data 35

2.4. Numerical experiments 43

2.5. Calibrating Fayans energy density functional with missingness 51

2.6. Concluding remarks 55

Chapter 3. MODELING HIGH-DIMENSIONAL STOCHASTIC SIMULATION OUTPUTS WITH HETEROGENEOUS ERROR 57

3.1. Settings and preliminaries 61

3.2. Gaussian processes background 62

3.3. Latent component Gaussian process 73

3.4. Illustrative 3D function with 1D input 84

3.5. Implementation and numerical experiments 86

3.6. Viscous anisotropic hydrodynamic simulation model 95

3.7. Discussion 97

Chapter 4. EMULATING HIGH-DIMENSIONAL SIMULATION FAILURES 99

4.1. Preliminary background 104

4.2. Logistic principal component analysis Gaussian process 109

4.3. Illustrative sine failure example 118

4.4. Implementation and numerical comparisons 119

4.5. Emulating failures from the Fayans energy density functional 128

4.6. Concluding remarks 131

Chapter 5. CONCLUSION AND FUTURE DIRECTIONS 134

References 137

Appendix A. Proofs and computational details for PCGPwM 152

Appendix B. Additional figures from PCGPwM numerical experiments 169

Appendix C. Proofs and computational details for LCGP 182

List of Tables

2.1 Construction times (in seconds) and predictive accuracies of PCGPwM and colGP at output dimension $m = 200$. 51

2.2 Posterior 90% widths relative to their respective ranges for the 13-dimensional parameter using different emulation techniques. The narrowest 90% interval for each parameter is shown in boldface. 55

3.1 Predictive mean and variance of latent components 83

3.2 Mean training time (seconds) across emulator methods. 93

4.1 Average misclassification error (standard deviation) and average AUC (standard deviation) for the sine failure illustration are presented. LPCAGP uniformly achieves lower misclassification error and higher AUC than LPCA. 119

4.2 Misclassification error (standard deviation) comparisons for the circular failure example. The LPCAGP variants and LPCA achieve comparable error while LR achieves much higher error in both training and testing. 126

4.3 AUC comparisons for the circular failure example. LPCAGP-MM and LPCA achieve near-perfect and similar performance in terms of AUC, consistently better than LR across n. LPCAGP-LBFGS

seems to achieve slightly lower AUC when n is smaller, and achieves comparable AUC with LPCA and LPCAGP-MM at $n = 500$. The standard deviation over replications of any AUC measure is < 0.001. 126

List of Figures

2.1 Illustration of partially observed output in the Fayans EDF case study (Section 2.5). Each horizontal line (1–500) is a length 198 simulation output. A dark patch indicates where a response is missing. The horizontal lines are sorted by number of missing values in the output. 28

2.2 Illustrations of the effect of hyperparameter α in the variance term v_{ik} in Equation (2.8). Except for $\alpha = 0$, v_{ik} approaches infinity as w_{ik} approaches 1. 43

2.3 Comparison (log-log scale) of prediction accuracy of surrogate methods for (MNAR, 5%) scenario as the number of available data points increases. 49

2.4 Comparison of 90% coverage of surrogate methods for (MNAR, 5%) scenario as the number of available data points increases. 50

3.1 Comparison of basis representation GP methods. Each row presents one output location. The predictive mean and 95% predictive interval (left), and estimation errors of predictive means (right) are plotted for each method. The true functions are plotted in black with its 95% predictive intervals shaded in gray. LCGP and OILMM recover the true function and the heterogeneous errors while GPPCA does not. 86

3.2 Comparison of latent component uncertainty treatments. Each row presents one output location. The predictive mean and 95% predictive interval (left), and estimation errors of predictive means (right) are plotted for each approach. Both the full posterior and ELBO approach recover the function mean, while the profile likelihood approach does not. 87

3.3 Comparison of latent component behaviors. Each row presents one latent component. Predictive means (left) and variances (right) are plotted. The predictive means from the full posterior approach each captures some functional behavior, where the ELBO and profile likelihood approaches exhibit tapering of functional dynamics. The predictive variances from the full posterior approach are generally larger than those from the ELBO and the profile likelihood. 88

3.4 Performance comparisons for the five emulator methods using the Forrester 3D illustrative function. Log RMSE (left) and D-S score (center) are plotted against log training time. Median statistics are plotted with bold markers. The 95% empirical coverage (right) are presented in the boxplot and the nominal 95% level is indicated with the dashed line. 92

3.5 Accuracy comparisons for test functions with LCGP, OILMM, and SVGP. Log NRMSE are plotted against log training time for borehole (left), OTL circuit (center), and wing weight (right) functions. Median statistics are plotted with bold markers. 94

3.6 Coverage comparisons for test functions with LCGP, OILMM, and SVGP. Boxplots of empirical coverages are plotted for borehole (left), OTL circuit (center), and wing weight (right) functions. The coverage boxplot for SVGP for OTL circuit is concentrated around 1.0 with two runs at 0 and 0.5 respectively. 95

3.7 Summary performance measures for the VAH example with LCGP and other emulators. Log NRMSE (left) and log D-S score (center) are plotted against log training time. Median statistics are plotted with bold markers. The 95% empirical coverage (right) are presented in the boxplot and the nominal 95% level is indicated with the dashed line. 97

4.1 Comparison between separate and joint optimization of parameters. Top panel shows the ground truth probability, the predictive probability with LPCA, and the predictive probability with LPCAGP (left to right.) The bright yellow region corresponds to a high probability for $\pi_j(x)$, whereas the dark navy region corresponds to a low probability. Bottom panel shows the estimated GP components $\boldsymbol{w}_k, k = 1, 2, 3$, with LPCA and LPCAGP. The estimated latent components used in LPCA are non-smooth, as they directly result from the sparse logistic PCA method, whereas LPCAGP recovers smoother latent components with the joint optimization. 120

4.2 Misclassification error comparisons for the sine failure illustrative example. The boxplots of training (left) and testing (right) errors

are plotted against number of training points, n. Output dimension $p = 10$. The LPCAGP variants achieve lowest misclassification errors across n, where LR produces the highest errors. 123

4.3 AUC comparisons for the sine failure illustrative example. The boxplots of training (left) and testing (right) errors are plotted against number of training points, n. Output dimension $p = 10$. Similarly, the LPCAGP variants achieve the highest AUC across n, where LR performs poorly with the lowest AUCs. 124

4.4 Circular failure observations (top) and LPCAGP-MM predictive probabilities (bottom) at four sample inputs. The bright yellow region corresponds to a high probability for $\pi_j(\boldsymbol{x})$, whereas the dark navy region corresponds to a low probability. 125

4.5 Construction time comparisons (in seconds) of LPCAGP-MM, LPCAGP-LBFGS, LPCA, and LR emulators for the sine failure example. The LPCAGP variants require the longest among all methods in emulator constructions. The use of LPCAGP-LBFGS seems to scale better than LPCAGP-MM as n increases, but there is not a clear distinction in this numerical test. 128

4.6 Performance comparisons for the Fayans EDF failures. Training and testing misclassification errors (left) and Training and testing AUCs (right) are presented over 10 replications. LPCAGP variants achieve best performance among the emulator methods in both misclassification errors and AUCs. 131

4.7 Prediction visualization with LPCAGP-MM for the Fayans EDF failures. Testing set binary outputs (left), prediction error (middle) and predictions (right) are plotted, where the inputs are ordered by the number of failures encountered, increasing from left to right for visualization purpose. In the middle panel, a bright spot indicates that the emulator predicted failure where there is not, whereas a dark spot indicates the opposite. 132

CHAPTER 1

INTRODUCTION

Computer models, or simulation models, have become a standard tool of understanding and analyzing the behavior of physical systems. A computer model generally comprises a collection of code, which takes a user input, evaluates the simulation with respect to the input, and produces an output. Computer models are used in a multitude of scientific disciplines, for example, in nuclear physics [45, 76, 42, 125], climate science [29, 127], epidemiology [145, 157], and engineering [80, 143, 78]. Although the use of computer models is highly valuable in scientific discovery, the increasing computational burden associated with their growth in sophistication has presented significant challenges. Many computer models become expensive to use, in terms of computational time and resources. In such cases, emulators can be designed to capture the relationship between simulation inputs and outputs with a small set of simulation runs.

This thesis concerns the construction and analysis of emulators for addressing challenges due to common features of high-dimensional simulation outputs. In addition to the general challenge of analyzing high-dimensional output, two features are highlighted: (i) simulation models can fail, producing missing data; and (ii) simulation outputs can be noisy, and the error structure of the noise is unknown. This thesis presents three emulator strategies to analyze simulation models that exhibit these features.

1.1. Gaussian process background

Gaussian process (GP) emulators, or surrogates, are a popular alternative to directly using computationally expensive simulation models [135, 130, 137, 57]. GPs are popular for two major reasons. Assuming deterministic simulation outputs, a GP has an interpolation property. This means that predictions at previously seen inputs will reproduce the simulation outputs. GPs also inherit a natural notion of uncertainty quantification. This means that for any prediction, there is also a measure of how accurate the prediction is, in the form of a predictive distribution.

Consider a real-valued vector of simulation outputs as $\boldsymbol{y} = (y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_n))^\mathsf{T}$, collected at a set of inputs $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. To model the simulation function $y(\cdot)$ as a GP means that the output vector follows a multivariate Gaussian distribution, i.e., $\boldsymbol{y} \sim \mathsf{N}(\boldsymbol{\mu}, \boldsymbol{C})$, where $\boldsymbol{\mu} = (\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_n))^\mathsf{T}$ is a mean vector governed by a mean function μ, and $\boldsymbol{C}$ is a covariance matrix with elements $C_{ij} = c(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathrm{cov}(y(\boldsymbol{x}_i), y(\boldsymbol{x}_j))$, with $c(\cdot, \cdot)$ being a covariance function that characterizes the covariance between any two outputs. A prediction with this GP at an unseen input $\boldsymbol{x}_0$ follows a normal distribution with mean and variance

$$\widehat{\mu}(\boldsymbol{x}_0) = \mu(\boldsymbol{x}_0) + \boldsymbol{c}(\boldsymbol{x}_0)^\mathsf{T}\boldsymbol{C}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}), \quad \widehat{\sigma}^2(\boldsymbol{x}_0) = c(\boldsymbol{x}_0, \boldsymbol{x}_0) - \boldsymbol{c}(\boldsymbol{x}_0)^\mathsf{T}\boldsymbol{C}^{-1}\boldsymbol{c}(\boldsymbol{x}_0),$$

where $\boldsymbol{c}(\boldsymbol{x}_0) = (c(\boldsymbol{x}_1, \boldsymbol{x}_0), \ldots, c(\boldsymbol{x}_n, \boldsymbol{x}_0))^\mathsf{T}$.

When the simulation output is considered to be noisy, a positive nugget term is typically added to the diagonal of $\boldsymbol{C}$ [57]. Adding a nugget term is also common to improve the numerical stability of using a GP [62, 6, 124]. Note that the resulting GP no longer has the interpolation property.

1.2. High-dimensional simulation outputs

When the simulation output is high-dimensional, consisting of many responses, construction of GPs is less straightforward. To account for the relationship between all output responses across inputs produces a significant computational challenge [48]. Dimension-reduction techniques are often employed to construct an emulator in this situation. In this thesis, we generally employ a basis representation formulation to alleviate the computational burden, see e.g., Bayarri et al. [14], Higdon et al. [74, 76], Gu and Shen [65], Bruinsma et al. [25]. In Chapter 3, we discuss other possible GP treatments of high-dimensional outputs in detail.

As simulation models grow in complexity, failures may be encountered in some simulation runs, hence producing missing data. Dimension-reduction techniques generally require completeness of simulation outputs [14, 74, 76, 136]. However, if missing data are present in the simulation outputs, these methods break down. Moreover, when the simulation runs are time-consuming, there is a need for predicting if the run might fail, in order to reduce wasted time in computation.

In addition to simulation failures, many simulation models produce stochastic multivariate outputs. Current Gaussian process methods generally assume deterministic outputs or outputs with a homogeneous error structure, see e.g., Higdon et al. [76], Gu and Shen [65], Bruinsma et al. [25]. Modeling multivariate outputs with a general heterogeneous error structure poses significant modeling and computational challenges. In cases where this error heterogeneity is addressed, some form of approximations has been required [5, 77, 32].

Chapter 2 proposes a novel emulator method, PCGPwM (Principal Component Gaussian Process with Missingness), that permits the use of partially observed output while remaining computationally efficient. The PCGPwM method involves the imputation of missing values and the adjustment of the covariance matrix used for Gaussian process inference. The resulting emulator represents the available responses, disregards the missing responses, and provides meaningful uncertainty quantification. The proposed approach is shown to offer sharper inference than alternatives in a set of numerical tests and a case study calibrating the Fayans energy density functional model, which frequently returns incomplete output. (This work is published in Chan et al. [28].)

Chapter 3 proposes a new emulator method, LCGP (Latent Component Gaussian process), to handle stochastic multivariate outputs with heterogeneous errors. The proposed method is designed to learn heterogeneous error structures from the noisy outputs while remaining computationally efficient. Unlike previous emulators designed for heterogeneous errors, LCGP does not use an approximation method and offers exact inferences. This emulator approach shows promising numerical results when compared with state-of-the-art approaches when evaluated with a set of test functions and a viscous anisotropic hydrodynamics simulation model.

Chapter 4 proposes the method of LPCAGP (Logistic Principal Component Analysis Gaussian Process), an emulator for simulation failures, or multivariate binary outputs broadly. The proposed method utilizes a set of logistic principal components and a multivariate GP for modeling underlying failure probabilities. A majorization-minimization algorithm is developed to jointly estimate the principal components and GP hyperparameters. The proposed LPCAGP is evaluated alongside other current methods for binary

classification in two synthetic failure examples. Chapter 4 revisits the Fayans density functional in Chapter 2 and emulates the failures from the functional directly.

CHAPTER 2

CONSTRUCTING A SIMULATION SURROGATE WITH PARTIALLY OBSERVED OUTPUTS

Computer simulations are used to understand and analyze systems where directly experimenting on the real system is difficult or infeasible. The output of a simulation model depends on a user-specified input configuration that defines the physical and controllable properties of the system. When a user simulates the system, also referred to as running the simulation, they receive outputs related to quantities of interest to the user. Running a simulation can be computationally expensive; each run can cost thousands of core-hours, see for examples the simulation of storm surge [127], influenza spread [157], and nuclear dynamics [125]. Because these simulations are expensive, it is often helpful to build an emulator, or "surrogate," trained on simulation data to predict at unsimulated (i.e., out-of-sample) configurations.

Surrogate technology is often deployed for calibration [87], where an input configuration is represented by a multidimensional parameter. When run at a parameter, the simulation returns a high-dimensional output consisting of multiple responses collected on a set of fixed locations. While there are many variations of the exact type of inference [153, 59, 126], the overall goal is to learn the parameters by aligning the physical observations with the simulation outputs using computational tools like Markov Chain Monte Carlo (MCMC). Because simulation runs are expensive, it is not possible to run the simulation the millions

of times needed for MCMC. Instead, the idea is to run the simulation model for a set of representative parameters and to collect the simulation output from each run; consequently, the surrogate is constructed as the prediction conditional on the simulation output data. Important to solving the calibration problem is that the surrogate produces a measure of uncertainty in the surrogate's predictions.

The most prominent tool for building statistical surrogates involves Gaussian processes (GPs) [137, 57]. GPs offer both an accurate prediction and a measure of uncertainty. The case of high-dimensional outputs leads to the computational intractability of many surrogate construction tools. Two approaches have been proposed to remedy this computational challenge. The first one employs a Kronecker structure of the covariance function of the GP, which assumes a separation between parameters and locations [133, 80, 67, 111]. The second approach employs a dimension-reduction step for the outputs before the building of surrogates [14, 74, 66]. This is in contrast to data-reduction procedures that seek to choose a smaller set of points to represent the entire parameter space (e.g., the selection of support points [109]). Examples of dimension-reduction procedures include extractions of principal components [74, 97, 66], wavelet coefficients [14], and calibration-optimal bases [136] from the simulation output data. Salter et al. [136] require the knowledge of physical observations, in addition to the simulation output. These procedures require complete data, meaning for each run, the entire output has to be returned by the simulation. However, in seeking high levels in both performance and fidelity, modern simulation codes may produce partially observed outputs. Hung et al. [80] have developed an EM algorithm to address the issue of partially observed output prior to constructing a surrogate, and their method

is included for comparison in this chapter (see Section 2.4). In this chapter, we focus on extending the dimension-reduction approach to incorporate partially observed outputs.

The presence of partially observed output can be attributed to various causes of code failures. One cause is failure in parallel computing environments where separate computations are scattered over a large number of computing nodes. If a computing node experiences failure during calculations, only some of the calculations may be completed. Another cause is related to numerical calculations embedded in simulation codes. For example, in a simulation that involves solving a system of nonlinear equations, if the system corresponding to a response is inconsistent or particularly ill-conditioned, then no meaningful solution may be found numerically. Another cause comes from simulations where some responses in an output are undefined. It is not always easy to identify a single underlying cause. Consider a parallel computing environment where a response is not returned because a simulation is terminated by the environment when it exceeds a time limit. The response could be missing because the time limit was set too low, but it could also be missing because the numerical calculations within the simulation would never have terminated. Regardless of the cause, the presence of partially observed outputs renders most surrogate methods inapplicable. Partially observed outputs can be viewed in the context of data missingness. There are several classical categories of missingness mechanisms considered by statisticians: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). In practice the underlying causes of partially observed output are difficult to untangle because of the typically deterministic nature of computer code. Failure in computing nodes seems as though it can be considered MCAR, but in many cases it is MAR because run time depends on the input and longer

simulations are more likely to encounter a failure than a shorter one. There are many examples of missingness in simulation codes, including climate studies [30, 107]; fluid dynamics [78]; nuclear physics [21]; and weather dynamics [127].

Despite the common occurrence of partially observed output, there are few methods available for high-dimensional surrogate construction in this setting. The naïve approach would be to simply discard the dimension-reduction step entirely and treat each response separately. In this case, one discards the missing responses and pushes forward with surrogate construction with the available responses. The naïve approach includes locations as additional input dimensions. This inclusion allows for the correlation structures among both parameters and locations to be modeled, similar to the inclusion of time index as an additional input [13]. However, this approach can easily exceed the limit of standard GP inference when the number of data points goes above several thousands. The only recourse while staying with GP inference is to use GP approximations, which are still significantly more expensive than dimension-reduction approaches and lead to decreased accuracy. Another approach is to neglect correlations between locations and build an independent surrogate for each location. When the output contains a small number of responses, building separate surrogates often suffices [11]. However, as the number of responses increases, this approach is prohibited by its computational burden. Simplistic imputation is yet another option, where one imputes the missing values and then builds a surrogate using dimension-reduction tools (e.g., [127]). The imputation process can be done in various ways besides prediction, such as a "constant-liar" imputation used in optimization (e.g., [31]). For purposes of uncertainty quantification, these methods are dangerous as they will interpolate the imputed values with zero residual uncertainty.

For example, if the entire output from a run is missing, the imputation approach would simply interpolate the imputed values instead of representing them as missing, which seems contrary to the goals of a surrogate to faithfully represent the predictive uncertainty.

In this chapter we propose a new GP surrogate method that operates well in the presence of partially observed simulation output. The new method retains the computational efficiency found with dimension-reduction methods like those described in Higdon et al. [74]. However, in contrast to previous such approaches, the new method is not limited to complete data. This method involves two components: the imputation of missing values in the data using the principal components and an adjustment to the covariance matrix used in GP prediction. The adjustment to the covariance matrix ensures that one does not interpolate the imputed values at the missing locations. The resulting surrogate permits the use of data with partially observed output, and it has two appealing properties: (i) In the case where complete data is observed for a run, the resulting surrogate will interpolate those results and (ii) in the case where no data is observed for a run, the associated data row (even though imputed) will be ignored. The new surrogate method demonstrates robustness empirically under various missingness mechanisms, and provides improved uncertainty quantification in calibration.

The organization of this chapter is as follows. Section 2.1 describes the Fayans energy density functional (EDF), the simulation model that motivates this surrogate development. Section 2.2 introduces the setting and notation employed, and reviews the principal component GP method for surrogate construction. Section 2.3 details the imputation and covariance adjustment components in the new surrogate method alongside its properties. Section 2.4 details a numerical experiment to illustrate the surrogate performance under

multiple missingness scenarios. Section 2.5 discusses the calibration of the Fayans EDF model. Section 2.6 provides further discussions and closes the chapter.

2.1. Fayans energy density functional

The development of our proposed surrogate method is motivated by problems such as the calibration of the Fayans EDF [44, 45]. The development and refinement of EDF models have proven effective in understanding atomic nuclei [131]. The EDF model investigated in this chapter was developed by S. A. Fayans and collaborators for describing ground-state properties of nuclei [44, 45], and has since been used for nuclei predictions (e.g., see Yu and Bulgac [164], Reinhard and Nazarewicz [131]). Bollapragada et al. [21] provides a full description of the numerical implementation under consideration in this chapter. The model takes a 13-dimensional parameter as input and outputs 198 responses. Each response corresponds to a spherical, ground-state, even-even nuclear configuration and an observable class. A total of 72 nuclear configurations and 9 observable classes are considered, totalling to the 198 responses. The list of responses can be found in Table 1 of Bollapragada et al. [21].

The main relevant feature of this model is that it occasionally produces partial outputs, i.e., within one output, some out of the 198 responses are missing due to code failures, presenting a challenge in constructing a surrogate effectively. Missingness often occurs because an iterative method for solving equations is used in model evaluation, and failure is reported when the method fails to achieve a prescribed accuracy tolerance within an allotted internal iteration budget. Bollapragada et al. [21] outline the intricacies of possible failures and show that it is not likely the output is MCAR. However, no systematic

missingness mechanism is proposed. In previous analyses of another EDF model, $\approx 9\%$ of the model outputs were discarded due to such types of failures [76].

In Section 2.5 we use a dataset of 500 parameters to construct a surrogate and calibrate the Fayans EDF model. An illustration of this is presented in Figure 2.1, where out of 500 runs of the model, near 60% (359/500) have at least one missing value.

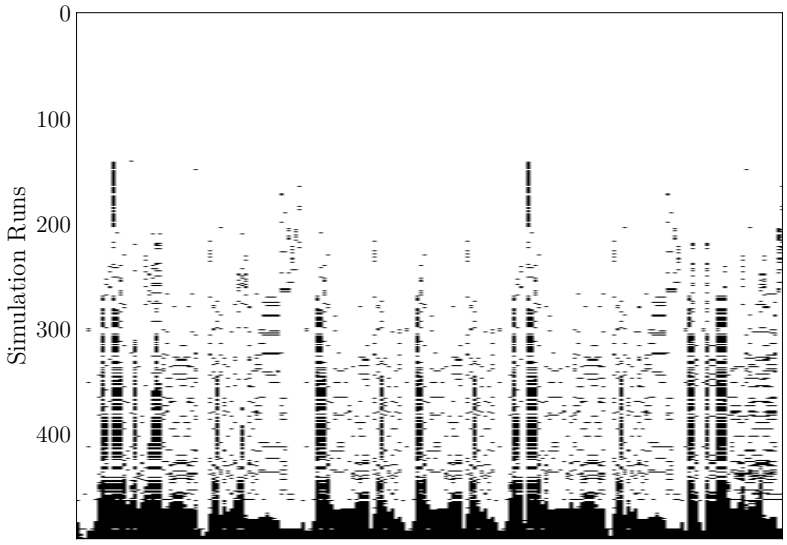


Figure 2.1. Illustration of partially observed output in the Fayans EDF case study (Section 2.5). Each horizontal line (1–500) is a length 198 simulation output. A dark patch indicates where a response is missing. The horizontal lines are sorted by number of missing values in the output.

In calibrating the Fayans EDF model, Bollapragada et al. [21] have previously employed a point minimization of the total mean-squared loss, or χ^2 loss, with respect to the parameters. Other works calibrated other EDFs in a similar manner, with minimization of the χ^2 loss [40, 89, 91, 90, 131]. Such approaches do not directly result in the uncertainty quantification sought by nuclear physicists [39]. For some other EDFs, uncertainty quantification was performed under a Bayesian framework by Higdon et al. [76] and

McDonnell et al. [113], but these works assume no missing data (or simply remove output with missing data).

The existing literature does not contain a viable methodology to solve problems like this one. One method is proposed in Ma et al. [107] to extract functional principal components, which originates from the analysis of partially observed longitudinal data. The locations in Ma et al. [107] are irregularly spaced between output dimensions, producing partial, often sparse, output at each dimension. In our problem, all responses are expected for each parameter, but missing values may arise for some responses because of various code failures. Lawrence et al. [97] have proposed to extract principal components using complete output data, and project the partially observed output data onto a subset of basis vectors to obtain the weight coefficients. This treatment of partially observed output data is effective when the locations of missing data are regular. Similarly, in Gu and Xu [66], the output data are modeled to follow a partition of complete and missing data blocks. The complete data block is then used for principal component analysis following Higdon et al. [74]. In their case, the complete data block contains a large proportion of the output data. However, it is not applicable in our setting since our missing values are irregular. Thus the equivalent complete data block retains only a small proportion of the output data, resulting in inaccurate estimation of the principal components. Another method is proposed by Hung et al. [80], in which an expectation-maximization procedure is developed to tackle the issue of missing data, coupled with a separable correlation function that reduces the computational cost. The scale of their intended application is quite small, preventing direct application for our intended application. Hung et al. [80] have studied a problem with 30 runs and only a few runs with missing data; whereas in the Fayans EDF case,

the number of simulation runs is 500, and over half of them result in a partially observed output, recall Figure 2.1. Hung et al. [80] is revisited in the numerical experiments in Section 2.4.

2.2. High-dimensional surrogates

This section introduces the notations used in this chapter and reviews standard GP modeling and its principal component based extension to high-dimensional surrogate construction [74].

2.2.1. Setting and notations. We label the user-specified parameter $\boldsymbol{\theta} \in \mathbb{R}^d$. Because surrogate inference is often deployed in calibration settings, $\boldsymbol{\theta}$ is referred to as a "parameter." We assume that for a simulation run at $\boldsymbol{\theta}$, the simulation output $\boldsymbol{f}(\boldsymbol{\theta}) \in \mathbb{R}^m$ is collected over a fixed set of locations, labeled as $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$, and thus $\boldsymbol{f}(\boldsymbol{\theta}) = (f(\boldsymbol{\theta}, \boldsymbol{x}_1), \ldots, f(\boldsymbol{\theta}, \boldsymbol{x}_m))^\mathsf{T}$, meaning the simulation evaluated at $\boldsymbol{\theta}$ will produce an output vector $\boldsymbol{f}(\boldsymbol{\theta})$ consisting of m elements. Each element in $\boldsymbol{f}(\boldsymbol{\theta})$ is considered an individual response. For example, in the Fayans EDF, the user-specified $\boldsymbol{\theta}$ represents a 13-dimensional parameter and $\boldsymbol{x} \in \mathcal{X}$ defines a nuclear configuration that corresponds to a response where $m = 198$ (see Section 2.1).

We presume that we have generated some parameters $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ from a designed computer experiment (such as a Latin Hypercube sample [114] or optimized variations thereof [84]), and the collection of simulation responses is arranged as the $n \times m$ matrix $\boldsymbol{F} = (\boldsymbol{f}(\boldsymbol{\theta}_1)^\mathsf{T}, \ldots, \boldsymbol{f}(\boldsymbol{\theta}_n)^\mathsf{T})^\mathsf{T}$. Each row represents the simulation output for one parameter at all locations $\boldsymbol{x} \in \mathcal{X}$. Each column represents the response collected at all parameters in the computer experiment. Since individual responses may be missing, we denote N as the

total number of available responses. If the collection has complete data, $N = nm$. For any matrix $\boldsymbol{A}$, we use $\boldsymbol{A}_{i,\cdot}$ to refer to the ith row, $\boldsymbol{A}_j$ to refer to the jth column of $\boldsymbol{A}$. We use $\boldsymbol{A}_{\mathcal{I}\mathcal{J}}$ to denote a submatrix of $\boldsymbol{A}$ with entries from the set of row indices $\mathcal{I} \subseteq \{1, \ldots, n\}$ and column indices $\mathcal{J} \subseteq \{1, \ldots, m\}$. Commas will be used to separate the row and column indices when the notation is ambiguous.

2.2.2. Gaussian process surrogates. A surrogate is constructed to enable the prediction of output at an unsimulated parameter. A good surrogate is designed to provide a predictive distribution, which can then be converted to point estimates and uncertainty around those estimates. In this chapter, a surrogate provides a predictive distribution for $\boldsymbol{f}(\boldsymbol{\theta})$ with a mean $\boldsymbol{\mu}(\boldsymbol{\theta})$ and a covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$. GPs offer a path to do exactly this with a multivariate normal predictive distribution [137, 57]. GP is a common choice to build a surrogate because of its flexibility and ability to interpolate.

2.2.2.1. *Univariate GP.* This section will review the basics behind GP inference. Consider $g(\boldsymbol{\theta})$ as a univariate function that takes $\boldsymbol{\theta}$ as its input. A GP model with mean function $\gamma(\cdot)$, scale parameter λ, and correlation function $\rho(\cdot, \cdot)$ assumes that for any collection of n scalar outputs (e.g., $\boldsymbol{g} = (g(\boldsymbol{\theta}_1), \ldots, g(\boldsymbol{\theta}_n))^\mathsf{T}$), follows a multivariate normal distribution with mean $\boldsymbol{\gamma} = (\gamma(\boldsymbol{\theta}_1), \ldots, \gamma(\boldsymbol{\theta}_n))^\mathsf{T}$ and covariance matrix $\lambda \boldsymbol{R}$ where $\boldsymbol{R} = (\rho(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j))_{i,j=1}^n$. Typical choices for ρ include a squared exponential or a Matérn correlation function [68], but this choice does not impact the rationale of our method and is left open in this chapter. Using the GP as a surrogate means that the predictive distribution given $\boldsymbol{g}$ at any test parameter $\boldsymbol{\theta}$ is a normal distribution with mean $\gamma(\boldsymbol{\theta}) + \boldsymbol{r}(\boldsymbol{\theta})^\mathsf{T} \boldsymbol{R}^{-1} (\boldsymbol{g} - \boldsymbol{\gamma})$ and variance $\lambda \left(\rho(\boldsymbol{\theta}, \boldsymbol{\theta}) - \boldsymbol{r}(\boldsymbol{\theta})^\mathsf{T} \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{\theta}) \right)$, where $\boldsymbol{r}(\boldsymbol{\theta}) = (\rho(\boldsymbol{\theta}, \boldsymbol{\theta}_1), \ldots, \rho(\boldsymbol{\theta}, \boldsymbol{\theta}_n))^\mathsf{T}$ is the correlation between $\boldsymbol{\theta}$ and $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$. The standard form of this inference thus requires

$\boldsymbol{R}^{-1}(\boldsymbol{g}-\boldsymbol{\gamma})$, which costs typically $O(n^3)$ operations to compute (although approximations do exist).

The computational cost is a significant obstacle that requires consideration while conducting GP inference. It is critical that the construction of the surrogate and predictions via the surrogate should be fast compared to running the simulation itself [57]. In the high-dimensional case, the total number of responses is $N = nm$. One could use off-the-shelf GP inference, where each response received from the simulation is individually modeled over both θ and x, i.e., the previously mentioned naïve approach, typically uses $O(N^3)$ computations to obtain the necessary matrix inverses and determinants. When N is moderately large (e.g., 10^4), naïvely using GP inference is difficult because of this computational burden. For our case study, $m = 198$ and hence $N \leq 10^4$ would constrain us to roughly n less than about 50. The accuracy of a GP surrogate is tied to n, and the practical results for our case study found that at $n = 50$ the surrogate is not sufficiently accurate. Researchers have considered this setting before and noted that the problem arises when the model output is multivariate with many responses. A high-dimensional output happens when the number of responses m gets large (often above > 20). There are a few choices for high-dimensional GP surrogates [14, 74, 36], with the majority of these methods seeking to reduce the required computations to $O(n^3)$, such that the computational burden effectively depends only on n.

2.2.2.2. *High-dimensional GP surrogates.* Higdon et al. [74] describe a powerful tool for building surrogates in high-dimensional output settings. Examples of successful applications include nuclear physics [76] and storm surge [92] modeling. The method works by leveraging a low-rank representation for the m-dimensional output by applying a

singular value decomposition to the matrix $\boldsymbol{F}$. Specifically, this method seeks $\boldsymbol{\Phi}$, where $\boldsymbol{\Phi}$ is an $m \times \kappa$ matrix defined by a set of κ orthonormal (i.e., $\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\Phi} = \boldsymbol{I}$) m-dimensional basis vectors. These are chosen such that for some reasonable m-dimensional centering vector $\boldsymbol{c}$, we have $\boldsymbol{F} - \mathbf{1}_n\boldsymbol{c}^\mathsf{T} \approx (\boldsymbol{F} - \mathbf{1}_n\boldsymbol{c}^\mathsf{T})\boldsymbol{\Phi}\boldsymbol{\Phi}^\mathsf{T}$, meaning we can approximately recover our simulation outputs using only the $n \times \kappa$ matrix $\boldsymbol{G} = (\boldsymbol{F} - \mathbf{1}_n\boldsymbol{c}^\mathsf{T})\boldsymbol{\Phi}$. The centering vector is often chosen as the mean of each column in $\boldsymbol{F}$. Then, the original data $\boldsymbol{F}$ are nearly recovered by $\mathbf{1}_n\boldsymbol{c}^\mathsf{T} + \boldsymbol{G}\boldsymbol{\Phi}^\mathsf{T}$. The value of κ is typically chosen to offer sufficient recovery of $\boldsymbol{F}$ from $\boldsymbol{G}$. This representation method is especially effective when there is a strong relationship among the responses of the simulation output because then κ can be made small (i.e., $\kappa \ll m$). Suppose $\boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{\Phi}^\mathsf{T}(\boldsymbol{f}(\boldsymbol{\theta}) - \boldsymbol{c})$. Then, the surrogate for the simulation is constructed as $\boldsymbol{c} + \boldsymbol{\Phi}\boldsymbol{g}(\boldsymbol{\theta})$. Thus our goal shifts from building a surrogate on an m-dimensional output $\boldsymbol{f}(\boldsymbol{\theta})$ to building a surrogate on a κ-dimensional output $\boldsymbol{g}(\boldsymbol{\theta})$.

Our goal is now to use information in the matrix $\boldsymbol{G}$ to infer on the projected output $\boldsymbol{g}(\boldsymbol{\theta})$ for any $\boldsymbol{\theta}$. Due to the orthogonal construction of $\boldsymbol{\Phi}$, each component of $\boldsymbol{g}(\cdot) = (g_1(\cdot), \ldots, g_\kappa(\cdot))^\mathsf{T}$ is modeled using an independent GP. Each $g_k(\cdot)$ then follows

$$g_k(\cdot) \sim \mathrm{GP}\left(0, \lambda_k \rho_k(\cdot, \cdot)\right), \tag{2.1}$$

where, for component k, $\lambda_k > 0$ is a scale parameter, $\rho_k(\cdot, \cdot)$ is a correlation function, and we have set the mean to be zero for ease of exposition. With data column $\boldsymbol{G}_k$, under the GP model in Equation (2.1), prediction for $g_k(\boldsymbol{\theta})$ follows a normal distribution with mean and variance given by

$$\widehat{\mu}_k(\boldsymbol{\theta}) = \boldsymbol{r}_k(\boldsymbol{\theta})^\mathsf{T}\boldsymbol{R}_k^{-1}\boldsymbol{G}_k \text{ and } \sigma_k^2(\boldsymbol{\theta}) = \lambda_k\left(\rho_k(\boldsymbol{\theta}, \boldsymbol{\theta}) - \boldsymbol{r}_k(\boldsymbol{\theta})^\mathsf{T}\boldsymbol{R}_k^{-1}\boldsymbol{r}_k(\boldsymbol{\theta})\right) \tag{2.2}$$

respectively, where $\boldsymbol{r}_k(\boldsymbol{\theta}) = (\rho_k(\boldsymbol{\theta}, \boldsymbol{\theta}_1), \ldots, \rho_k(\boldsymbol{\theta}, \boldsymbol{\theta}_n))^\mathsf{T}$ and $\boldsymbol{R}_k = (\rho_k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j))_{i,j=1}^n$. From this, the surrogate is then defined by

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \boldsymbol{c} + \boldsymbol{\Phi}(\widehat{\mu}_1(\boldsymbol{\theta}), \ldots, \widehat{\mu}_\kappa(\boldsymbol{\theta}))^\mathsf{T} \text{ and } \boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{\Phi} \begin{pmatrix} \sigma_1^2(\boldsymbol{\theta}) & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_\kappa^2(\boldsymbol{\theta}) \end{pmatrix} \boldsymbol{\Phi}^\mathsf{T}. \tag{2.3}$$

We refer to this method of constructing a surrogate as Principal Component Gaussian Process (PCGP). For PCGP methods, $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ is not of full rank when $\kappa < m$, thus the predictive distribution is often degenerate. However, similar to Higdon et al. [74], when a surrogate is used to facilitate parameter calibration, $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ is summed with a (typically diagonal) symmetric positive definite observation error covariance matrix (see Section 2.5.1). The resulting sum of the matrices is then full rank and thus $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ being only positive semi-definite does not pose a problem.

The inferences in (2.3) are used to provide the mean and covariance of the surrogate for $\boldsymbol{f}(\boldsymbol{\theta})$. If there are partially observed simulation outputs, there are reasonable ways to approximate the principal component matrix $\boldsymbol{\Phi}$ and the centering vector $\boldsymbol{c}$ (see, e.g., Roweis [134], Tipping and Bishop [151]). The details of approximating $\boldsymbol{\Phi}$, an expectation-maximization algorithm inspired by Roweis [134], are included in Appendix A. However, partially observed simulation output in $\boldsymbol{F}$ means that $\boldsymbol{G}$ cannot be computed. Take the ith row of the data for example, if $\boldsymbol{f}(\boldsymbol{\theta}_i)^\mathsf{T}$ has one or more missing responses, then $\boldsymbol{G}_{i,\cdot}$ cannot be computed. When m is large, it can often be the case that at least one response can be missing in a row, which leaves the rest of the data in the same row unusable without remedy. We then risk tossing out a large amount of data because of a few failures. Higdon

et al. [74] also commented on the necessity of complete data for the use of PCGP. This leaves us at a crossroads. If we use the PCGP approach, we cannot handle missing data. If we do not adopt the PCGP approach, the surrogate computation may be impossible due to the explosive increase in required computations. This chapter explains how one can expand the PCGP surrogate methodology to handle missing data.

2.3. Fast surrogates with missing data

We now introduce our method for constructing a surrogate for high-dimensional outputs in the presence of missing data. Specifically, we propose an imputation method for the missing observations during the computation of $\boldsymbol{G}$ alongside a novel adjustment of the covariance matrix. The resulting method is computationally inexpensive and therefore the benefit of fast construction using PCGP is retained.

Section 2.3.1 describes and illustrates the desired properties of the proposed surrogate. Sections 2.3.2 and 2.3.3 detail the imputation and covariance adjustment procedures. Section 2.3.4 shows how the proposed method delivers the desired surrogate properties. Section 2.3.5 describes the hyperparameter estimation used in the construction of the surrogate. Section 2.3.6 provides additional justification for the choice of covariance adjustment.

2.3.1. Desired surrogate properties. In constructing this surrogate, we consider it necessary to have two desirable properties, the recovery of complete data rows and the ignorance of entirely missing data rows. This means that we want to nearly interpolate to chosen precision where the output is complete and disregard rows where the output is entirely missing.

The motivation for these desired properties can be explained through a thought experiment. Let the collected data $\boldsymbol{F}$ be an $n \times m$ matrix, corresponding to simulation outputs at the parameters $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$. Let $\boldsymbol{F}$ only have data available in $n_0 < n$ rows, and for these rows, the outputs are complete. Let the $n_0 \times m$ matrix $\boldsymbol{F}^{(0)}$ denote the available data rows; without loss of generality, we take $\boldsymbol{F}^{(0)}_{i,\cdot} = \boldsymbol{F}_{i,\cdot}$ for all $i \leq n_0$. We assume that all of the remaining rows are entirely missing; this can occur, for example, when the computer running the simulations went down after completing n_0 rows. Suppose we separately construct two surrogates using a proposed method: one with data $\boldsymbol{F}$ and one with data $\boldsymbol{F}^{(0)}$. Let the surrogate with $\boldsymbol{F}$ be described by $(\boldsymbol{\mu}(\boldsymbol{\theta}), \boldsymbol{\Sigma}(\boldsymbol{\theta}))$, and the surrogate with $\boldsymbol{F}^{(0)}$ be described by $(\boldsymbol{\mu}^{(0)}(\boldsymbol{\theta}), \boldsymbol{\Sigma}^{(0)}(\boldsymbol{\theta}))$. The recovery of complete data rows means that the surrogate should nearly interpolate for the rows with complete data; that is, $\boldsymbol{\mu}(\boldsymbol{\theta}_i) \approx \boldsymbol{f}(\boldsymbol{\theta}_i)$ and $\boldsymbol{\mu}^{(0)}(\boldsymbol{\theta}_i) \approx \boldsymbol{f}(\boldsymbol{\theta}_i)$ for all $i \leq n_0$. The only interpolation error that should exist is due to the dimension reduction ($\kappa < m$) representation we have chosen. The ignorance of entirely missing data rows means that the surrogate should not depend on any of the rows $\boldsymbol{f}(\boldsymbol{\theta}_i)^\mathsf{T}$ for any i larger than n_0; that is, $\boldsymbol{\mu}(\boldsymbol{\theta}) = \boldsymbol{\mu}^{(0)}(\boldsymbol{\theta})$ and $\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{\Sigma}^{(0)}(\boldsymbol{\theta})$ for any $\boldsymbol{\theta}$.

Let us see how obvious approaches, namely the naïve and simplistic imputation approaches described in the introduction, fare with respect to these goals. The naïve method, where one treats each data point individually, would interpolate all observed points and ignore the remainder. Thus it meets our goals, but this method becomes computationally intractable in our settings because of the large N problem. The simplistic imputation approach would impute all rows when dealing with $\boldsymbol{F}$ and then interpolate that imputation. This implies it would nearly interpolate all observed rows, but the predictions from the simplistic imputation approach using $\boldsymbol{F}^{(0)}$ and $\boldsymbol{F}$ are inconsistent.

This inconsistency implies that throwing out or keeping rows with fully missing output will give different predictions.

The method proposed in this chapter is able to nearly interpolate complete rows and ignore missing rows similar to the naïve method, yet it remains computationally tractable. While these criteria do not guarantee interpolation of outputs in the partially observed output case, it is our expectation and experience that by matching these two extremes, the predictions in the partially observed case offer notably better predictions than the simplistic imputation approach. We later justify this with simulation experiments in Section 2.4.

2.3.2. Gaussian process-based imputation. We will assume the GPs modeling $g_k(\cdot)$ are stationary, and thus, without loss of generality, let $\rho_k(\boldsymbol{\theta}, \boldsymbol{\theta}) = 1$ for $k = 1, \ldots, \kappa$. Furthermore, this section will assume the centering constant $\boldsymbol{c} = \mathbf{0}$ for ease of exposition. If we consider the relationship that our output is (nearly) $\boldsymbol{\Phi}\boldsymbol{g}(\boldsymbol{\theta})$, its covariance, following (2.1), is of the form

$$\boldsymbol{\Phi}\mathrm{Cov}(\boldsymbol{g}(\boldsymbol{\theta}))\boldsymbol{\Phi}^{\mathsf{T}} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^{\mathsf{T}}, \tag{2.4}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix $\mathrm{diag}(\lambda_1, \ldots, \lambda_\kappa)$, due to the independence among the κ GPs in (2.1). These values are presumed decided in advance. For example, a reasonable choice deployed in our examples sets $\lambda_1, \ldots, \lambda_\kappa$ to the square of the corresponding singular values from the decomposition of $\boldsymbol{F}$.

The covariance matrix in (2.4), similar to the one in (2.3), may not be of full rank. Therefore, it is difficult to use (2.4) because the low-rank nature implies that one can have effectively "fully observed" $\boldsymbol{f}(\boldsymbol{\theta})$ after observing only κ entries. Instead, we will use a

full-rank extension of this matrix. Pick $\varepsilon > 0$ such that $\lambda_k > \varepsilon$ for all $k \leq \kappa$. This choice of ε ensures that this covariance matrix extension for $\boldsymbol{f}(\boldsymbol{\theta})$, defined as

$$\boldsymbol{B} = \boldsymbol{\Phi}(\boldsymbol{\Lambda} - \varepsilon \boldsymbol{I})\boldsymbol{\Phi}^{\mathsf{T}} + \varepsilon \boldsymbol{I}, \tag{2.5}$$

is of full rank, and thus $\boldsymbol{B}$ and any principal submatrix of $\boldsymbol{B}$ are invertible.

Using the covariance matrix in (2.5), we impute the missing observations to build $\tilde{\boldsymbol{G}}$, an $n \times \kappa$ matrix with complete entries, to replace $\boldsymbol{G}$.

Let $\mathcal{J}(i) \subseteq \{1, \ldots, m\}$ be the set of column indices where data are not missing in $\boldsymbol{f}(\boldsymbol{\theta}_i)^{\mathsf{T}}$. Then, let $\boldsymbol{F}_{i,\mathcal{J}(i)} = (f(\boldsymbol{\theta}_i, \boldsymbol{x}_j))_{j \in \mathcal{J}(i)}^{\mathsf{T}}$ be the row vector that contains the available data for parameter $\boldsymbol{\theta}_i$ and $\boldsymbol{B}_{\mathcal{J}(i),\cdot}, \boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)}$ be the submatrices corresponding to the indices. Because this section assumes a centering vector of zeros,

$$\begin{pmatrix} \boldsymbol{f}(\boldsymbol{\theta}_i) \\ \boldsymbol{F}_{i,\mathcal{J}(i)}^{\mathsf{T}} \end{pmatrix} \sim N\left(\begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix}, \begin{pmatrix} \boldsymbol{B} & \boldsymbol{B}_{\mathcal{J}(i),\cdot}^{\mathsf{T}} \\ \boldsymbol{B}_{\mathcal{J}(i),\cdot} & \boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)} \end{pmatrix} \right),$$

where this is a degenerate multivariate normal distribution, as $\boldsymbol{B}_{\mathcal{J}(i),\cdot}$ is a submatrix of $\boldsymbol{B}$. Subsequently the conditional mean and covariance of $\boldsymbol{f}(\boldsymbol{\theta}_i)$ given $\boldsymbol{F}_{i,\mathcal{J}(i)}^{\mathsf{T}}$, following standard normal theory (e.g., Gelman et al. [51]), are

$$\mathbb{E}(\boldsymbol{f}(\boldsymbol{\theta}_i) | \boldsymbol{F}_{i,\mathcal{J}(i)}^{\mathsf{T}}) = \left(\boldsymbol{F}_{i,\mathcal{J}(i)} \boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)}^{-1} \boldsymbol{B}_{\mathcal{J}(i),\cdot} \right)^{\mathsf{T}} \quad \text{and}$$

$$\mathrm{Cov}(\boldsymbol{f}(\boldsymbol{\theta}_i) | \boldsymbol{F}_{i,\mathcal{J}(i)}^{\mathsf{T}}) = \boldsymbol{B} - \boldsymbol{B}_{\mathcal{J}(i),\cdot}^{\mathsf{T}} \boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)}^{-1} \boldsymbol{B}_{\mathcal{J}(i),\cdot},$$

respectively. We propose then to infer $\boldsymbol{G}_{i,k}$ using the conditional quantities

$$\tilde{\boldsymbol{G}}_{i,k} = \boldsymbol{F}_{i,\mathcal{J}(i)} \boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)}^{-1} \boldsymbol{B}_{\mathcal{J}(i),\cdot} \boldsymbol{\Phi}_{\cdot,k} \quad \text{and} \tag{2.6}$$

$$u_{ik} = \boldsymbol{\Phi}_{\cdot,k}^{\mathsf{T}} \left(\boldsymbol{B} - \boldsymbol{B}_{\mathcal{J}(i),\cdot}^{\mathsf{T}} \boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)}^{-1} \boldsymbol{B}_{\mathcal{J}(i),\cdot} \right) \boldsymbol{\Phi}_{\cdot,k}. \tag{2.7}$$

This proposed inference is the key insight for our approach. While (2.6) is a reasonable use of our principal components for imputation, (2.7) is a valuable measure of goodness for the imputation, since the conditional variances reflect the imputation uncertainty. The quantities u_{ik} from (2.7) are used to adjust our GP's covariance matrix to account for the uncertainty due to imputation. For the suggested inferences to be practically useful, the inversion of the submatrix of $\boldsymbol{B}$ should be efficient. The details of such inversion are provided in Appendix A.

2.3.3. Covariance adjustment. There is a need to account for the additional uncertainty from the imputation of missing data. We propose adjusting the covariance matrix by adding an extra term to $\boldsymbol{R}_k$, the original covariance matrix in (2.2), in order to model the increase in variance of prediction due to imputation. Defining the scaled predictive variances by $w_{ik} = u_{ik}/\lambda_k$, the proposed adjusted covariance matrix is

$$\tilde{\boldsymbol{R}}_k = \boldsymbol{R}_k + \beta_k \text{diag}(v_{1k}, \ldots, v_{nk}), \qquad \text{where } v_{ik} = \min \left\{ \eta, \frac{w_{ik}}{(1 - w_{ik})^{\alpha}} \right\}, \tag{2.8}$$

where $\eta > 0$ is a large constant introduced to prevent infinite values from corrupting our linear algebra, and $\alpha > 0$ and $\beta_k > 0$ are hyperparameters that affect the magnitude of the additional term. The intuition of covariance adjustment in Equation (2.8) is that the variance will be higher in the rows where fewer responses are available. In particular, α controls the penalty for extreme missingness in an output and β_k controls the amplification of additional variances for component k. In the case where we set $\beta_k = 0$, the variance

added from imputation is not accounted for, and the imputed values are treated as observed and interpolated.

Using (2.6) on every row with missing data, $\tilde{\boldsymbol{G}}_k$ is the completed kth column of the low-dimensional data. Combining that with our covariance in (2.8) gives that prediction for $g_k(\theta)$ with

$$\tilde{\mu}_k(\boldsymbol{\theta}) = \boldsymbol{r}_k(\boldsymbol{\theta})^\mathsf{T} \tilde{\boldsymbol{R}}_k^{-1} \tilde{\boldsymbol{G}}_k \quad \text{and} \tag{2.9}$$

$$\tilde{\sigma}_k^2(\boldsymbol{\theta}) = \lambda_k \left(\rho_k(\boldsymbol{\theta}, \boldsymbol{\theta}) - \boldsymbol{r}_k(\boldsymbol{\theta})^\mathsf{T} \tilde{\boldsymbol{R}}_k^{-1} \boldsymbol{r}_k(\boldsymbol{\theta}) \right), \tag{2.10}$$

where $\boldsymbol{v}_k = (v_{1k}, \ldots, v_{nk})^\mathsf{T}$. The completed version of the surrogate provides a prediction for $\boldsymbol{f}(\boldsymbol{\theta})$ that has a multivariate normal distribution with updated mean and covariance

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \boldsymbol{\Phi}(\tilde{\mu}_1(\boldsymbol{\theta}), \ldots, \tilde{\mu}_\kappa(\boldsymbol{\theta}))^\mathsf{T}, \text{ and } \boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{\Phi} \begin{pmatrix} \tilde{\sigma}_1^2(\boldsymbol{\theta}) & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \tilde{\sigma}_\kappa^2(\boldsymbol{\theta}) \end{pmatrix} \boldsymbol{\Phi}^\mathsf{T}. \tag{2.11}$$

Recall that for this section we set $\boldsymbol{c} = \boldsymbol{0}$, thus these terms agree closely with the PCGP terms in (2.3), where we have only modified the predictions of means and variances. If there is no missingness in our simulation output, the expressions in (2.3) and (2.11) will exactly agree.

2.3.4. Properties of the proposed surrogate. The following result guarantees that if there is no missing data in the ith row (i.e., $\mathcal{J}(i) = \{1, \ldots, m\}$), then $\tilde{\boldsymbol{G}}_{i,\cdot}$ recovers $\boldsymbol{G}_{i,\cdot}$ without additional uncertainty.

THEOREM 2.1 (Recovery of fully observed row). *If $\mathcal{J}(i) = \{1, \ldots, m\}$, then $\tilde{\mu}_k(\boldsymbol{\theta}_i) = \boldsymbol{f}(\boldsymbol{\theta}_i)^\mathsf{T} \boldsymbol{\Phi}_{\cdot,k}$ and $\tilde{\sigma}_k^2(\boldsymbol{\theta}_i) = 0$ for $k = 1, \ldots, \kappa$.*

The proof is given in Appendix A. On the other hand, notice that if there is no data in the ith row (i.e., $\mathcal{J}(i)$ is the empty set), then

$$u_{ik} = \boldsymbol{\Phi}_{\cdot,k}^{\mathsf{T}} \left(\boldsymbol{\Phi}(\boldsymbol{\Lambda} - \varepsilon \boldsymbol{I})\boldsymbol{\Phi}^{\mathsf{T}} + \varepsilon \boldsymbol{I} \right) \boldsymbol{\Phi}_{\cdot,k} = \lambda_k - \varepsilon + \varepsilon = \lambda_k.$$

Our next result establishes that our method naturally ignores these rows of missing data.

THEOREM 2.2 (Ignorance of missing rows). *Define the kth surrogate component, where all rows with entirely missing data are excluded from construction, by $\underline{\tilde{\mu}}_k$ and $\underline{\tilde{\sigma}}_k^2$ from* (2.9) *and* (2.10)*, respectively. If $\mathcal{J}(i) = \emptyset$, then for any $\alpha > 0$, $\beta_k > 0$, $\boldsymbol{\theta}$, and $k \in \{1, \ldots, \kappa\}$, we have that $\tilde{\mu}_k(\boldsymbol{\theta}) - \underline{\tilde{\mu}}_k(\boldsymbol{\theta}) \to 0$ and $\tilde{\sigma}_k^2(\boldsymbol{\theta}) - \underline{\tilde{\sigma}}_k^2(\boldsymbol{\theta}) \to 0$ as $\eta \to \infty$ in Equation* (2.8).

The proof is provided in Appendix A. Although Theorem 2.2 is stated in a limit as our numerical stabilizing parameter η goes to infinity, we note that in our deployments on real problems that setting η to 10 results in reasonable ignoring behavior.

2.3.5. Estimation of hyperparameters in surrogate construction. To construct the surrogate is to fit the hyperparameters for all $k = 1, \ldots, \kappa$ components. Besides the covariance adjustment coefficient β_k, additional hyperparameters are implicitly included in the notation of the covariance function $\rho_k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \rho_k(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\psi}_k)$, where $\boldsymbol{\psi}_k$ encapsulates the covariance function hyperparameters. Within this section, we use the notation $\tilde{\boldsymbol{R}}_k(\beta_k, \boldsymbol{\psi}_k)$ to highlight the dependence of the adjusted covariance matrix $\tilde{\boldsymbol{R}}_k$ on both β_k and $\boldsymbol{\psi}_k$. The surrogate construction involves the maximization of the log-likelihood with respect to the hyperparameters, which can be simplified to

$$\max_{\beta_k, \boldsymbol{\psi}_k} \Big\{ -\frac{1}{2} \log |\tilde{\boldsymbol{R}}_k(\beta_k, \boldsymbol{\psi}_k)| - \frac{1}{2\lambda_k} \tilde{\boldsymbol{G}}_k^{\mathsf{T}} \tilde{\boldsymbol{R}}_k(\beta_k, \boldsymbol{\psi}_k)^{-1} \tilde{\boldsymbol{G}}_k \Big\},$$

where $|\boldsymbol{A}|$ returns the determinant of $\boldsymbol{A}$. The evaluation of the log-likelihood involves the decomposition of the covariance matrix that costs $O(n^3)$ operations, where n is the number of parameters. In our deployment, we use a gradient-based L-BFGS optimization solver to approximate the maximum likelihood estimate.

2.3.6. Rationale for the extra term. We now provide justification for the choice of the extra term in (2.8) alongside hyperparameter α. While w_{ik} can be used directly as the added variance terms, it is insufficient to represent the intuition that (i) if the row has complete data, no adjustments should be made, and (ii) if no data are observed in the row, the added term should be infinite, because we have no information about the row. In the case where $\alpha = 0$, $v_{ik} = w_{ik}$ does not achieve infinity when no data are observed. When $\alpha > 0$, the added term $v_{ik} = 0$ when $w_{ik} = 0$, and $v_{ik} = \infty$ when $w_{ik} = 1$. Moreover, the magnitude of α controls the rate of increase of variance v_{ik} as w_{ik} increases. The variance term v_{ik} represents the uncertainty introduced from observing only at the index set $\mathcal{J}(i)$ for row i. As α increases, v_{ik} increases with a faster rate to infinity as w_{ik} approaches 1. The effect of α is illustrated in Figure 2.2. The hyperparameter β_k is introduced to control the inflation (or deflation) of additional variances for the kth surrogate component. The β_k value can vary across the components. When $\beta_k = 0$, the additional variances have no effect to the surrogate, as if there is no uncertainty due to data missingness. When β_k is infinite, the overwhelming additional variances will cause any row with any amount of missing data to be ignored. When $\beta_k = 1$, no adjustment is made. The hyperparameters $\beta_k, k = 1, \ldots, \kappa$, are estimated in the surrogate construction.

A numerical experiment is presented in Appendix A that investigates the choice of α and if β_k should be either equal to 1 or optimized as a hyperparameter. We found the

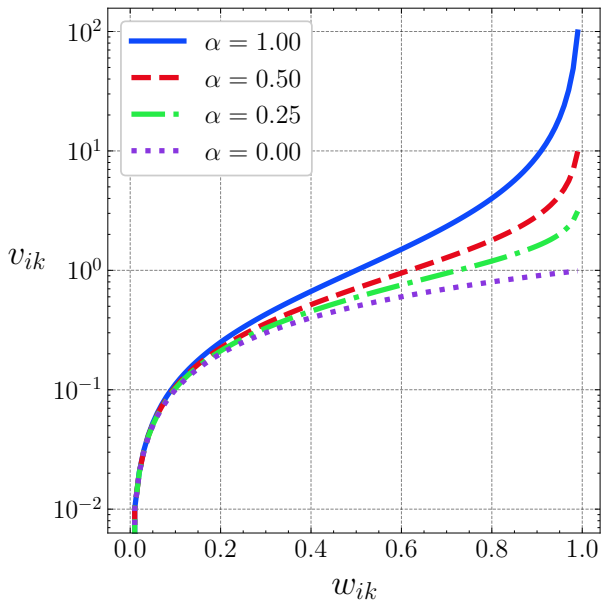


Figure 2.2. Illustrations of the effect of hyperparameter α in the variance term v_{ik} in Equation (2.8). Except for $\alpha = 0$, v_{ik} approaches infinity as w_{ik} approaches 1.

best behavior when the β_k are optimized and found $\alpha > 0.5$ could negatively impact the prediction accuracy. We suggest $\alpha = 0.3$ as a default choice.

2.4. Numerical experiments

We now present numerical experiments evaluating the performance of our proposed surrogate method.

2.4.1. Setup of numerical experiments. We compare the proposed method with other methods that employ common strategies for dealing with missing data. A total of six methods are considered. We label our method as "PCGPwM," for PCGP with Missingness. Among the other methods, the first method ("GP-OM", standing for GP-Omit-Missingness) is to omit the missing data, which is the naïve approach that treats the N responses

individually and discards the missing points. The second method ("colGP") is to treat locations as independent and construct a GP for each of m locations. Gu and Berger [64] refer to colGP as the "Many Single" emulator approach and have commented on its computational intractability when m is large. Gu and Berger [64] resolves this computation issue by aligning common Gaussian process parameters across locations, but this fix is not needed here as our m is relatively small. We consider the expectation-maximization-based method by Hung et al. [80] as the third method ("EMGP"). The next two methods involve the simplistic imputation approach with two different imputation procedures. The two imputation procedures considered are the k-nearest neighbor method (see, e.g., Altman [4]), and Bayesian ridge regression [150]. Since the PCGP method is used on the imputed data, these two methods are termed "PCGP-kNN" and "PCGP-BR," respectively. In addition to these methods, we compare PCGPwM with a baseline method where the principal components are assumed known. This comparison is reported in Appendix A.

GP-OM treats the missing data as if they were not present. We discard any response that is missing. After omitting the missing data, the remaining data do not necessarily form a data matrix. For example, one row may have no missing data, while another row may have one or more entries removed. Instead, the remaining data are stacked into a column and a single GP is constructed as a surrogate. Due to the stacking of data, this method becomes intractable as the computational complexity is $O(N^3)$, with N being the number of available data points. The colGP method treats each $\boldsymbol{x} \in \mathcal{X}$ as independent locations and constructs m GPs. If m is large, computational costs may not permit the use of colGP simply due to the number of GPs to be constructed. The EMGP method estimates the missing values via an EM algorithm and uses a separable covariance matrix

to speed up its GP construction. For each of the simplistic imputations, the missing data are first imputed and PCGP is then applied to construct the surrogate. The k-nearest neighbor method imputes the missing value by averaging the k closest available data points, using the Euclidean distance between couples $(\boldsymbol{\theta}, \boldsymbol{x})$. The Bayesian ridge regression fits a regression model and imputes the missing values with the predictions of the model. In the implementation of the imputation methods using `scikit-learn` [123], missing values in each data column are imputed in a round-robin fashion for 10 iterations, and the result from the final iteration is returned. See the package documentation on `IterativeImputer` for details.

We use four functions as test examples, namely the borehole function, piston function, wingweight function, and OTLcircuit function. These are common test functions for emulation and uncertainty quantification, located at `http://www.sfu.ca/~ssurjano/index.html`. The input variables of each function are partitioned into $(\boldsymbol{\theta}, \boldsymbol{x})$. Our proposed method does not address the source of missingness in the data. To test the robustness of the method, we generate partially observed output with the three missing mechanisms mentioned in the Introduction of this chapter. The missing mechanisms are MCAR, MNAR, and MAR. Under MCAR, any response has the same probability of missing. Under MNAR, the response missingness depends on unobserved quantities, for example, the value of the response. Under MAR, the response missingness only depends on quantities that are observed, for example, the parameter and location in our case. For each of these missingness mechanisms, the percentage of missingness is specified. The methods are tested at 1%, 5%, and 25% missingness levels. With MNAR, values of the borehole and the wingweight functions are missing when the evaluated function value exceeds a threshold;

and the values of the OTLcircuit and the piston functions are missing according to the probability determined by a logistic model over the fixed locations. With MAR, the missingness is randomly assigned over a subset of the fixed locations. The test functions and the details of the MNAR and MAR generation are included in Appendix A.

The size of the output, m, is set as 15. The set of locations $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$ is uniformly sampled from their respective ranges. The number of parameters, n, are set to take values $n = 50, 100, 250, 1000$, and 2500. The parameters are sampled using a Latin Hypercube design [137] in the unit hypercube, and then scaled to their respective ranges. Denoting a missingness scenario to be the missing mechanism and the missing fraction combined, nine scenarios are considered: (MCAR, 1%), (MCAR, 5%), (MCAR, 25%), (MNAR, 1%), (MNAR, 5%), (MNAR, 25%), (MAR, 1%), (MAR, 5%), and (MAR, 25%). We construct a surrogate using each method by supplying it the output of the model, for each combination of n and missingness scenario. To efficiently compare the methods, we fix the locations, the parameters, and the set of missing values across the four methods in one replication. Each experiment is run for 20 replications for all test functions. Each replication is given 1 hour of run time and is canceled if the surrogate construction and prediction takes longer than that.

All methods are implemented through the Python package `surmise` [128], a modular package that interfaces different statistical emulation and calibration methods. Our proposed method is implemented under the name `PCGPwM`. For GP-OM, `GPy` [56] is used to construct the surrogate. The colGP method is implemented under the name `colGP`. The EMGP method is implemented under the name `EMGP`. For PCGP-kNN and PCGP-BR, first the imputations are performed with `scikit-learn` [123], then the completed data are

supplied to the `PCGP` method to construct a surrogate. The simplistic imputation approach is implemented under the name `PCGPwImpute` with an option of which imputation method to use.

2.4.2. Results of comparison experiments. This section will discuss representative results for the (MNAR, 5%) missingness scenario. For a full description of the results of our numerical experiments, we direct the reader to Appendix A.

All methods are competitive in computation, except GP-OM, which costs roughly 30 times as long for all n. Naïvely omitting missing values in the data destroys the regular structure, and further leads to tractability issues. In addition, EMGP does not complete at the largest data size within the time limit. The remaining constructions are completed under the time limit. Note that although the colGP method completes all surrogate constructions within time limit here with $m = 15$, its computation time may be prohibitive as m is too large, further explored in Section 2.4.3.

The quality of the surrogate methods is measured by the root mean squared error (RMSE), the coverage probability of the 90% prediction interval (90% coverage), and the width of the same interval (90% width). While RMSE concerns the predictive accuracy of the mean, 90% coverage and 90% width are empirical measures of the quality of the surrogate's uncertainty quantification. The measures are evaluated against holdout simulation runs, and any missing values in the runs are excluded from evaluation.

Figure 2.3 shows the RMSEs for the surrogate methods being compared. For the borehole and the wingweight functions, the RMSE decreases for all methods except the simplistic imputation methods as N increases; whereas for the piston and the OTLcircuit

functions, the RMSEs decrease for all methods. GP-OM shows to be generally not competitive in both its computation time and predictive accuracy. The simplistic imputation methods, PCGP-kNN and PCGP-BR, fail to circumvent the issue of missing values, especially when the missingness is MNAR. Since the simplistic imputation approach relies on the availability of close neighbors to the missing points, if the function values are never observed within a certain region, the missing values would be imputed with far-away values of little relevant information. As a result, they result in poor predictions. The EMGP method performs comparably with the simplistic imputation methods, and somewhat better in the case of the wingweight function. The colGP method performs well across all methods in predictive accuracy, especially in the piston and OTLcircuit test functions. The continuous improvement with colGP as N increases shows a potential drawback in using principal-component methods for dimension reduction. Similar conclusions are drawn for (MNAR, 1%) and (MNAR, 25%). We find that in the case of MCAR and MAR, the accuracy of all methods improves as N grows, where PCGPwM performs better than all methods except for colGP.

The 90% coverage records how often the simulation response is contained in the interval produced by the surrogate. The quality of the surrogate is measured by comparing the 90% empirical coverage with the nominal level (in this case, 90%). A coverage close to the nominal level with a narrow width is an indication of a good surrogate. Figure 2.4 shows the 90% coverage for the surrogate methods being compared. GP-OM results in overcoverage in the borehole function and significantly undercoverage in the remaining functions. By investigating the interval widths, we observe that GP-OM produces a prediction interval too wide in the borehole and too narrow in the others. The EMGP method attains adequate

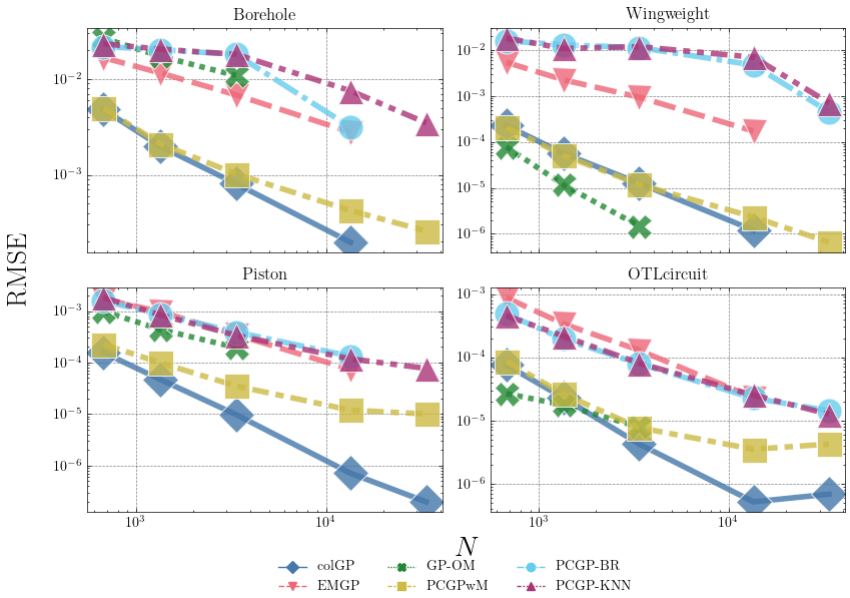


Figure 2.3. Comparison (log-log scale) of prediction accuracy of surrogate methods for (MNAR, 5%) scenario as the number of available data points increases.

coverage in the piston and wingweight functions, while undercovers in the borehole and wingweight function. As the data size grows, the coverage for the wingweight function increases and attains the prescribed level. The simplistic imputation methods, PCGP-kNN and PCGP-BR, exhibit overcoverages in the borehole and wingweight functions, as a result of wide intervals. PCGPwM achieves the prescribed coverages while being able to provide sharper predictions, indicated by narrower intervals. In the MCAR and MAR scenarios, both the coverage and the prediction interval widths improve as N grows for all methods.

Overall, the proposed PCGPwM method performs generally well in terms of RMSEs and is robust to different types of data missingness. The method also preserves the efficiency of surrogate construction found in dimension-reduction methods such as PCGP [74].

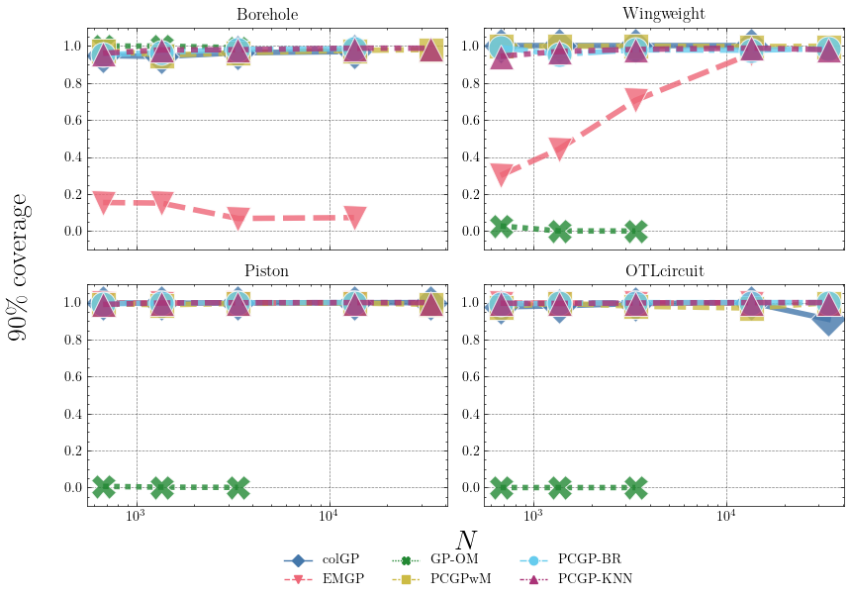


Figure 2.4. Comparison of 90% coverage of surrogate methods for (MNAR, 5%) scenario as the number of available data points increases.

2.4.3. Additional comparisons of PCGPwM and colGP at higher output dimension. The predictive performance of colGP is laudible but it may be slow to construct at higher output dimensions. To further compare PCGPwM and colGP in terms of computational cost, we have conducted an additional experiment. We have increased the output dimension to $m = 200$ and focused on the performance at larger $n = (1000, 2500)$ with the borehole function. A maximum of 4 hours is permitted for each surrogate construction. Table 2.1 reports the respective construction time, RMSE, 90% coverage, and 90% width. PCGPwM achieves a higher RMSE than colGP, but maintains the right coverage with a narrower width. At $n = 2500$, the construction time of colGP exceeds the allowed time limit with higher dimensions and the computation is aborted. According to the time scaling, the construction time would have taken colGP an estimated 18 hours.

	Construction time (s)		RMSE ($\times 10^{-4}$)		90% coverage		90% width	
n	PCGPwM	colGP	PCGPwM	colGP	PCGPwM	colGP	PCGPwM	colGP
1,000	640	4,225	3.9	2.5	0.912	0.962	0.764	0.920
2,500	9,680	–	0.62	–	0.981	–	0.571	–

Table 2.1. Construction times (in seconds) and predictive accuracies of PCGPwM and colGP at output dimension $m = 200$.

2.5. Calibrating Fayans energy density functional with missingness

The section describes the case study mentioned in Section 2.1 that relies on the described surrogate method to conduct calibration. We review a surrogate-based calibration framework in Section 2.5.1. We present the results of applying the proposed method to the case of calibrating the Fayans EDF in Section 2.5.2.

2.5.1. Bayesian calibration with a surrogate. Suppose $\boldsymbol{y} = (y_1, \ldots, y_m)^\mathsf{T}$ is a set of observations from the physical system that the simulation $\boldsymbol{f}(\boldsymbol{\theta})$ is representing. The differences between the observations and the simulation responses are assumed to follow a multivariate normal distribution with zero mean and covariance $\boldsymbol{W}$. This follows the canonical Kennedy and O'Hagan [87] framework with the assumption that a systematic bias is negligible. The Kennedy and O'Hagan [87] framework has generated much research interest; for examples, see Higdon et al. [75], Williams et al. [161], Bayarri et al. [14], Higdon et al. [74], Brynjarsdóttir and O'Hagan [26].

Let π denote a probability density and $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ be the conditional probability density of $\boldsymbol{\theta}$ given $\boldsymbol{y}$. The purpose for calibration is to infer about the parameter $\boldsymbol{\theta}$. In the Bayesian setting, we are interested in the quantity $\pi(\boldsymbol{\theta}|\boldsymbol{y})$, which is the posterior density of $\boldsymbol{\theta}$. By Bayes rule, given a prior density $\pi(\boldsymbol{\theta})$, we have $\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$, where $\propto$ denotes equality up to a constant multiplier. Given the distribution of the differences, the

expression is expanded to be

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto |\boldsymbol{W}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{f}(\boldsymbol{\theta}))^{\mathsf{T}}\boldsymbol{W}^{-1}(\boldsymbol{y}-\boldsymbol{f}(\boldsymbol{\theta}))\right)\pi(\boldsymbol{\theta}),$$

where $\boldsymbol{W}$ is the covariance matrix of the observation error. Often in a physical experiment, this covariance matrix is diagonal, as in the Fayans EDF model.

Since missing data may be present, $\boldsymbol{f}(\boldsymbol{\theta})$ refers to the hypothetical output responses at a given parameter. When a surrogate, defined by $\boldsymbol{\mu}(\boldsymbol{\theta})$ and $\boldsymbol{\Sigma}(\boldsymbol{\theta})$, constructed with data $\boldsymbol{F}$ is used in place of the simulation model, which is a common strategy in the calibration literature, the posterior density is revised as

$$\pi(\boldsymbol{\theta}|\boldsymbol{y},\boldsymbol{F}) \propto |\boldsymbol{W}+\boldsymbol{\Sigma}(\boldsymbol{\theta})|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu}(\boldsymbol{\theta}))^{\mathsf{T}}\left(\boldsymbol{W}+\boldsymbol{\Sigma}(\boldsymbol{\theta})\right)^{-1}(\boldsymbol{y}-\boldsymbol{\mu}(\boldsymbol{\theta}))\right)\pi(\boldsymbol{\theta}), \tag{2.12}$$

by considering the joint distribution of $\boldsymbol{y}$ and $\boldsymbol{F}$ being again multivariate normal. For a fixed simulation model sample $\boldsymbol{F}$ (with missing data), this expression can be used to draw from the posterior using MCMC methods.

2.5.2. Surrogate construction and calibration for the Fayans EDF. In this section we apply the previously described surrogate methods for the calibration of the Fayans EDF model to provide uncertainty estimates for the parameter, using the knowledge of point minimizers obtained from the previous study. The first method is the proposed PCGPwM method. The second uses the simplistic imputation approach with PCGP-kNN. The third uses the colGP method, and the last method is simply to only use data rows with complete data. GP-OM discussed in the preceding section is not usable in this setting due to the data size ($\approx$ 90,000) causing it to be computationally infeasible: we estimate

that the surrogate construction alone would take more than 30 days, before any calibration can be performed. The EMGP method is not applicable either, due to its computational instability. In the original paper by Hung et al. [80], an isotropic correlation function is used for the categorical variable, which has two levels. In the Fayans EDF model, the categorical variable has nine levels. To adopt the EMGP method, the correlation is chosen to be 1 if the categorical variable is the same, and 0 if not. However, the hyperparameter estimation for the EMGP consistently fails to converge for this application.

We construct the surrogates using simulation outputs of 500 well-spaced parameters. All parameters reside close to a local minimizer of the χ^2 loss from Bollapragada et al. [21]. The parameter space is previously scaled to a unit hypercube such that the dimensions are comparable. To recover the unscaled parameter the centroid of the unscaled hypercube and the scale for each dimension are provided in Table A.5 in Appendix A, which reproduces Table 5 of Bollapragada et al. [21].

The simulation outputs are partially observed, where approximately 10% of the 99,000 responses are missing. Figure 2.1, as seen in the introduction, shows the missing value pattern in the sampled outputs arranged by increasing number of failures in parameters. Only 141 rows have complete data.

To calibrate the Fayans EDF, the constructed surrogates are then supplied to the calibrator module in `surmise`. The chosen prior is the Beta$(2, 2)$ distribution over each dimension of the scaled parameter, representing the stability region studied in Bollapragada et al. [21]. The choice Beta$(2, 2)$ reflects the understanding that the parameters closer to the centroid of the scaled hypercube are more plausible. The prior density outside the scaled hypercube is zero, which reflects the boundary of stability region defined by the

lower and upper bounds. The prior is then

$$\pi(\boldsymbol{\theta}) \propto \prod_{l=1}^{d} \theta_l \left(1 - \theta_l\right), \boldsymbol{\theta} \in [0,1]^d,$$

where θ_l is the lth element of $\boldsymbol{\theta}$. With $\pi(\boldsymbol{\theta})$ specified, the posterior is then given by (2.12). Samples are drawn from the posterior by the Langevin Monte Carlo method [132], an MCMC method that utilizes gradient information at the current iterate. In addition, the sampling method is strengthened by incorporating parallel tempering [54, 51]. The method is implemented in the utility module of `surmise` under the name `PTLMC`. We note that the closed-form nature of our surrogate allows for easy deployment of gradient-based approaches.

To investigate the utility of the surrogate-based inference using these methods, we compare the posterior distributions. Since the considered parameter space is scaled around the centroid of the hypercube, we expect the posterior means to be close. This is verified in our computation that the posterior means estimated using all surrogate methods considered are found to be close to each other. We are then concerned with the precision of the posterior, the idea being that a more precise surrogate in this setting should lead to a narrower posterior on the parameters. This can be seen in expression (2.12), where a decrease to the surrogate covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ yields a more concentrated posterior, given a fixed $\boldsymbol{\mu}(\boldsymbol{\theta})$. The precision of the surrogate is measured by the widths of the intervals between the 5% and 95% quantiles (called the 90% width) for each parameter relative to the upper and lower bounds. Table 2.2 contains the 90% widths relatively scaled from all surrogates. The credible intervals from all the surrogates shrink compared with the Beta$(2, 2)$ 90% relative width, 0.730. PCGPwM results in the smallest intervals

all but two of the model parameters. This constraining of plausible parameter region is attributed to the improved surrogate offering more precise predictions of simulation responses. The benefit for this case study is that the analysis using the proposed surrogate method provides the uncertainty estimates for the parameter of the Fayans EDF model in contrast to previous studies. We find that there are considerable differences in the resulting interval widths, with some parameters (e.g., f_{ex}^{ξ} and h_{+}^{ξ}) being estimated more precisely, and some parameters (e.g., $E/A, K$, and L) having smaller precision improvements. To more concretely understand the benefit, if we only use the complete output data to build our surrogate, the resulting posterior intervals would have been 6%–58% wider.

EDF Parameters	PCGPwM	PCGP-kNN	colGP	Complete Data	Prior, Beta$(2,2)$
ρ_{eq}	**0.355**	0.500	0.546	0.491	0.730
E/A	**0.345**	0.463	0.389	0.607	0.730
K	0.455	0.493	**0.361**	0.643	0.730
J	**0.303**	0.557	0.535	0.497	0.730
L	**0.437**	0.484	0.393	0.576	0.730
h_{2-}^{v}	**0.370**	0.462	0.401	0.587	0.730
a_{+}^{s}	**0.421**	0.569	0.654	0.450	0.730
h_{∇}^{s}	**0.337**	0.405	0.351	0.637	0.730
κ	**0.339**	0.479	0.461	0.614	0.730
κ'	**0.198**	0.319	0.273	0.421	0.730
f_{ex}^{ξ}	**0.125**	0.345	0.246	0.300	0.730
h_{∇}^{ξ}	0.386	0.530	**0.347**	0.536	0.730
h_{+}^{ξ}	**0.128**	0.367	0.215	0.254	0.730

Table 2.2. Posterior 90% widths relative to their respective ranges for the 13-dimensional parameter using different emulation techniques. The narrowest 90% interval for each parameter is shown in boldface.

2.6. Concluding remarks

This chapter details a new surrogate construction method developed to handle missing data. The construction relies on an imputation of the missing data and a covariance

adjustment to account for the added uncertainty due to imputation. This method is efficient and adds minimal burden on computations. The surrogate construction is effective in ignoring entirely missing data in a multivariate output context. Furthermore, it retains the efficiency of modern approaches to building surrogates of high-dimensional output data.

It is expected that as nascent models are used in large computing environments, partially observed output data will become more prevalent. For example, Lin et al. [103] conduct their inference by constructing a missingness classifier in addition to a surrogate using available simulation data. When missingness is important to inference, the method proposed in this chapter could be used in to help with the surrogate to prevent tractability issues.

CHAPTER 3

MODELING HIGH-DIMENSIONAL STOCHASTIC SIMULATION OUTPUTS WITH HETEROGENEOUS ERROR

Computer models, or simulators, are a major tool in studying complex phenomena across scientific disciplines. Broadly speaking, any code that takes a set of user inputs and returns some output can be called a computer model. Such action of assigning inputs and retrieving outputs are termed as running the simulator, or performing a simulation run. Recall that a simulation run refers to the process of providing an input, evaluating the simulation, and retrieving an output. This chapter focuses on computer models with one or more of the following characteristics:

(i) The simulation model produces multivariate outputs. Multiple quantities observed from simulating a system can be of importance. For examples, in epidemiology, a COVID-19 simulator returns counts of hospitalizations, patients in the intensive care unit, and hospital admissions for a time period [145]; in modeling storm surge, a set of partial differential equations is solved to return both the water elevation and velocity for a number of locations [127]; and in simulating heavy-ion collisions, multiple measurements are returned for one simulation [42].

(ii) The simulation model is stochastic, meaning that for the same input, the simulator can return different outputs when being run multiple times. For example, an agent-based model simulates individual actions for a large number of agents in the system, where the

actions are governed by user-defined rules [108]. An agent-based model is particularly apt to represent stochastic dynamics in transportation, epidemiology, and social sciences [167, 141, 166]. Due to these characteristics, the analysis of stochastic computer models presents significant challenges and opportunities, requiring development of sophisticated statistical tools [11].

In addition, each simulation run is time-consuming. Advances in computer models generally increase its complexity to model additional features or to refine its fidelity. Improving a computer model to better represent a certain phenomenon means that the required run time also increases. For example, to obtain 2000 outputs from a state-of-the-art heavy-ion collision simulation in nuclear physics can take approximately 1000 core-hours [42].

Using a statistical surrogate, or emulator, in place of a simulator is particularly attractive when running the simulator is time-consuming. The key idea is to use a small set of outputs from the simulators to predict values at unseen inputs. One usage of an emulator is to substitute for the simulator in Bayesian calibration of computer models [87, 153, 154, 126], where calibration refers to finding the set of inputs that aligns simulation outputs with some experimental data from the physical system. In the case where the simulator is moderately time-consuming, procedures like Markov chain Monte Carlo sampling are unusable because the simulator has to be run hundreds of thousands of times. Substituting an emulator for a simulator allows for efficient calibration of computer models. Gaussian processes (GPs) have been a popular emulator choice as early as Sacks et al. [135], Currin et al. [37] and a plethora of research has emerged since then. In the deterministic output case, GPs are effective due to their ability to interpolate

collected outputs and to naturally provide uncertainty quantification over predictions. In the stochastic output case, GPs remain an effective choice provided that modifications are made to address output variability [88]. We refer to Santner et al. [137], Gramacy [57], Baker et al. [11] for a selection of excellent expositions. We highlight two particular directions of GP development relevant to our proposed method:

(i) High-dimensional or multivariate outputs. Constructing a multivariate GP emulator while taking into account between-output correlations can quickly become intractable as the number of training inputs increases [48]. We explore the computational burden of a full multivariate GP in Section 3.2.2. It is common to impose some structure onto the covariance matrix of the GP such that GP constructions are still tractable while considering output correlations, making use of a separable covariance [36, 80], convolution [73, 5, 32], or basis representation [14, 74, 76, 65, 25, 42, 106, 28].

(ii) Stochastic outputs. Many of the GP methods are developed to handle deterministic computer models, where the simulator returns the same output every time given the same input. The literature on the handling of stochastic outputs is relatively scarce. For a univariate GP, one typically adds a "nugget" on the diagonal of the GP covariance matrix [130, 137, 57]. The magnitude of the nugget can be estimated using simulation outputs. In the case where the outputs are multivariate and stochastic, the treatment is less clear. Modeling the full error covariance is practically intractable, as we explore in Section 3.2.2. To retain computational efficiency, one approach is to assume a homogeneous error variance across all outputs [65, 25], another is to model a general error structure but use an approximation on the GP likelihood [5, 32]. However, the assumption in the former approach may be unrealistic when the behavior across output locations differs, while the

latter may not fully represent the simulation model, especially its uncertainty. One related approach to maintain tractability is to partition the input space and build multiple GPs [61, 33], but these methods do not directly tackle the issue of multivariate stochastic outputs. We also note that this stochasticity differs from the notion of heteroskedasticity, which refers to the change of variances over the input space. Such issues may be tackled via the use of stochastic kriging [7], and heteroskedastic GP [19].

In this chapter we propose a novel GP method, termed latent component Gaussian process (LCGP), that accounts for heterogeneous error in emulating multivariate stochastic outputs. The learning of uncertainty of the latent components, formally introduced in Section 3.2.2.4, forms the foundation of this method. LCGP utilizes a basis representation model in emulating multivariate output, with proper linear transformation of the output. LCGP provides exact inference in its prediction without incurring additional computational costs to a basis representation GP.

The remainder of the chapter is structured as follows: Section 3.1 introduces the settings and preliminary results for exposition. Section 3.2 provides a background on Gaussian processes with stochastic univariate and multivariate outputs. Section 3.3 details the method of LCGP, its training, and its prediction. Section 3.4 includes an illustrative example for the treatment of heterogeneous error structures and latent component uncertainty. Section 3.5 provides the implementation details and a set of numerical experiments to highlight the performance of LCGP. Section 3.6 applies LCGP to emulating a set of simulation outputs from a viscous anisotropic hydrodynamics model used in heavy-ion collision simulations. Section 3.7 summarizes and closes the chapter.

3.1. Settings and preliminaries

We consider a simulation model $\boldsymbol{y} : \mathcal{X} \to \mathbb{R}^p$ that produces a multivariate output for some design input $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$. The simulation outputs at a set of n inputs $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ are arranged into $\boldsymbol{Y} = (\boldsymbol{y}(\boldsymbol{x}_1)^\mathsf{T}, \ldots, \boldsymbol{y}(\boldsymbol{x}_n)^\mathsf{T})^\mathsf{T}$, a matrix of size $n \times p$, such that each column contains all outputs at an output location and each row contains a full output from an input. Depending on the purpose of collecting the simulation outputs, different strategies of choosing the design inputs are available. The choice of a certain collection of inputs is termed an experimental design, or simply a design [137, ch.5]. A type of design is the space-filling design, including examples like Latin hypercube sampling [114] and maximin distance design [116]. The space-filling design strategy seeks to spread out the design inputs evenly throughout the input space, with a hope to observe the simulation model over the entire input space.

For an invertible matrix $\boldsymbol{A}$ of size $n \times n$, we denote its inverse by $\boldsymbol{A}^{-1}$ and its determinant by $|\boldsymbol{A}|$.

The column vectors of a general matrix $\boldsymbol{A}$ of size $n \times p$ are denoted by $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_p$. The vectorization of $\boldsymbol{A}$ is defined as the stacking of columns, i.e., $\mathrm{vec}(\boldsymbol{A}) = (\boldsymbol{A}_1^\mathsf{T}, \ldots, \boldsymbol{A}_p^\mathsf{T})^\mathsf{T}$.

The Woodbury identites are used widely in inverting a matrix with a specific structure [70, ch.18]. The general version is stated here as the identities are critical for our exposition. For nonsingular matrices $\boldsymbol{R} \in \mathbb{R}^{n\times n}, \boldsymbol{T} \in \mathbb{R}^{p\times p}$, and matrices $\boldsymbol{S} \in \mathbb{R}^{n\times p}, \boldsymbol{U} \in \mathbb{R}^{p\times n}$, the Woodbury identities are stated as

$$(\boldsymbol{R} + \boldsymbol{STU})^{-1} = \boldsymbol{R}^{-1} - \boldsymbol{R}^{-1}\boldsymbol{S}(\boldsymbol{T}^{-1} + \boldsymbol{UR}^{-1}\boldsymbol{S})^{-1}\boldsymbol{UR}^{-1}, \tag{3.1}$$

$$|\boldsymbol{R} + \boldsymbol{STU}| = |\boldsymbol{R}|\ |\boldsymbol{T}|\ |\boldsymbol{T}^{-1} + \boldsymbol{UR}^{-1}\boldsymbol{S}|. \tag{3.2}$$

3.2. Gaussian processes background

This section provides the general background of univariate and multivariate GPs for emulating stochastic simulation outputs. Three common models of multivariate GPs are reviewed: separable covariance, convolution process, and basis representation. Our proposed method adopts the basis representation model for its use of dimension-reduction technique and the expressiveness of its formulation.

3.2.1. Univariate GP. A univariate GP is the building block for any multivariate GP. A GP is a random process where any finite collection of its elements follows a multivariate Gaussian distribution [130]. Consider a vector of scalar responses $(g(\boldsymbol{x}_1), \ldots, g(\boldsymbol{x}_n))^\mathsf{T}$ from the function $g : \mathcal{X} \to \mathbb{R}$, evaluated at the set of inputs $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

In a typical stochastic simulation setting, the observed responses are assumed to be corrupted by Gaussian noise with variance σ^2. Therefore, instead of directly observing $g(\boldsymbol{x}_i)$, we observe some noisy output $y(\boldsymbol{x}_i) = g(\boldsymbol{x}_i) + \varepsilon_i$, where $\varepsilon_i \overset{\text{iid}}{\sim} \mathsf{N}(0, \sigma^2)$. The observations are collected in the vector $\boldsymbol{y} = (y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_n))^\mathsf{T}$. In such case, we consider a simulation run as collecting the noisy output once. Multiple simulation runs at the same input will produce multiple noisy outputs.

Modeling $g(\cdot)$ with a GP means that $(g(\boldsymbol{x}_i)_{i=1}^n)^\mathsf{T} \sim \mathsf{N}(\boldsymbol{0}, \boldsymbol{C})$, where $\boldsymbol{C}$ is an $n \times n$ covariance matrix with entries $c(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}) = \text{cov}(g(\boldsymbol{x}_i), g(\boldsymbol{x}_j))$ for any pair of inputs $\boldsymbol{x}_i, \boldsymbol{x}_j, i, j = 1, \ldots, n$, with a covariance function $c(\cdot, \cdot)$ parameterized by $\boldsymbol{\theta}$. We refer to $\boldsymbol{\theta}$ as the set of GP covariance hyperparameters. The mean of the GP is chosen to be zero for brevity in exposition, but the results in this chapter extend to non-zero mean function. We note that other choices of modeling the mean are available, e.g., through universal

kriging (or "blind kriging") [85, 9]. We opt to focus on modeling of the covariance matrix in this chapter.

The predictive distribution for a new observed output, $y(\boldsymbol{x})$ given observation $\boldsymbol{y}$, evaluated at a new input $\boldsymbol{x}$ then follows a Gaussian distribution with mean and variance [130, ch.2]

$$\mathbb{E}[y(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}, \sigma^2] = \boldsymbol{c}(\boldsymbol{x})^\mathsf{T}(\boldsymbol{C} + \sigma^2 \boldsymbol{I}_n)^{-1}\boldsymbol{y}, \text{ and} \tag{3.3}$$

$$\mathbb{V}[y(\boldsymbol{x}) \mid \boldsymbol{X}; \boldsymbol{\theta}, \sigma^2] = c(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{c}(\boldsymbol{x})^\mathsf{T}(\boldsymbol{C} + \sigma^2 \boldsymbol{I}_n)^{-1}\boldsymbol{c}(\boldsymbol{x}) + \sigma^2, \tag{3.4}$$

where $\boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{c}(\boldsymbol{x}; \boldsymbol{\theta}) = (c(\boldsymbol{x}_1, \boldsymbol{x}; \boldsymbol{\theta}), \ldots, c(\boldsymbol{x}_n, \boldsymbol{x}; \boldsymbol{\theta}))^\mathsf{T}$ is the vector of covariances between the training inputs and the new input, and $\boldsymbol{I}_n$ is the $n \times n$ identity matrix. When the responses are assumed to be noiseless, by setting $\sigma^2 = 0$, the predictive equations Equations (3.3) and (3.4) resemble the univariate GP predictions in Equation (2.2).

Fitting this univariate GP is through maximizing the log-likelihood function with respect to the hyperparameters $\boldsymbol{\theta}$ and σ^2,

$$\ell(\boldsymbol{\theta}, \sigma^2; \boldsymbol{X}, \boldsymbol{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{C} + \sigma^2 \boldsymbol{I}_n| - \frac{1}{2}\boldsymbol{y}^\mathsf{T}(\boldsymbol{C} + \sigma^2 \boldsymbol{I}_n)^{-1}\boldsymbol{y}. \tag{3.5}$$

Evaluating Equation (3.5) involves computing the matrix determinant and inverse of the $n \times n$ covariance matrix, costing $\mathcal{O}(n^3)$. While solving the problem of the reliance on the number of data points is not the focus of this chapter, approximation strategies are available to alleviate the cubic cost. Most methods introduce sparsity in the covariance matrix or its inverse, including the use of inducing points [142, 12], partitioning of input space

[63], local structures [58, 33], and Vecchia approximations [156, 86]. The maximization of Equation (3.5) can be performed using optimization algorithms such as L-BFGS [120].

3.2.2. Multivariate output GP. Now consider the multivariate output case, where the simulation model produces a p-variate output at any input $\boldsymbol{x} \in \mathcal{X}$. Each dimension of the output is called a location. Consider that the vector-valued responses from the function $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^p$, collected across the set of inputs $\boldsymbol{X}$, are arranged as an $n \times p$ matrix $\boldsymbol{F} = (\boldsymbol{f}(\boldsymbol{x}_1)^\mathsf{T}, \ldots, \boldsymbol{f}(\boldsymbol{x}_n)^\mathsf{T})^\mathsf{T}$. Analogous to the univariate case, the observations are now assumed to be corrupted by multivariate Gaussian noise. Therefore, instead of directly observing $\boldsymbol{F}$, we observe a corrupted version $\boldsymbol{Y} = (\boldsymbol{y}(\boldsymbol{x}_1)^\mathsf{T}, \ldots, \boldsymbol{y}(\boldsymbol{x}_n)^\mathsf{T})^\mathsf{T}$, with each row being $\boldsymbol{y}(\boldsymbol{x}_i) = \boldsymbol{f}(\boldsymbol{x}_i) + \boldsymbol{\varepsilon}_i$, where $\boldsymbol{\varepsilon}_i \overset{\text{iid}}{\sim} \mathsf{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a $p \times p$ (positive definite) covariance matrix.

Next, we introduce the full formulation for a multivariate output GP, followed by three common strategies for alleviating the computational burden in directly using the full formulation.

3.2.2.1. *Full formulation for multivariate output GP.* The multivariate GP models correlations across the locations with a covariance matrix capturing the within-location and between-location covariances. Denote the cross-covariance between the column vectors of $\boldsymbol{F}$ as $\boldsymbol{C}_{m,m'} = \operatorname{cov}(\boldsymbol{F}_m, \boldsymbol{F}_{m'}) = (c(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}_{m,m'})_{i,j=1}^n), m, m' = 1, \ldots, p$. Then, to model $\boldsymbol{f}(\cdot)$ with a multivariate GP means that $\operatorname{vec}(\boldsymbol{F}) \sim \mathsf{N}(\boldsymbol{0}, \mathsf{K})$, where K is the full $np \times np$ covariance matrix

$$\mathsf{K} = \begin{pmatrix} \boldsymbol{C}_{1,1} & \ldots & \boldsymbol{C}_{1,p} \\ \vdots & \ddots & \vdots \\ \boldsymbol{C}_{1,p}^\mathsf{T} & \ldots & \boldsymbol{C}_{p,p} \end{pmatrix}. \tag{3.6}$$

At a new input $\boldsymbol{x}$, the predictive distribution for $\boldsymbol{y}(\boldsymbol{x})$ given observation $\boldsymbol{Y}$ follows a multivariate Gaussian distribution with mean and covariance

$$\mathbb{E}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\Theta}, \boldsymbol{\Sigma}] = \mathbf{k}(\boldsymbol{X}, \boldsymbol{x})^{\mathsf{T}}(\mathbf{K} + \boldsymbol{\Sigma})^{-1}\mathrm{vec}(\boldsymbol{Y}), \text{ and} \tag{3.7}$$

$$\mathrm{cov}(\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}; \boldsymbol{\Theta}, \boldsymbol{\Sigma}) = \mathbf{k}(\boldsymbol{x}, \boldsymbol{x}) - \mathbf{k}(\boldsymbol{X}, \boldsymbol{x})^{\mathsf{T}}(\mathbf{K} + \boldsymbol{\Sigma})^{-1}\mathbf{k}(\boldsymbol{X}, \boldsymbol{x}) + \boldsymbol{\Sigma}, \tag{3.8}$$

where $\mathbf{k}(\boldsymbol{X}, \boldsymbol{x})$ is the $np \times p$ cross-covariance matrix between $\boldsymbol{F}_m$ and $\boldsymbol{f}(\boldsymbol{x}), m = 1, \ldots, p$, and $\mathbf{k}(\boldsymbol{x}, \boldsymbol{x})$ is the $p \times p$ covariance matrix of $\boldsymbol{f}(\boldsymbol{x})$. Let $\boldsymbol{\Theta}$ collect all covariance hyperparameters, and $\boldsymbol{\Sigma} = \Sigma \otimes \boldsymbol{I}_n$. The log-likelihood function for this multivariate GP is given as

$$\ell(\boldsymbol{\Theta}, \boldsymbol{\Sigma}; \boldsymbol{X}, \boldsymbol{Y}) = -\frac{np}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{K} + \boldsymbol{\Sigma}| - \frac{1}{2}\mathrm{vec}(\boldsymbol{Y})^{\mathsf{T}}(\mathbf{K} + \boldsymbol{\Sigma})^{-1}\mathrm{vec}(\boldsymbol{Y}). \tag{3.9}$$

Evaluating this log-likelihood function, similar to that of the univariate GP (Equation (3.5)), costs $\mathcal{O}(n^3p^3)$, arising from the inversion of the full covariance matrix $(\mathbf{K} + \boldsymbol{\Sigma})$ of size $np \times np$. This evaluation cost prohibits the direct use of a multivariate GP in practice, due to its computational burden.

Remedies have been proposed to address this computational issue. The idea is to impose additional structure on the form of the covariance matrix $\mathbf{K}$, such as separable covariance [17, 133, 36, 18, 80], convolution processes [73, 32], and basis representation [14, 74, 76, 65]. In each of these methods, the primary aim is to avoid inverting the full covariance matrix while maintaining model fidelity.

However, in the stochastic case, the computational burden is not necessarily alleviated with the imposition of structure, since the matrix in consideration is $\mathbf{K} + \boldsymbol{\Sigma}$, instead of

$\mathbf{K}$ itself. Unless $\mathbf{\Sigma}$ also exhibits certain structure, inverting $\mathbf{K} + \mathbf{\Sigma}$ does not benefit from the structure in $\mathbf{K}$. Directly inverting $\mathbf{K} + \mathbf{\Sigma}$ is numerically intractable, without resorting to approximations, such as through constructing a variational objective [148, 22, 5, 32], and/or using a covariance approximation with the Nyström method using inducing points [142, 12]. Therefore, often in existing literature, either $\mathbf{\Sigma} = \mathbf{0}$ for deterministic simulation models, or $\mathbf{\Sigma}$ takes the form of a homogeneous error variance, i.e., $\mathbf{\Sigma} = \sigma^2 \boldsymbol{I}_p \otimes \boldsymbol{I}_n$, such that the benefit from the structure of $\mathbf{K}$ can be realized through applying various matrix tricks, such as the Woodbury identities in Equations (3.1–3.2).

In the case where replications are available from repeated simulation runs at the same input, it may be advantageous to directly estimate the error covariance. However, the number of total training inputs quickly poses a computational challenge to GP inference. In the univariate case, Ankenman et al. [7] provides a plug-in approach (stochastic kriging) by using the sample means and variances at each unique input. As a result, the covariance matrix in the corresponding GP predictions is of size $\tilde{n} \times \tilde{n}$, where $\tilde{n} < n$ is the number of unique inputs, potentially much smaller than the number of total training inputs. This idea of employing sample estimators may be extended to the multivariate GP case. However, a large number of replications may be required for the estimators to be reliable, which can be impractical. In this chapter, we consider the case where the inputs are unique and we estimate the covariance matrix $\mathbf{\Sigma}$ as part of the method.

3.2.2.2. *Separable covariance.* The separable covariance model assumes that the covariance, $\mathbf{K} = \boldsymbol{C} \otimes \boldsymbol{S}$, is a Kronecker product of two positive definite matrices, with an $n \times n$ matrix $\boldsymbol{C} = (c(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta})_{i,j=1}^n)$ containing entries of covariance between inputs $\boldsymbol{X}$ and a $p \times p$ matrix $\boldsymbol{S}$ being the between-location covariance matrix. As long as $\boldsymbol{C}$ and $\boldsymbol{S}$ are

positive definite, $\mathbf{K}$ will also be positive definite. The separable form leads to an efficient computation for the matrix inverse and determinant of $\mathbf{K}$, reducing the computation from $\mathcal{O}(n^3p^3)$ to $\mathcal{O}(n^3 + p^3)$. By employing the separable covariance structure, the predictive equations Equations (3.7) and (3.8) become

$$\mathbb{E}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\Theta}, \boldsymbol{\Sigma}] = \mathbf{k}(\boldsymbol{x})^\mathsf{T}(\boldsymbol{I}_{np} + (\boldsymbol{C}^{-1} \otimes \boldsymbol{S}^{-1})\boldsymbol{\Sigma})^{-1}(\boldsymbol{C}^{-1} \otimes \boldsymbol{S}^{-1})\mathrm{vec}(\boldsymbol{Y}), \text{ and}$$

$$\mathrm{cov}(\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}; \boldsymbol{\Theta}, \boldsymbol{\Sigma}) = \mathbf{k}(\boldsymbol{x}, \boldsymbol{x}) - \mathbf{k}(\boldsymbol{x})^\mathsf{T}(\boldsymbol{I}_{np} + (\boldsymbol{C}^{-1} \otimes \boldsymbol{S}^{-1})\boldsymbol{\Sigma})^{-1}(\boldsymbol{C}^{-1} \otimes \boldsymbol{S}^{-1})\mathbf{k}(\boldsymbol{x}) + \boldsymbol{\Sigma},$$

and correspondingly, the log-likelihood in Equation (3.9) becomes

$$\begin{aligned}\ell(\boldsymbol{\theta}, \boldsymbol{\Sigma}; \boldsymbol{X}, \boldsymbol{Y}) = &- \frac{np}{2}\log(2\pi) - \frac{1}{2}\left(p\log|\boldsymbol{C}| + n\log|\boldsymbol{S}| + \log|\boldsymbol{I}_{np} + (\boldsymbol{C}^{-1} \otimes \boldsymbol{S}^{-1})\boldsymbol{\Sigma}|\right) \\ &- \frac{1}{2}\mathrm{vec}(\boldsymbol{Y})^\mathsf{T}(\boldsymbol{I}_{np} + (\boldsymbol{C}^{-1} \otimes \boldsymbol{S}^{-1})\boldsymbol{\Sigma})^{-1}(\boldsymbol{C}^{-1} \otimes \boldsymbol{S}^{-1})\mathrm{vec}(\boldsymbol{Y}).\end{aligned}$$

Notice that although the separable covariance allows for efficient computation for $\mathbf{K}$, applying the separable form for the full covariance $\mathbf{K} + \boldsymbol{\Sigma}$ does not remove the need to invert an $np \times np$ matrix, which requires the same computational effort as the full model in Equation (3.9), i.e., $\mathcal{O}(n^3p^3)$. In addition, the separability assumption is restrictive in the sense that only one hyperparameter $\boldsymbol{\theta}$ controls the spatial correlations among inputs for every location. In cases where the spatial correlations vary across locations, the empirical performance under this model suffers [48].

3.2.2.3. *Convolution process.* A convolved GP is an example of non-separable covariance model, constructed by the convolution of a selection of smoothing kernel $s_m(\cdot)$ with a zero-mean GP $g(\cdot)$ with covariance function $c(\cdot, \cdot; \boldsymbol{\theta})$ [73, 5, 48]. An example choice of the smoothing kernel is a radial Gaussian kernel [73], The elements of $\boldsymbol{f}(\cdot)$ are then constructed

as

$$f_m(\boldsymbol{x}) = \int_{\mathcal{X}} s_m(\boldsymbol{u} - \boldsymbol{x}) g(\boldsymbol{u}) \mathrm{d}\boldsymbol{u}, \quad m = 1, \ldots, p. \tag{3.10}$$

Such construction results in the covariance function between elements of $\boldsymbol{f}(\cdot)$ to be

$$\mathrm{cov}(f_m(\boldsymbol{x}_i), f_{m'}(\boldsymbol{x}_j)) = \int_{\mathcal{X}} \int_{\mathcal{X}} s_m(\boldsymbol{u} - \boldsymbol{x}_i) s_{m'}(\boldsymbol{u}' - \boldsymbol{x}_j) c(\boldsymbol{u}, \boldsymbol{u}'; \boldsymbol{\theta}) \mathrm{d}\boldsymbol{u} \mathrm{d}\boldsymbol{u}'.$$

Therefore, the block entries of the covariance matrix $\mathbf{K}$ in Equation (3.6) are

$$\boldsymbol{C}_{m,m'} = \mathrm{cov}(f_m(\boldsymbol{x}_i), f_{m'}(\boldsymbol{x}_j))_{i,j=1}^{n}. \tag{3.11}$$

With $\mathbf{K}$ defined in Equation (3.11), the predictive equations and log-likelihood are given by Equations (3.7–3.9). The use of a common GP $g(\cdot)$ in the convolution has been proposed by Ver Hoef and Barry [158] and extended to include multiple GPs in the work of Álvarez et al. [5]. The strength of such formulation is in the independence of the column vectors $\boldsymbol{F}_m, m = 1, \ldots, p$, given a fixed g [32]. The idea to using a convolved multivariate output GP is in approximating $g(\boldsymbol{u})$ with the conditional mean given a small set of observations. Consider a set of "inducing" inputs $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r)$ where the integrals in Equations (3.10) and (3.11) are approximated by observing $\boldsymbol{g} = (g(\boldsymbol{u}_1), \ldots, g(\boldsymbol{u}_r))^{\mathsf{T}}$. As a result, the predictive equations in Equations (3.7) and (3.8) become

$$\mathbb{E}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\Theta}, \boldsymbol{\Sigma}] = \mathbf{k}(\boldsymbol{x})^{\mathsf{T}} (\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}}^{\mathsf{T}} \boldsymbol{C}_{\boldsymbol{g},\boldsymbol{g}}^{-1} \boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}} + \boldsymbol{\Delta} + \boldsymbol{\Sigma})^{-1} \mathrm{vec}(\boldsymbol{Y}), \text{ and}$$

$$\mathrm{cov}(\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}; \boldsymbol{\Theta}, \boldsymbol{\Sigma}) = \mathbf{k}(\boldsymbol{x}, \boldsymbol{x}) - \mathbf{k}(\boldsymbol{x})^{\mathsf{T}} (\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}}^{\mathsf{T}} \boldsymbol{C}_{\boldsymbol{g},\boldsymbol{g}}^{-1} \boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}} + \boldsymbol{\Delta} + \boldsymbol{\Sigma})^{-1} \mathbf{k}(\boldsymbol{x}) + \boldsymbol{\Sigma}$$

respectively, where $\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{g}} = (c(\boldsymbol{u}_i, \boldsymbol{u}_j; \boldsymbol{\theta})_{i,j=1}^{r}), \boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}_m} = (c(\boldsymbol{u}_i, \boldsymbol{x}_j; \boldsymbol{\theta}_m)_{i=1}^{r})_{j=1}^{n}$. The cross-covariance matrix $\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}} = \left(\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}_1}, \ldots, \boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}_p}\right)$ contains the covariance matrices between $\boldsymbol{g}$

and vec($\boldsymbol{F}$), and $\boldsymbol{\Delta}$ is a correction factor to the approximation. For example, one choice of $\boldsymbol{\Delta}$ is $\boldsymbol{\Delta} = \text{bdiag}\left((\boldsymbol{\Delta}_m)_{m=1}^p\right)$ and $\boldsymbol{\Delta}_m = \boldsymbol{C}_{m,m} - \boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}_m}^\mathsf{T}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{g}}^{-1}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}_m}$. The notation bdiag$(\cdot)$ refers to a block diagonal matrix with the entries placed on the main diagonal. This choice is referred to as the partially independent training conditional [129], such that the approximation is exact when the inducing and the original inputs align, i.e., $\boldsymbol{U} = \boldsymbol{X}$. See Álvarez et al. [5] for a summary of the impact of choices of this correction factor. The log-likelihood in Equation (3.9) correspondingly becomes

$$\begin{aligned}\ell(\boldsymbol{\Theta}, \boldsymbol{\Sigma}; \boldsymbol{X}, \boldsymbol{Y}) = &- \frac{np}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}}^\mathsf{T}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{g}}^{-1}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}} + \boldsymbol{\Delta} + \boldsymbol{\Sigma}| \\ &- \frac{1}{2}\text{vec}(\boldsymbol{Y})^\mathsf{T}(\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}}^\mathsf{T}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{g}}^{-1}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}} + \boldsymbol{\Delta} + \boldsymbol{\Sigma})^{-1}\text{vec}(\boldsymbol{Y}).\end{aligned}$$

The appeal of replacing the covariance matrix K by $\left(\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}}^\mathsf{T}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{g}}^{-1}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}} + \boldsymbol{\Delta}\right)$ is in the drastically reduced computational cost of the latter, facilitated by the Woodbury matrix identities in Equations (3.1–3.2). The computational benefit in the convolved GP approach relies on picking $r \ll n$ such that only a much smaller matrix is required for inversion. The application of Woodbury identities results in

$$\begin{aligned}&(\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}}^\mathsf{T}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{g}}^{-1}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}} + \boldsymbol{\Delta} + \boldsymbol{\Sigma})^{-1} \\ &\quad = (\boldsymbol{\Delta} + \boldsymbol{\Sigma})^{-1} - (\boldsymbol{\Delta} + \boldsymbol{\Sigma})^{-1}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}}^\mathsf{T}\left(\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{g}} + \boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}}(\boldsymbol{\Delta} + \boldsymbol{\Sigma})^{-1}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}}^\mathsf{T}\right)^{-1}\boldsymbol{C}_{\boldsymbol{g},\boldsymbol{F}}(\boldsymbol{\Delta} + \boldsymbol{\Sigma})^{-1},\end{aligned}$$

where the typical speedup results from first having a trivially invertible $\boldsymbol{\Delta} + \boldsymbol{\Sigma}$, e.g., a diagonal or block diagonal matrix, and second having a much smaller matrix to invert, e.g., in this case the middle matrix is of size $r \times r$, r being the number of inducing points chosen. Notice this speedup is not guaranteed. While $\boldsymbol{\Delta}$ can always be chosen to be

diagonal or block diagonal, $\boldsymbol{\Sigma}$ is not diagonal when a general error covariance matrix is considered. As a result, the speedup advantage is lost since the inverse $(\boldsymbol{\Delta} + \boldsymbol{\Sigma})^{-1}$ again requires a computation of $\mathcal{O}(n^3p^3)$. If the error covariance $\boldsymbol{\Sigma}$ is a diagonal matrix, then applying the Woodbury identities is beneficial to reduce the computational burden.

The choice of inducing inputs, also called pseudo-inputs [142, 152, 33], is non-trivial. Approaches to select inducing points vary greatly, from choosing a subset of the original inputs [142], to applying a greedy algorithm [152, 33, 117], to optimization [27]. The posterior predictive distribution is obtained by marginalizing out $\boldsymbol{U}$, which requires the evaluation of the log-likelihood. However, evaluation of the log-likelihood function is numerically intractable. A common remedy is to resort to variational inference, estimating the posterior distribution by maximizing an evidence lower bound (ELBO) of the likelihood function [20, 32], along the line of research Álvarez et al. [5] termed as "stochastic variational GP" and its subsequent variants. A systematic review can be found in Liu et al. [104].

3.2.2.4. *Basis representation.* The basis representation model, also called linear model of coregionalization, is another example of a non-separable covariance model [50, 48, 25, 65]. The basis representation model is also categorized under the use of "sum of separable covariance kernels" (see e.g., Álvarez et al. [5]). Basis representation assumes that the simulation output can be modeled as

$$\boldsymbol{y}(\boldsymbol{x}_i) = \boldsymbol{f}(\boldsymbol{x}_i) + \boldsymbol{\varepsilon}_i, \tag{3.12}$$

$$\boldsymbol{f}(\boldsymbol{x}_i) = \boldsymbol{\Psi}\boldsymbol{g}(\boldsymbol{x}_i), \tag{3.13}$$

$$g_k(\cdot) \sim \mathsf{GP}(0, c(\cdot, \cdot; \boldsymbol{\theta}_k)), \quad k = 1, \ldots, q, \tag{3.14}$$

for some error vector $\boldsymbol{\varepsilon}$, a transformation basis matrix $\boldsymbol{\Psi} = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_q) \in \mathbb{R}^{p \times q}, q \leq p$, and latent components $\boldsymbol{g}(\cdot) = (g_1(\cdot), \ldots, g_q(\cdot))^{\mathsf{T}}$, with each $g_k(\cdot), k = 1, \ldots, q$, being an independent GP with zero mean and covariance function $c(\cdot, \cdot; \boldsymbol{\theta}_k)$. The latent components have also been termed as latent processes [25], building-block processes [48], and factor processes [65]. The correlation between outputs are modeled by the choice of the basis matrix. Depending on the choice, this model can represent a wide range of methods, including principal component analysis (PCA) [74, 76, 28], functional PCA [107], generalized probabilistic PCA (GPPCA) [65], orthogonal instantaneous linear mixing model (OILMM) [25], wavelet decomposition [14], and Fourier features [72].

Define $\boldsymbol{G} = (\boldsymbol{g}(\boldsymbol{x}_1)^{\mathsf{T}}, \ldots, \boldsymbol{g}(\boldsymbol{x}_n)^{\mathsf{T}})^{\mathsf{T}}$ to be the unknown latent components evaluated at the original inputs $\boldsymbol{X}$. Then, the covariance has the form

$$\mathsf{K} = (\boldsymbol{\Psi} \otimes \boldsymbol{I}_n)\mathsf{C}(\boldsymbol{\Psi}^{\mathsf{T}} \otimes \boldsymbol{I}_n) = \sum_{k=1}^{q} \boldsymbol{\psi}_k \boldsymbol{\psi}_k^{\mathsf{T}} \otimes \boldsymbol{C}_k,$$

where $\mathsf{C} = \operatorname{bdiag}\left((\boldsymbol{C}_k)_{k=1}^{q}\right)$, and each block being $\boldsymbol{C}_k = \left(c(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}_k)_{i,j=1}^{n}\right)$, the covariance matrix of $\boldsymbol{G}_k$. The GPPCA method by Gu and Shen [65] explored the basis representation in Equations (3.12–3.14) with an orthonormal basis (i.e., $\boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{\Psi} = \boldsymbol{I}_p$) and a homogeneous error variance (i.e., $\boldsymbol{\varepsilon}_i \sim \mathsf{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_p)$.) The predictive equations presented in Gu and Shen [65] are

$$\mathbb{E}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\Theta}, \sigma^2] = \boldsymbol{\Psi}\boldsymbol{\mu}_{\boldsymbol{g}}(\boldsymbol{x}), \tag{3.15}$$

$$\boldsymbol{\mu}_{\boldsymbol{g}}(\boldsymbol{x}) = (\mu_1(\boldsymbol{x}), \ldots, \mu_q(\boldsymbol{x}))^{\mathsf{T}}, \text{ with}$$

$$\mu_k(\boldsymbol{x}) = \boldsymbol{c}_k(\boldsymbol{x})^{\mathsf{T}}(\boldsymbol{C}_k + \sigma^2 \boldsymbol{I}_n)^{-1} \boldsymbol{Y} \boldsymbol{\psi}_k,$$

and

$$\mathbb{V}(\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}; \boldsymbol{\Theta}, \sigma^2) = \boldsymbol{\Psi}\boldsymbol{\Sigma}_{\boldsymbol{g}}(\boldsymbol{x})\boldsymbol{\Psi}^{\mathsf{T}} + \sigma^2 \boldsymbol{I}_p, \tag{3.16}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{g}}(\boldsymbol{x}) = \operatorname{diag}(\sigma_1^2(\boldsymbol{x}), \ldots, \sigma_q^2(\boldsymbol{x})), \text{ with}$$

$$\sigma_k^2(\boldsymbol{x}) = c_k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{c}_k(\boldsymbol{x})^{\mathsf{T}}(\boldsymbol{C}_k + \sigma^2 \boldsymbol{I}_n)^{-1}\boldsymbol{c}_k(\boldsymbol{x}),$$

where $\boldsymbol{c}_k(\boldsymbol{x}) = (c(\boldsymbol{x}, \boldsymbol{x}_1; \boldsymbol{\theta}_k), \ldots, c(\boldsymbol{x}, \boldsymbol{x}_n; \boldsymbol{\theta}_k))^{\mathsf{T}}$, and $\boldsymbol{C}_k = (c(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}_k))_{i,j=1}^n$. The corresponding log-likelihood presented in Gu and Shen [65], up to some additive constant, is

$$\begin{aligned}\ell(\boldsymbol{\Theta}, \sigma^2; \boldsymbol{X}, \boldsymbol{Y}) = \quad & -\frac{np}{2}\log\sigma^2 - \frac{1}{2}\sum_{k=1}^{q}\log|\sigma^{-2}\boldsymbol{C}_k + \boldsymbol{I}_n| \\ & - \frac{1}{2\sigma^2}\left(\operatorname{vec}(\boldsymbol{Y})^{\mathsf{T}}\operatorname{vec}(\boldsymbol{Y}) - \sum_{k=1}^{q}\boldsymbol{\psi}_k^{\mathsf{T}}\boldsymbol{Y}^{\mathsf{T}}(\sigma^2\boldsymbol{C}_k^{-1} + \boldsymbol{I}_n)^{-1}\boldsymbol{Y}\boldsymbol{\psi}_k\right). \end{aligned} \tag{3.17}$$

In Equation (3.17), the collection of covariance hyperparameters is $\boldsymbol{\Theta} = (\boldsymbol{\theta}_k)_{k=1}^q$. Gu and Shen [65] have presented the covariance expression in Equation (3.16) as $\boldsymbol{\Psi}(\boldsymbol{\Sigma}_{\boldsymbol{g}}(\boldsymbol{x}) + \sigma^2\boldsymbol{I}_p)\boldsymbol{\Psi}^{\mathsf{T}} + \sigma^2(\boldsymbol{I}_p - \boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathsf{T}})$, allowing for an interpretation where $\sigma^2(\boldsymbol{I}_p - \boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathsf{T}})$ is the loss in truncating transformation basis. The computationally efficient form involving k of the $n \times n$ matrix inverses is a direct result of the homogeneous assumption with the application of Woodbury identities in Equations (3.1–3.2).

When the simulation outputs are stochastic, the variability in the outputs propagates to the latent components via the basis transformation. A desirable property of a surrogate is that it represents the dependence of variability among the outputs. In the remainder of the chapter, we present the latent component Gaussian process (LCGP), a GP-based surrogate

method that provides high model fidelity and is computationally efficient. Specifically, LCGP extends GPPCA to the case of *heterogeneous* error structure.

3.3. Latent component Gaussian process

Consider the basis representation model for the output from $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^p$, a stochastic simulation model evaluated at an input $\boldsymbol{x}_i$, as

$$\begin{aligned}
\boldsymbol{y}(\boldsymbol{x}_i) &= \boldsymbol{f}(\boldsymbol{x}_i) + \boldsymbol{\varepsilon}_i, \quad \boldsymbol{\varepsilon}_i \sim \mathsf{N}(\boldsymbol{0}, \boldsymbol{\Sigma}), \\
\boldsymbol{f}(\boldsymbol{x}_i) &= \boldsymbol{\Psi}\boldsymbol{g}(\boldsymbol{x}_i), \\
g_k(\cdot) &\sim \mathsf{GP}(0, c(\cdot, \cdot; \boldsymbol{\theta}_k)), \quad k = 1, \ldots, q,
\end{aligned}$$

with $\boldsymbol{y}(\boldsymbol{x}_i) \in \mathbb{R}^p$ being the observed output, $\boldsymbol{\varepsilon}_i \in \mathbb{R}^p$ being the observation error with covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$, $\boldsymbol{\Psi} \in \mathbb{R}^{p \times q}$ being the transformation basis, and $\boldsymbol{g}(\cdot) \in \mathbb{R}^q$ containing independent latent components, each following a GP with covariance function $c(\cdot, \cdot; \boldsymbol{\theta}_k)$ parameterized by hyperparameter $\boldsymbol{\theta}_k$. Collect all GP hyperparameters as $\boldsymbol{\Theta} = (\boldsymbol{\theta}_k)_{k=1}^q$. This formulation is the same as Equations (3.12–3.14), developed by Gu and Shen [65]. However, we consider a general error covariance structure instead of a homogeneous one. Suppose observations are collected from a fixed set of inputs $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^\mathsf{T}$. Denote the observed data as $\boldsymbol{Y} \in \mathbb{R}^{n \times p}$, and the latent components as $\boldsymbol{G} \in \mathbb{R}^{n \times q}$. Recall that $\boldsymbol{Y}$ is arranged such that each row represents one output corresponding to an input, and each column represents all outputs at one location. Furthermore, we consider $\boldsymbol{Y}$ to be standardized in each column such that each column has zero mean and approximately unit variance. The details of standardization choice are deferred to Section 3.5.

The predictive equations follow Equations (3.7) and (3.8) and the log-likelihood function follows Equation (3.9), substituting $\mathbf{K} = (\boldsymbol{\Psi} \otimes \boldsymbol{I}_n)\mathbf{C}(\boldsymbol{\Psi}^\mathsf{T} \otimes \boldsymbol{I}_n)$. Let us examine the covariance matrix, $\mathbf{K} + \boldsymbol{\Sigma}$, requiring inversion. After substituting $\mathbf{K}$ with its basis representation version, we have $\mathbf{K} + \boldsymbol{\Sigma} = (\boldsymbol{\Psi} \otimes \boldsymbol{I}_n)\mathbf{C}(\boldsymbol{\Psi}^\mathsf{T} \otimes \boldsymbol{I}_n) + \boldsymbol{\Sigma} \otimes \boldsymbol{I}_n$. Applying the Woodbury identities from Equations (3.1–3.2) to $\mathbf{K} + \boldsymbol{\Sigma}$, ignoring for now whether the identities actually help reduce computational costs, results in

$$
\begin{aligned}
(\mathbf{K} + \boldsymbol{\Sigma})^{-1} &= \boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_n \\
&\quad - (\boldsymbol{\Sigma}^{-1}\boldsymbol{\Psi} \otimes \boldsymbol{I}_n)\left[\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Psi} \otimes \boldsymbol{I}_n + \mathbf{C}^{-1}\right]^{-1}(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_n), \quad \text{and} \qquad (3.18) \\
|\mathbf{K} + \boldsymbol{\Sigma}| &= |\boldsymbol{\Sigma}|\ |\mathbf{C}|\ |\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Psi} \otimes \boldsymbol{I}_n + \mathbf{C}^{-1}|.
\end{aligned}
$$

Examining Equation (3.18), the first term involves inverting $\boldsymbol{\Sigma}$, a $p \times p$ error covariance matrix, and the second term involves inverting a matrix of size $nq \times nq$. The latter matrix is in general much larger. While this matrix can be considerably smaller than the original covariance matrix, which is of size $np \times np$ (because $q \leq p$), the computational cost equals $\mathcal{O}(n^3q^3)$ and is still considered to be too high for practical surrogate usage. The special case is when $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}_p$, as in Equation (3.16), because the matrix simplifies to be block diagonal. However, this matrix is no longer block diagonal when a general error covariance $\boldsymbol{\Sigma}$ is considered.

Inverting the matrix, $\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Psi} \otimes \boldsymbol{I}_n + \mathbf{C}^{-1}$, is at the center of solving the issue at hand, namely, build a surrogate for a simulation model with multiple outputs and a heterogeneous error structure across the outputs. Noting this challenge, we propose to choose and estimate the transformation basis in a way such that constructing this

surrogate is not prohibited by its cost while addressing heterogeneous errors. Moreover, the stochasticity in the latent components are fully incorporated.

3.3.1. Modeling of transformation basis. Consider choosing the transformation basis to be the product of some other transformation matrix and the square root of the error covariance matrix, i.e.,

$$\boldsymbol{\Psi} = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Phi}, \quad \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi} = \boldsymbol{D}, \tag{3.19}$$

where $\boldsymbol{\Sigma}^{\frac{1}{2}}$ is defined such that $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Sigma}^{\frac{1}{2}}$, $\boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_q) \in \mathbb{R}^{p\times q}$ is an orthogonal transformation matrix, and $\boldsymbol{D} = \text{diag}(d_k)_{k=1}^{q}$. The existence of $\boldsymbol{\Sigma}^{\frac{1}{2}}$ is a result from $\boldsymbol{\Sigma}$ being a symmetric positive definite matrix [70, ch.21]. Define $\boldsymbol{\Sigma}^{-\frac{1}{2}}$ such that if $\boldsymbol{\Sigma}^{-1}$ exists, then $\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Sigma}^{-\frac{1}{2}}$. The orthogonal matrix $\boldsymbol{\Phi}$ can be chosen from the singular value decomposition of $\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y}$, and $\boldsymbol{D}$ is the diagonal matrix containing the square of the singular values. In the case where $q \leq p$, the first q columns of $\boldsymbol{\Phi}$ and the corresponding singular values are chosen.

This basis choice is the key insight to a computationally efficient method while enriching model fidelity. Note that this basis requires estimation, since $\boldsymbol{\Sigma}$ is unknown. The estimation of unknown error covariance and other quantities are detailed in Section 3.3.3.

3.3.2. Main results. We present the main results for LCGP, including the posterior of latent components, the predictive equations, and the log-likelihood function.

THEOREM 3.1 (Posterior distribution of latent components). *The posterior distribution of each latent component follows a multivariate Gaussian distribution*

$$\boldsymbol{G}_k \mid \boldsymbol{Y}, \boldsymbol{\Sigma}, \boldsymbol{\theta}_k \sim \mathsf{N}\left(\boldsymbol{m}_k, \boldsymbol{S}_k\right),$$

where $\boldsymbol{m}_k = (d_k \boldsymbol{I}_n + \boldsymbol{C}_k^{-1})^{-1} \boldsymbol{Y} \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\phi}_k$ *and* $\boldsymbol{S}_k = (d_k \boldsymbol{I}_n + \boldsymbol{C}_k^{-1})^{-1}$.

The results follow from the observation that the posterior density of $\boldsymbol{G}$ is proportional to the product of two Gaussian densities. The full proof for Theorem 3.1 is provided in Appendix C.1.

THEOREM 3.2 (Marginal distribution of simulation output). *The marginal distribution of* $\mathrm{vec}(\boldsymbol{Y})$ *follows a multivariate Gaussian distribution*

$$\mathrm{vec}(\boldsymbol{Y}) \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta} \sim \mathsf{N}(\boldsymbol{0}, \mathsf{K} + \boldsymbol{\Sigma}),$$

where the inverse of $\mathsf{K} + \boldsymbol{\Sigma}$ *is*

$$(\mathsf{K} + \boldsymbol{\Sigma})^{-1} = \boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_n - \sum_{k=1}^{q} \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\phi}_k \boldsymbol{\phi}_k^{\mathsf{T}} \boldsymbol{\Sigma}^{-\frac{1}{2}} \otimes \left(d_k \boldsymbol{I}_n + \boldsymbol{C}_k^{-1}\right)^{-1}.$$

The results can be achieved by integrating out $\boldsymbol{G}$. The full proof for Theorem 3.2 is provided in Appendix C.1.

The posterior of the latent components $\boldsymbol{G}$ decouples into a product of the posterior of the q components, conditional on $\boldsymbol{Y}, \boldsymbol{\Sigma}$, and their corresponding hyperparameter $\boldsymbol{\theta}_k$. This decoupling further leads to tractable computations as the original matrix inversion required is of size $np \times np$, whereas now only q of the $n \times n$ matrix inversions are required. As a result, computations are greatly reduced.

COROLLARY 3.1 (log-likelihood function). *From Theorem 3.2, the log-likelihood function, up to an additive constant, is*

$$\ell(\boldsymbol{\Sigma}, \boldsymbol{\Theta} \mid \boldsymbol{X}, \boldsymbol{Y}) \tag{3.20}$$

$$= -\frac{n}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{k=1}^{q}\log|d_k\boldsymbol{C}_k + \boldsymbol{I}_n|$$

$$-\frac{1}{2}\mathrm{vec}(\boldsymbol{Y})^{\mathsf{T}}\Big(\boldsymbol{\Sigma}^{-1}\otimes\boldsymbol{I}_n - \sum_{k=1}^{q}\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\phi}_k\boldsymbol{\phi}_k^{\mathsf{T}}\boldsymbol{\Sigma}^{-\frac{1}{2}}\otimes\big(d_k\boldsymbol{I}_n + \boldsymbol{C}_k^{-1}\big)^{-1}\Big)\mathrm{vec}(\boldsymbol{Y}). \tag{3.21}$$

THEOREM 3.3 (Predictive distribution of latent components). *For any new input $\boldsymbol{x}$, the predictive distribution of the kth latent component is given as a Gaussian distribution with mean and variance*

$$\mu_k(\boldsymbol{x}) = \mathbb{E}[g_k(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\Sigma}, \boldsymbol{\theta}_k] = \boldsymbol{c}_k(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{C}_k^{-1}\boldsymbol{m}_k, \tag{3.22}$$

$$\sigma_k^2(\boldsymbol{x}) = \mathbb{V}[g_k(\boldsymbol{x}) \mid \boldsymbol{X}; \boldsymbol{\Sigma}, \boldsymbol{\theta}_k] = c_k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{c}_k^{\mathsf{T}}(\boldsymbol{x})\boldsymbol{T}_k\boldsymbol{c}_k(\boldsymbol{x}), \tag{3.23}$$

with

$$\boldsymbol{T}_k = \boldsymbol{C}_k^{-1} - \boldsymbol{C}_k^{-1}\boldsymbol{S}_k\boldsymbol{C}_k^{-1},$$

$$\boldsymbol{c}_k(\boldsymbol{x}) = (c(\boldsymbol{x}, \boldsymbol{x}_1; \boldsymbol{\theta}_k), \ldots, c(\boldsymbol{x}, \boldsymbol{x}_n; \boldsymbol{\theta}_k))^{\mathsf{T}}, \boldsymbol{C}_k = (c(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}_k))_{i,j=1}^{n},$$

and $\boldsymbol{m}_k, \boldsymbol{S}_k$ are the posterior mean and covariance of the latent components in Theorem 3.1.

THEOREM 3.4 (Predictive distribution of simulation output). *For any new input $\boldsymbol{x}$, the predictive distribution of the output is given as a multivariate Gaussian distribution*

with mean and covariance

$$\boldsymbol{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\Sigma}, \boldsymbol{\Theta}] = \boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{\Phi} \boldsymbol{\mu}_g(\boldsymbol{x}), \tag{3.24}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{y}}(\boldsymbol{x}) = \mathbb{V}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}; \boldsymbol{\Sigma}, \boldsymbol{\Theta}] = \boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{\Phi} \boldsymbol{\Sigma}_g(\boldsymbol{x}) \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{\Sigma}^{\frac{1}{2}}. \tag{3.25}$$

where $\boldsymbol{\mu}_g(\boldsymbol{x}) = (\mu_1(\boldsymbol{x}), \ldots, \mu_q(\boldsymbol{x}))^{\mathsf{T}}$ *and* $\boldsymbol{\Sigma}_g(\boldsymbol{x}) = \mathrm{diag}(\sigma_1^2(\boldsymbol{x}), \ldots, \sigma_q^2(\boldsymbol{x}))$.

The proofs for Theorems 3.3 and 3.4 are provided in Appendix C.1.

3.3.2.1. *Case of noiseless simulation outputs.* In the case where the simulation outputs are deterministic, or noiseless, the construction of transformation basis in Equation (3.19) has to be modified because letting $\boldsymbol{\Sigma} = \mathbf{0}$ results in a degenerate transformation basis, i.e., $\boldsymbol{\Psi} = \mathbf{0}$. The degenerate basis causes the predictions in Equations (3.22–3.23) to be undefined, which in turn causes the overall predictions in Equations (3.24–3.25) to be undefined. In such case, the transformation basis should be chosen as the orthonormal basis $\boldsymbol{\Phi}$, such that $\boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{\Phi} = \boldsymbol{I}_q$. As a result, the GP predictions in the noiseless case resemble the Principal Component Gaussian Process method in Equations (2.2) and (2.3).

3.3.3. Estimation of error covariance and hyperparameters. The estimation of error covariance and hyperparameters are performed via maximizing the log-likelihood function in Equation (3.9). The log-likelihood function with respect to $\boldsymbol{\Sigma}$ and hyperparameters $\boldsymbol{\Theta}$ does not provide an analytical form in general [36, 48, 65]. Therefore, numerical methods such as L-BFGS or gradient descent are employed to approximately solve for the estimators [120]. By fixing the covariance hyperparameters to their maximum likelihood estimates, the uncertainty about the hyperparameters is neglected [87]. To account for this source of uncertainty, one may provide a prior distribution over these quantities

and conduct a full Bayesian analysis to sample the posterior. However, due to the large number of samples typically required in employing Markov chain Monte Carlo (MCMC), the computational burden is greatly increased as the matrix inversions are necessary in each MCMC step.

3.3.4. Common error covariance structures. The structure of error covariance $\boldsymbol{\Sigma}$ can be chosen a-priori for efficient model fitting. Two common choices for this structure are homogeneous error and location-dependent error.

3.3.4.1. *Homogeneous error.* The following result shows that LCGP recovers the GP-PCA method presented in Gu and Shen [65] by choosing the parameters appropriately.

COROLLARY 3.2 (LCGP with homogeneous error). *Let* $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_p$. *The LCGP predictive equations and log-likelihood Equations* (3.21), (3.24) *and* (3.25) *are equivalent to those presented in Gu and Shen [65], i.e., Equations (3.15–3.17).*

PROOF. Choose $d_k = \sigma^{-2}$. □

3.3.4.2. *Location-dependent error.* Location-dependent errors are common when observation errors are expected to vary by the output location. In this case, the error covariance matrix assumes the form

$$\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_p^2).$$

For example, in nuclear physics applications (e.g., [76, 21, 144, 106]), the output locations correspond to various physical quantities that have significantly different behavior. The simulation outputs are often standardized such that they are comparable across locations, as done in Higdon et al. [76]. However, even in this diagonal error covariance case, the

predictive equations and log-likelihood function in Equations (3.24–3.25), and (3.21) do not further simplify, as the transformation basis now involves interactions of the error variances and the orthogonal matrix, where $\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\phi}_k = (\sigma_1^{-1}\phi_{1k}, \ldots, \sigma_p^{-1}\phi_{pk})^{\mathsf{T}}$. Therefore, the outer-product of such a vector produces a dense matrix.

3.3.5. Comparison with other approaches for latent component uncertainty. The LCGP approach accounts for the uncertainty in $\boldsymbol{g}(\boldsymbol{x})$ by considering the posterior distribution of the latent components,as in Theorem 3.1. We compare our approach with two other approaches in modeling the latent components to highlight the gain in model representation without incurring additional computational costs. The two other approaches are (i) deterministic latent components, and (ii) fully independent latent components.

3.3.5.1. *Deterministic latent components.* Deterministic latent components assume no stochasticity in the latent components. This is also referred to as the profile likelihood approach where $\boldsymbol{G}$ takes value that maximizes the joint log-likelihood function. The profile likelihood function is defined as $\text{pl}(\boldsymbol{\Sigma}, \boldsymbol{\Theta}; \boldsymbol{X}, \boldsymbol{Y}) = \max_{\boldsymbol{G}} p(\boldsymbol{Y}, \boldsymbol{G} \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta})$ [118], where the notation $p(y \mid x)$ refers to the conditional probability density function of y given x. Specifically, the details of $p(\boldsymbol{Y}, \boldsymbol{G} \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta})$ are elaborated in Equation (C.1). Then, we proceed with the estimation of error covariance and hyperparameters by maximizing the profile likelihood function. This approach is identical to when the simulation output is considered noiseless, as in Higdon et al. [74], Fricker et al. [48], and Sürer et al. [144]. The resulting predictive equations follow Equations (3.24) and (3.25) but the predictive

equations for the latent components, i.e., Equations (3.22) and (3.23), are replaced by

$$\mu_k^{\text{det}}(\boldsymbol{x}) = \boldsymbol{c}_k(\boldsymbol{x})^\mathsf{T} \boldsymbol{C}_k^{-1} \boldsymbol{m}_k,$$

$$\sigma_k^{2,\text{det}}(\boldsymbol{x}) = c_k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{c}_k^\mathsf{T}(\boldsymbol{x}) \boldsymbol{C}_k^{-1} \boldsymbol{c}_k(\boldsymbol{x}).$$

3.3.5.2. *Fully independent latent components.* Fully independent latent components assume that each element of $\boldsymbol{G}$, G_{ik}, is independently distributed with mean M_{ik} and variance V_{ik}. This is also referred to as the mean-field approximation in the context of variational inference [20], where the posterior is approximated by the member of a specified distributional family that minimizes the Kullback-Leibler (KL) divergence from the approximation to the true posterior. Equivalently, one may maximize the evidence lower bound objective (ELBO), which is the negative of the KL divergence up to an additive constant [20]. This approximate posterior is also termed as the variational distribution. To estimate the error covariance and hyperparameters, the ELBO is maximized. Concretely, let $G_{ik} \overset{\text{iid}}{\sim} \mathsf{N}(M_{ik}, V_{ik}), i = 1 \ldots, n, k = 1, \ldots, q$. Collectively denote $\boldsymbol{M} = (M_{ik})$ and $\boldsymbol{V} = (V_{ik})$. Let $q(\cdot)$ be the probability density of the variational distribution, or "variational density." To minimize confusion in notation, in all instances when the variational density is mentioned, $q(\cdot)$ is used. The scalar variable q on its own is always used to denote the number of latent components. Then, the ELBO is defined as

$$\mathsf{ELBO}(\boldsymbol{M}, \boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{\Theta}; \boldsymbol{Y}) = \mathbb{E}[\log p(\boldsymbol{Y} \mid \boldsymbol{G}) + \log p(\boldsymbol{G}) - \log q(\boldsymbol{G})], \tag{3.26}$$

where all expectations are taken with respect to $q(\cdot)$.

THEOREM 3.5 (Evidence lower bound objective for LCGP). *The ELBO function for LCGP, assuming fully independent latent components, up to an additive constant, is*

$$\begin{aligned}\mathsf{ELBO}(\boldsymbol{M}, \boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{\Theta}; \boldsymbol{Y}) = &-\frac{n}{2}\log|\boldsymbol{\Sigma}| \\ &-\frac{1}{2}\left(\mathrm{vec}(\boldsymbol{Y}-\boldsymbol{M}\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Sigma}^{\frac{1}{2}})^{\mathsf{T}}(\boldsymbol{\Sigma}^{-1}\otimes\boldsymbol{I}_n)\mathrm{vec}(\boldsymbol{Y}-\boldsymbol{M}\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Sigma}^{\frac{1}{2}}\right) \\ &+\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{q}\left(\log V_{ik}-\left[d_k+(\boldsymbol{C}_k^{-1})_{ii}\right]V_{ik}\right) \\ &-\frac{1}{2}\log|\mathsf{C}|-\frac{1}{2}\mathrm{vec}(\boldsymbol{M})^{\mathsf{T}}\mathsf{C}^{-1}\mathrm{vec}(\boldsymbol{M}).\end{aligned}$$

The proof for Theorem 3.5 involves evaluating the expectation of each term in Equation (3.26). The details are included in Appendix C.1.

THEOREM 3.6 (Variational distribution of latent components). *Assuming fully independent latent components, the variational distribution of* $\boldsymbol{G}$ *is a product of Gaussian distributions. The component* $\boldsymbol{G}_k$ *has mean and covariance*

$$\widehat{\boldsymbol{M}}_k = (d_k\boldsymbol{I}_n+\boldsymbol{C}_k^{-1})^{-1}\boldsymbol{Y}\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\phi}_k \quad \textit{and } \widehat{\boldsymbol{V}}_k = \mathrm{diag}\left(d_k\boldsymbol{I}_n+\boldsymbol{C}_k^{-1}\right)^{-1}$$

respectively, where $\mathrm{diag}(\boldsymbol{A})$ *extracts the main diagonal of matrix* $\boldsymbol{A}$*. In other words, this choice of* $\boldsymbol{M}$ *and* $\boldsymbol{V}$ *maximizes the ELBO function.*

The proof for Theorem 3.6 involves solving for $\boldsymbol{M}$ and $\boldsymbol{V}$ under first-order conditions, as shown in Appendix C.1.

Note that $\widehat{\boldsymbol{M}}_k = \boldsymbol{m}_k$, recovering the mean of the posterior of $\boldsymbol{G}$ in Theorem 3.1. On the other hand, the variances $\widehat{\boldsymbol{V}}_k = \left(d_k+(\boldsymbol{C}_k^{-1})_{ii}\right)^{-1}$. Similar to the deterministic latent components, the predictive equations follow Equations (3.24) and (3.25) but the predictive

equations for the latent components Equations (3.22) and (3.23) are replaced by

$$\mu_k^{\mathsf{ELBO}}(\boldsymbol{x}) = \boldsymbol{c}_k(\boldsymbol{x})^{\mathsf{T}} \boldsymbol{C}_k^{-1} \boldsymbol{m}_k,$$
$$\sigma_k^{2,\mathsf{ELBO}}(\boldsymbol{x}) = c_k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{c}_k^{\mathsf{T}}(\boldsymbol{x}) \left[\boldsymbol{C}_k^{-1} - \boldsymbol{C}_k^{-1} \mathrm{diag}(\widehat{\boldsymbol{V}}_k) \boldsymbol{C}_k^{-1} \right] \boldsymbol{c}_k(\boldsymbol{x}).$$

We note that all three approaches require the inversion of the $\boldsymbol{C}_k$'s, and any other required computations in the full posterior can be derived from the eigendecompositions of the $\boldsymbol{C}_k$.

In summary, the form of the predictive means of the latent components achieved from these three approaches are identical, while the predictive variances differ. The computation required in all three approaches is similar as the required matrix inverses can be obtained from the same number of matrix decompositions (see Section 3.5 and Appendix C for details.) Table 3.1 presents the predictive variances of the latent components for each approach. Firstly, it is clear that the variances of latent components are the lowest in the deterministic approach, which means the function variances are largely absorbed by the estimation of the error covariance. Secondly, even though the predictive means have the same form across methods, since the predictive means depend on the estimated GP hyperparameters, the actual predictive means typically differ. An illustration of the behaviors of the three treatment approaches of latent components is included in Section 3.4.

	Latent components	
	Mean, $\mu_k(\boldsymbol{x})$	Variance, $\sigma_k^2(\boldsymbol{x})$
Deterministic	$\boldsymbol{c}_k(\boldsymbol{x})^{\mathsf{T}} \boldsymbol{C}_k^{-1} \boldsymbol{m}_k$	$c_k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{c}_k^{\mathsf{T}}(\boldsymbol{x}) \boldsymbol{C}_k^{-1} \boldsymbol{c}_k(\boldsymbol{x})$
Fully independent		$c_k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{c}_k^{\mathsf{T}}(\boldsymbol{x}) \left[\boldsymbol{C}_k^{-1} - \boldsymbol{C}_k^{-1} \mathrm{diag}(\widehat{\boldsymbol{V}}_k) \boldsymbol{C}_k^{-1} \right] \boldsymbol{c}_k(\boldsymbol{x})$
Full posterior		$c_k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{c}_k^{\mathsf{T}}(\boldsymbol{x}) \left[\boldsymbol{C}_k^{-1} - \boldsymbol{C}_k^{-1} \boldsymbol{S}_k \boldsymbol{C}_k^{-1} \right] \boldsymbol{c}_k(\boldsymbol{x})$

Table 3.1. Predictive mean and variance of latent components

3.4. Illustrative 3D function with 1D input

Consider the following function $\boldsymbol{f}(x) = (f_1(x), f_2(x), f_3(x))^{\mathsf{T}}, x \in [0, 1]$, where

$$f_1(x) = (6x - 2)^2 \sin(12x - 4),$$

$$f_2(x) = 0.5 f_1(x) + 5(x - 0.5) + 5, \text{ and}$$

$$f_3(x) = -0.8 f_1(x) - 5(x - 0.5) - 4.$$

This function is developed in Forrester et al. [46, ch.8], originally to demonstrate the emulation of multi-fidelity functions. The functions $f_2(x)$ and $f_3(x)$ are modifications of $f_1(x)$, given by the form $f_j(x) = a_{1j} f_1(x) + a_{2j}(x - 0.5) - a_{3j}, j = 2, 3$ with constants a_{kj}'s. The function represents the case of a multivariate output with correlations among output locations. Suppose that we collect 100 noisy observations from this function $\boldsymbol{y}(x_i) = \boldsymbol{f}(x_i) + \boldsymbol{\varepsilon}_i, \boldsymbol{\varepsilon}_i \overset{\text{iid}}{\sim} \mathsf{N}(\mathbf{0}, \text{diag}(\sigma_1^2, \sigma_2^2, \sigma_3^2))$, for $x_i = (i - 1)/(n - 1), i = 1, \ldots, 100$. The values of $(\sigma_1^2, \sigma_2^2, \sigma_3^2)$ are set to $(0.10, 0.89, 5.32)$.

Two variants of LCGP are tested here, denoted as LCGP and LCGP_robust. The former standardizes the output by the mean and standard deviation for each output location; the latter standardizes the output at each location by the median and median absolute deviation, defined as $\text{med}\{y_j(x_i)\}$ and $\text{med}\{|y_j(x_i) - m_j|\}$ respectively. The discussion of standardization choices is deferred to Section 3.5 along with other implementation details. Both variants use a Matérn 3/2 covariance function for the latent GP components [137]. GPPCA [65] and OILMM [25] are also included for comparison. Both methods employ a basis representation GP approach, while GPPCA assumes a constant error variance across outputs, and OILMM augments a normalizing flow to the output at each location

while estimating the error covariance and GP hyperparameters. In particular, OILMM seeks to learn a linear transformation for each output in estimation. The number of latent components for all three methods are set to $q = p = 3$, equaling the output dimension.

Figure 3.1 presents the predictions by LCGP, LCGP_robust, GPPCA, and OILMM. GPPCA underestimates the variance in the last location and incurs large estimation errors. This underestimation is attributed to the assumption of a homogeneous error across the outputs. In contrast, LCGP, LCGP_robust, and OILMM more accurately captured the true mean of the output as they account for the heterogeneous error structure. In this illustration, LCGP_robust recovers the mean better than LCGP. However, as n increases, the difference between the standardization procedures becomes negligible.

3.4.1. Comparison of latent component uncertainty treatments. This subsection compares LCGP (which uses the full posterior), the ELBO, and the profile likelihood approaches introduced in Section 3.3.5 for estimating the error covariance and the GP hyperparameters. Figure 3.2 shows the predictions using these approaches. Estimation with the full posterior results in the most accurate prediction of the true mean, followed by the ELBO, then the profile likelihood. Since all approaches incur similar computational costs, the full posterior should be used. Figure 3.3 shows the behavior of the latent components in their predictions. The full posterior approach results in more expressive latent functions, while the ELBO results in comparatively less expressive ones. The ELBO approach achieves an approximation to the posterior function. As a result, the predictive performance is subpar to that of the full posterior approach. The profile likelihood approach only results in variations in the first component, which means that only the first principal component contributed in prediction. Since the profile likelihood approach

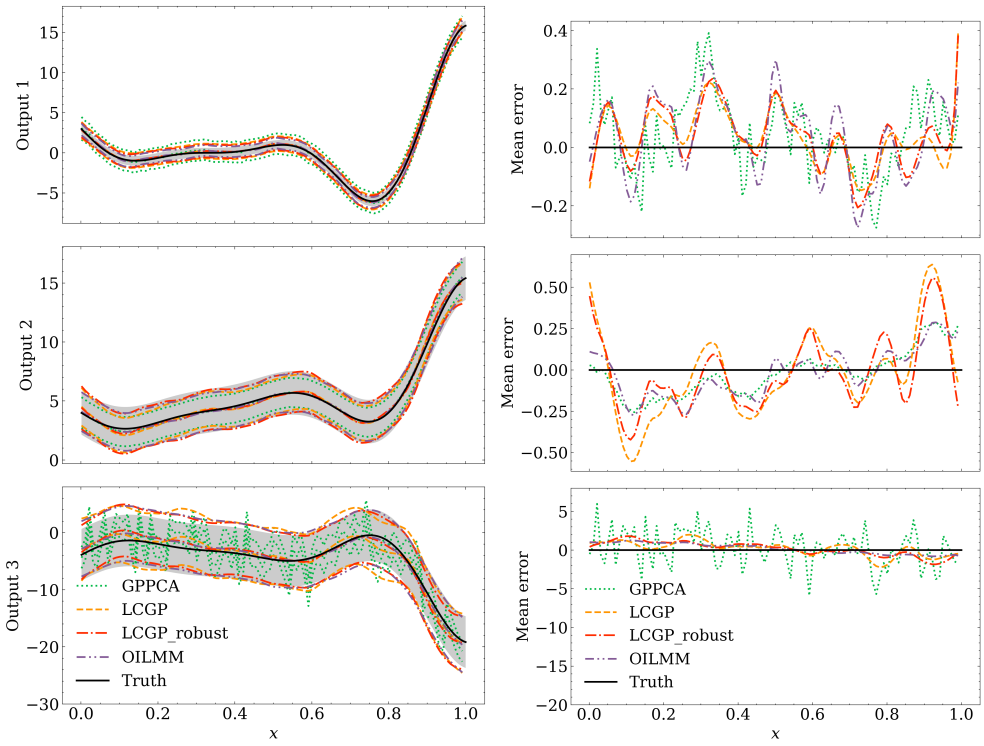


Figure 3.1. Comparison of basis representation GP methods. Each row presents one output location. The predictive mean and 95% predictive interval (left), and estimation errors of predictive means (right) are plotted for each method. The true functions are plotted in black with its 95% predictive intervals shaded in gray. LCGP and OILMM recover the true function and the heterogeneous errors while GPPCA does not.

largely assigns the output variance to the error covariance, the resulting prediction quality suffers, specifically in its lack of ability capturing the heterogeneous error variances.

3.5. Implementation and numerical experiments

This section provides the implementation details of LCGP and reports the results from numerical experiments to highlight the strength of using LCGP to emulate stochastic multivariate outputs.

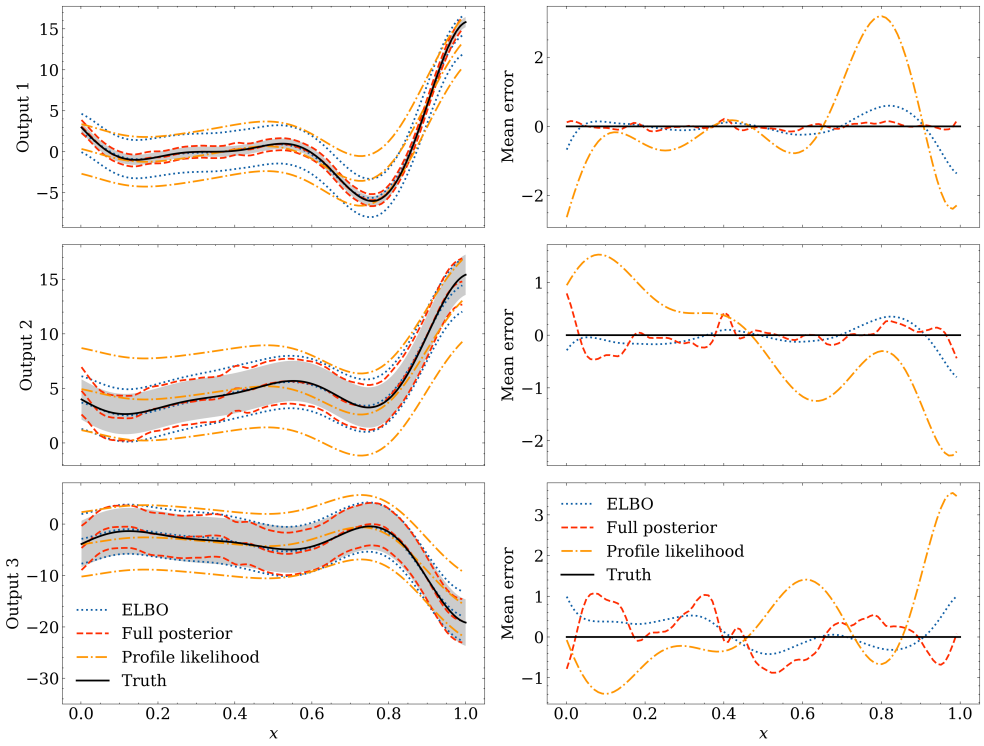


Figure 3.2. Comparison of latent component uncertainty treatments. Each row presents one output location. The predictive mean and 95% predictive interval (left), and estimation errors of predictive means (right) are plotted for each approach. Both the full posterior and ELBO approach recover the function mean, while the profile likelihood approach does not.

3.5.1. Implementation details. The Python implementation of LCGP is available in a git repository (`https://github.com/mosesyhc/lcgp`) or as `lcgp` on PyPI, the Python Package Index [149].

The implementation is written using PyTorch [122], an open-source Python framework that facilitates developing and deploying of machine learning models. LCGP leverages PyTorch's auto differentiation engine to compute gradients of functions, for example, the log posterior function. The estimations of error covariance and hyperparameters are performed by approximately minimizing the negative log posterior function using

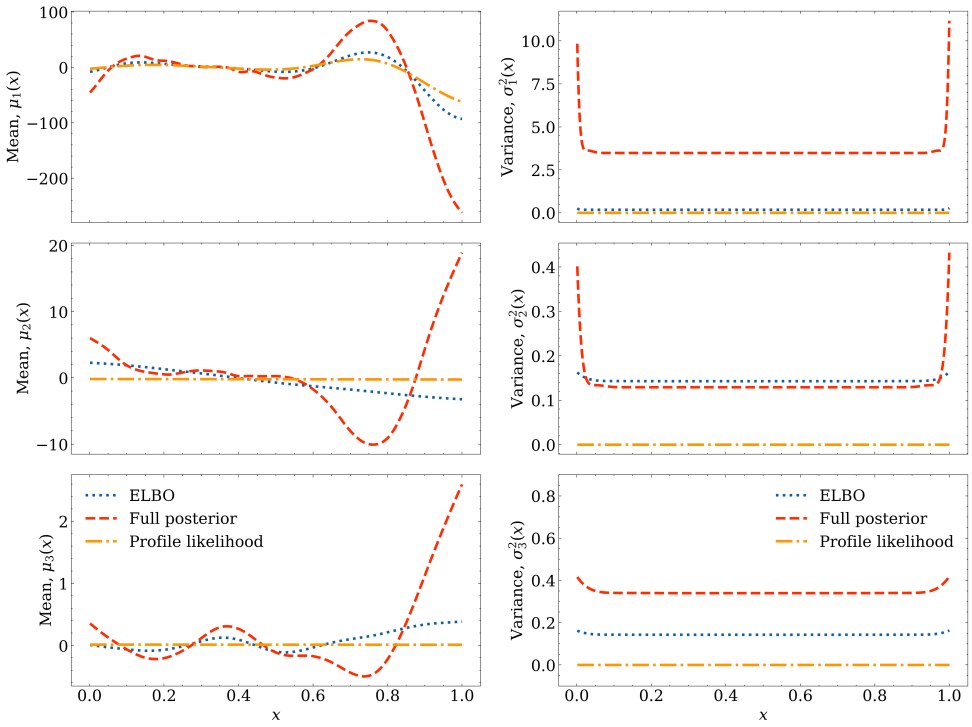


Figure 3.3. Comparison of latent component behaviors. Each row presents one latent component. Predictive means (left) and variances (right) are plotted. The predictive means from the full posterior approach each captures some functional behavior, where the ELBO and profile likelihood approaches exhibit tapering of functional dynamics. The predictive variances from the full posterior approach are generally larger than those from the ELBO and the profile likelihood.

`PyTorch-LBFGS`, an implementation of the L-BFGS algorithm [120], available at `https://github.com/hjmshi/PyTorch-LBFGS`. This implementation is chosen over the existing version in PyTorch for its stability and advanced line-search options. The log of covariance hyperparameters and error variances are implemented to ensure they are positive. The default values for the absolute tolerance and the projected gradient tolerance are set at 2×10^{-10} and 1×10^{-6} respectively. A maximum iteration budget of 1000 is used by default. A Matérn 3/2 covariance function is used as the GP covariance function by default.

The inputs are standardized into the unit hypercube and the outputs are standardized within each location before applying LCGP. Two standardization options are coded: (1) using mean and standard deviation, and (2) using median and median absolute deviation. Both standardization approaches allow guided estimations of the GP hyperparameters. The two options are both included in the numerical experiments for evaluations. In addition, grouping of errors is implemented such that locations of similar quantities can share an error variance estimate. For example, a diagonal grouping of three types of outputs has $\boldsymbol{\Sigma} = \text{bdiag}(\sigma_1^2 \boldsymbol{I}_{p_1}, \sigma_2^2 \boldsymbol{I}_{p_2}, \sigma_3^2 \boldsymbol{I}_{p_3})$, where the number of corresponding locations are p_1, p_2, and p_3 respectively, and $p_1 + p_2 + p_3 = p$. Regarding numerical computations, all matrix inverses are performed using an eigendecomposition, such that determinant and inverse of multiple closely related matrices can be obtained from computing only one decomposition. Quantities that are reused in prediction are stored to avoid duplicate computations. These details are available in Appendices C.2 to C.4.

3.5.2. Numerical experiments. This subsection covers the numerical experiments to assess the computational performance of LCGP. In addition to the illustrative function, three common test functions (borehole, wing weight, and OTL circuit) are used for comparisons [147]. Let n and n' be the number of observations for training and testing an emulator respectively. We choose $n = 50, 100, 250, 500$, and $n' = 800$ for all test functions. The experiment is repeated for 10 replications. For each method, the maximum number of optimization iteration is set at 1000. The full setup for the experiments are provided in following sections grouped by the input dimension (1D or higher). We report the following

evaluation metrics using the set of testing samples for each method:

$$\text{Training time,}$$

$$\text{Root mean squared error (RMSE)} = \sqrt{\frac{1}{n'}\sum_{i=1}^{n'} \|\boldsymbol{\mu}_{\boldsymbol{y}}(\boldsymbol{x}_i') - \boldsymbol{f}(\boldsymbol{x}_i')\|_2^2},$$

$$\text{Normalized root mean squared error (NRMSE)} = \sqrt{\frac{1}{n'}\sum_{i=1}^{n'}\sum_{j=1}^{p}\left(\frac{\mu_{\boldsymbol{y},j}(\boldsymbol{x}_i') - f_j(\boldsymbol{x}_i')}{\max_i\{f_j(\boldsymbol{x}_i')\} - \min_i\{f_j(\boldsymbol{x}_i')\}}\right)^2},$$

$$\text{95\% empirical coverage} = \frac{1}{n'p}\sum_{i=1}^{n'}\sum_{j=1}^{p} \mathbf{1}\{y_j(\boldsymbol{x}_i') \in \text{PI}_{ij}(95\%)\}, \text{ and}$$

$$\text{Dawid-Sebastani score (D-S score)} = \frac{1}{n'}\sum_{i=1}^{n'}\Bigg(\log|\boldsymbol{\Sigma}_{\boldsymbol{y}}(\boldsymbol{x}_i')| + (\boldsymbol{\mu}_{\boldsymbol{y}}(\boldsymbol{x}_i') - \boldsymbol{f}(\boldsymbol{x}_i'))^{\mathsf{T}}\boldsymbol{\Sigma}_{\boldsymbol{y}}(\boldsymbol{x}_i')^{-1}(\boldsymbol{\mu}_{\boldsymbol{y}}(\boldsymbol{x}_i') - \boldsymbol{f}(\boldsymbol{x}_i'))\Bigg),$$

where $y_j(\boldsymbol{x}_i')$ is the jth output of the observation at input $\boldsymbol{x}_i'$, $\mu_{\boldsymbol{y},j}(\boldsymbol{x}_i')$ is the emulator predictive mean of the jth output, and $\text{PI}_{ij}(95\%)$ is the 95% predictive interval produced for such a quantity. RMSE and NRMSE measure the predictive accuracy in estimating the true mean of the function. While RMSE measures the actual aggregated error, NRMSE measures the overall accuracy when the output at different locations have different scales, and can be used for comparing over multiple test functions. The Dawid-Sebastani score is a proper scoring rule (see e.g., Gneiting and Raftery [55, Eq. 25]) and takes the first two moments of the emulator into account. For a 1D output, the D-S score simplifies to the "Score" proposed by Baker et al. [11] to jointly evaluate the estimation of the mean and variance of the stochastic simulation function. For ease of interpretation, the Dawid-Sebastani score is presented as the negative of the proposed scores in Gneiting and

Raftery [55] and Baker et al. [11]. Therefore, lower RMSE and D-S score values represent better performance. The 95% empirical coverage returns how often the prediction lies inside the 95% predictive interval. The empirical coverage closer to 95% indicates a better performance.

3.5.2.1. *Numerical performance for the illustrative function.* In addition to LCGP, LCGP_robust, GPPCA, and OILMM, we now add a stochastic variational GP (SVGP) into the comparison. Developed in GPyTorch [49], SVGP is an example of a convolved GP that maximizes a variational objective. The number of inducing points for SVGP is set to n, equaling the number of training points. In total, five methods are considered here. The set of training inputs are generated as $x_i = (i-1)/(n-1), i = 1, \ldots, n$, whereas the set of testing inputs are generated as $x_i' = i/n, i = 1, \ldots, n'$; the corresponding noisy observations are collected for all inputs with an error covariance $\mathbf{\Sigma} = \text{diag}(0.10, 0.89, 5.32)$.

Figure 3.4 presents the evaluation metrics for the illustrative function. Among the five methods, OILMM tends to achieve the lowest RMSE, at the expense of a much longer training time. Since OILMM estimates the standardization constants along with the error covariance and GP hyperparameters, it typically requires a larger number of iterations to converge. LCGP and LCGP_robust are the fastest while achieving RMSE comparable with the methods other than OILMM. There are advantages to include standardization in estimation but there is a tradeoff between accuracy and time. GPPCA achieves similar or slightly better RMSE compared with LCGP but requires longer training time, but not as much as OILMM. In general, SVGP cannot compare with the other basis representation GP methods. Since the outputs across locations have comparable scales, NRMSE behaves similarly to RMSE. When comparing with the D-S score, LCGP, LCGP_robust, and

OILMM achieve similar scores. The scores are closer to each other compared with the RMSE, because the score weighs the error in mean estimation by the predictive covariance. GPPCA and SVGP achieve worse D-S scores, also evident in their empirical coverages. Since GPPCA undercovers and SVGP tends to overcover, the D-S scores reflect the subpar estimations in the predictive variances. LCGP, LCGP_robust, and OILMM achieves coverages close to the nominal 95% level.

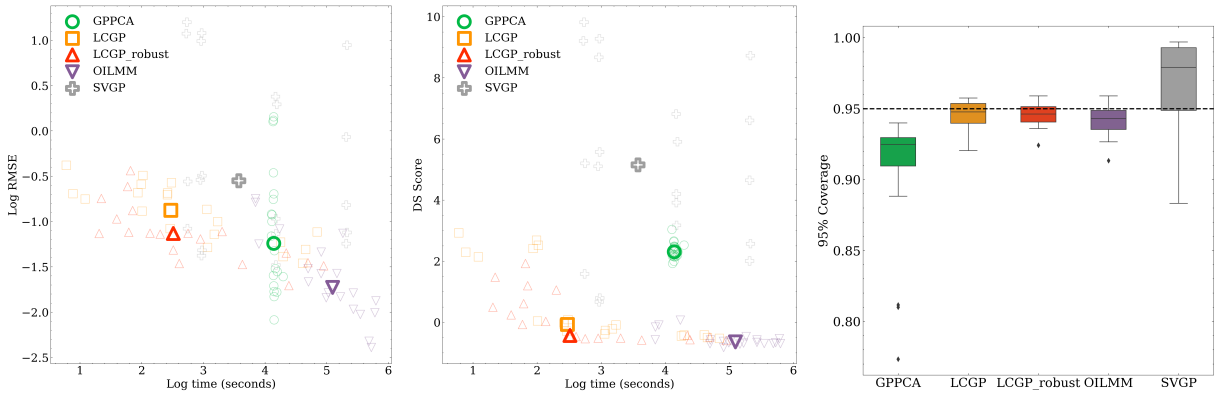


Figure 3.4. Performance comparisons for the five emulator methods using the Forrester 3D illustrative function. Log RMSE (left) and D-S score (center) are plotted against log training time. Median statistics are plotted with bold markers. The 95% empirical coverage (right) are presented in the boxplot and the nominal 95% level is indicated with the dashed line.

3.5.2.2. *Numerical performance for test functions.* For each of the borehole, wing weight, and OTL circuit functions, the input variables are partitioned into locations and inputs (see the supplementary materials of Chan et al. [28] for details). The resulting input dimensions for the three functions are $d = 4, 4, 2$. The inputs of these functions are standardized into a unit hypercube. Outputs of dimensions $p = 4$ and $p = 8$ are generated by first sampling p locations, treating them as fixed, then sampling n training inputs and n' testing inputs using a Latin Hypercube scheme [114]. The noise structures in generating

these outputs are set to

$$\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \ldots, \sigma_p^2) \quad \text{and} \quad \sigma_j^2 = 0.05^{j/2} \int_{\mathcal{X}} [f_j(\boldsymbol{x}) - \bar{f}_j(\boldsymbol{x})]^2 \text{d}\boldsymbol{x}, j = 1, \ldots, p,$$

where f_j is the jth location of the true function and $\bar{f}_j(\boldsymbol{x}) = \int_{\mathcal{X}} f_j(\boldsymbol{x}) \text{d}\boldsymbol{x}$. Lastly, the number of latent components are set to $q = 3p/4 = 3$ and $q = 6$, respectively. For these test functions, we compare LCGP, LCGP_robust, OILMM, and SVGP, since these methods are designed to tackle heterogeneous errors. GPPCA is thereby dropped from comparisons.

The average time in training the emulators is presented in Table 3.2 to show the effects of output dimension and number of training inputs. The training time for LCGP and LCGP_robust remain generally low across the board, occasionally slower than SVGP. OILMM, behaving similarly to the illustrative function, takes the longest. Training times generally increase with n and p.

	p	4		8	
	n	100	500	100	500
Test Function	Method				
Borehole	LCGP	14.4	124.5	45.6	623.9
	LCGP_robust	25.9	205.7	39.3	573.0
	OILMM	70.6	177.4	355.7	1060.1
	SVGP	39.7	319.3	25.5	397.1
OTL Circuit	LCGP	5.3	72.4	23.8	371.1
	LCGP_robust	5.5	73.6	13.3	168.5
	OILMM	198.0	520.0	399.5	1138.2
	SVGP	24.7	198.1	27.3	401.2
Wingweight	LCGP	7.5	67.2	26.0	225.4
	LCGP_robust	7.7	59.4	27.4	187.7
	OILMM	81.4	67.9	383.6	1068.5
	SVGP	24.9	224.3	28.6	399.0

Table 3.2. Mean training time (seconds) across emulator methods.

Figure 3.5 presents the tradeoff between accuracy and training time for each of the test functions. LCGP and LCGP_robust are in general the fastest to train, SVGP slower, and OILMM the slowest among the methods. OILMM achieves the lowest NRMSE in the wing weight and the OTL circuit function where LCGP_robust achieves the lowest in the borehole function. SVGP incurs higher NRMSEs in all three test functions. We note that LCGP and LCGP_robust behave very similarly across the test functions. Figure 3.6 presents the 95% empirical coverages for each of the test functions. For OTL circuit and wing weight functions, LCGP, LCGP_robust, and OILMM achieves close to 95% empirical coverage; for the borehole function, LCGP and LCGP_robust overcovers while OILMM achieves around 95% coverage. In all functions, SVGP significantly misses the nominal 95% coverage.

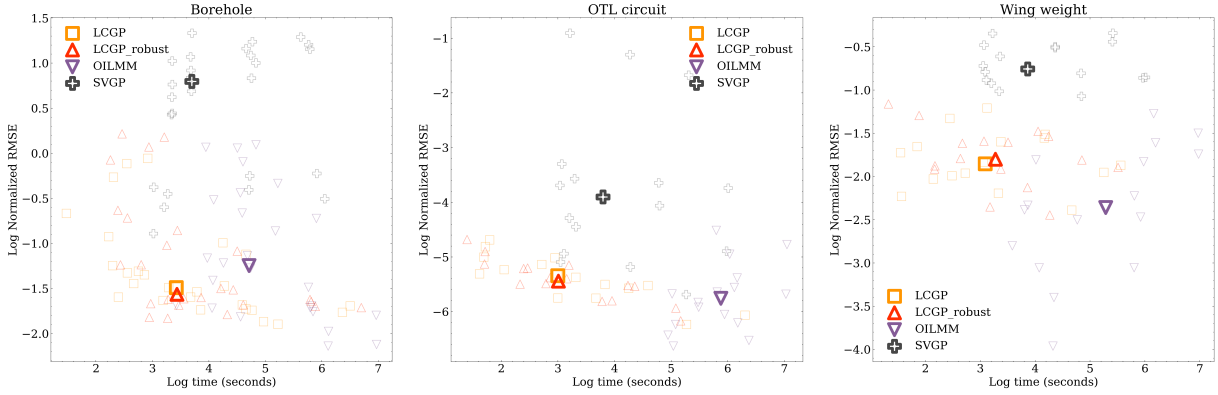


Figure 3.5. Accuracy comparisons for test functions with LCGP, OILMM, and SVGP. Log NRMSE are plotted against log training time for borehole (left), OTL circuit (center), and wing weight (right) functions. Median statistics are plotted with bold markers.

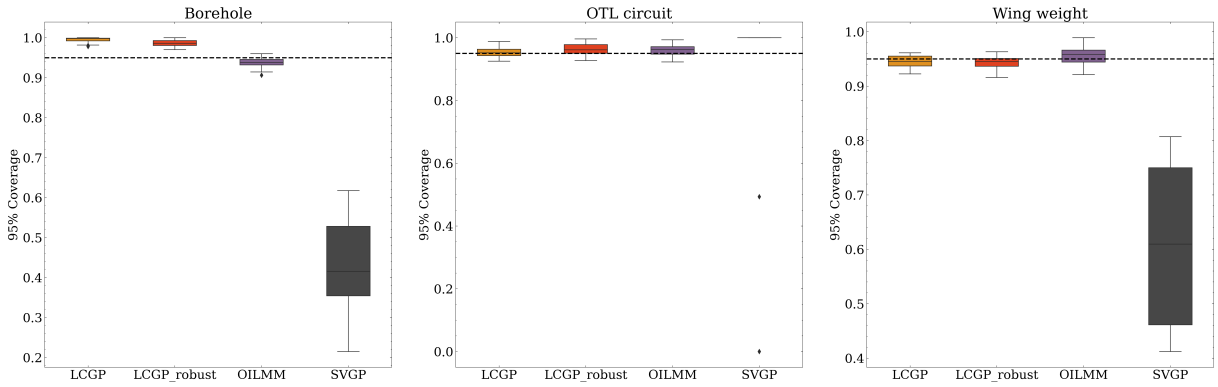


Figure 3.6. Coverage comparisons for test functions with LCGP, OILMM, and SVGP. Boxplots of empirical coverages are plotted for borehole (left), OTL circuit (center), and wing weight (right) functions. The coverage boxplot for SVGP for OTL circuit is concentrated around 1.0 with two runs at 0 and 0.5 respectively.

3.6. Viscous anisotropic hydrodynamic simulation model

The viscous anisotropic hydrodynamic (VAH) model is developed to improve the simulation of relativistic heavy-ion collisions [115]. To perform Bayesian calibration of the model parameters, the use of an emulator is required due to the expensive computations of the simulation model [42, 106]. The VAH model takes a input of 15 dimensions and returns an output with 98 locations, each corresponding to a measurement. A list of the inputs and their descriptions can be found in Everett et al. [42]. The 98 locations in the output are categorized in nine groups, with each group sharing similar behaviors. The categorization of measurements motivates the grouping of errors, as introduced in Section 3.5.

In addition to LCGP, LCGP_robust, OILMM, and SVGP, the original method in Liyanage et al. [106], PCSK, is included in the analysis for benchmarking. PCSK combines stochastic kriging (SK) [7] and principal component (PC) GP [74] to construct an emulator

to handle stochastic multivariate output. Stochastic kriging requires replications at each input for estimating the variance. Ankenman et al. [7] suggest performing at least 10 replications at all inputs to ensure the estimates are useful. However, it becomes prohibitive when the number of inputs is large with a limited simulation budget.

The dataset from the VAH model consists of 300 inputs, originally generated using a Latin Hypercube scheme [114]. We performed a 5-fold cross validation on this dataset to compare the five methods, with 240 training inputs and 60 testing inputs for each fold. Following the emulator setup by Liyanage et al. [106], the number of latent components is set to 12.

Figure 3.7 shows the performance comparison among the five emulator methods for the VAH model. LCGP, LCGP_robust, and PCSK achieve the lowest NRMSE and D-S score, while OILMM achieves slightly higher NRMSE and D-S score. SVGP incurs errors and scores that are magnitudes higher than the other four methods. The training time required by LCGP, LCGP_robust, and PCSK are comparable, with PCSK being the fastest. OILMM, as in other cases, requires a long training time due to the joint estimation of the standardization, error covariance, and the GP hyperparameters. Lastly, LCGP and LCGP_robust readily achieves the nominal 95% coverage, OILMM results in a large range of empirical coverages, and PCSK undercovers consistently. SVGP results in poor coverage, because the error in predictions is high. The use of LCGP and its variants do not require replications, therefore can lower the need for simulation replications and achieve better emulator properties, with similar accuracy and a better coverage.

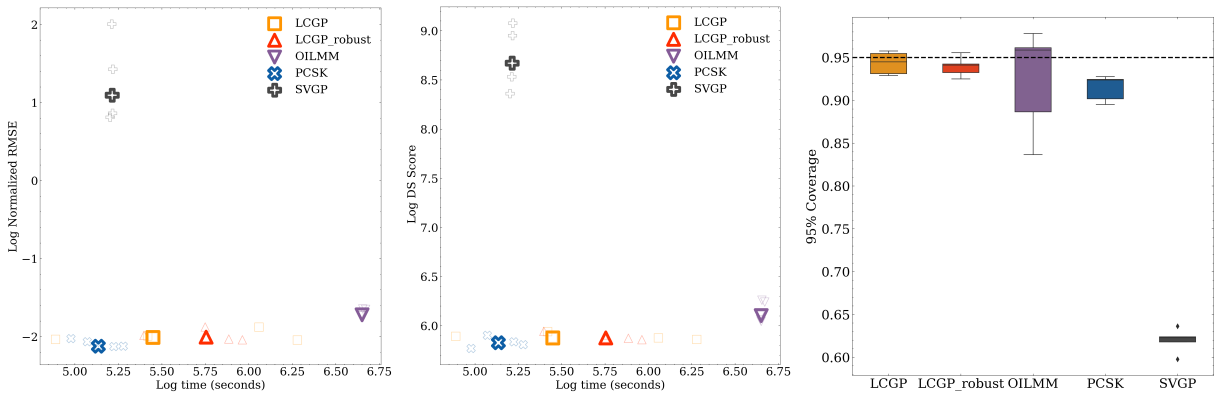


Figure 3.7. Summary performance measures for the VAH example with LCGP and other emulators. Log NRMSE (left) and log D-S score (center) are plotted against log training time. Median statistics are plotted with bold markers. The 95% empirical coverage (right) are presented in the boxplot and the nominal 95% level is indicated with the dashed line.

3.7. Discussion

The LCGP method is proposed to emulate noisy multivariate simulation outputs with a general error covariance structure. LCGP considers a heterogeneous error structure across outputs and provides exact inference for predictions. The method is founded upon modeling the uncertainty of the latent components. It enjoys the same computational benefits as any basis representation GP which can remain efficient for a relatively large number of outputs, as shown in the emulation of VAH model in Section 3.6. Since the basis representation model in Section 3.2.2.4 covers a rich class of methods, the enhancement in handling heterogeneous error can be readily applied. In this chapter, we have considered a heterogeneous diagonal error covariance. When a general positive definite error covariance is considered, the general forms of the prediction equations do not change. However, the log-likelihood function will not decouple neatly as in LCGP, and strategies in estimating

either a covariance or a precision matrix will be required, such as the estimators proposed by Lam [93].

CHAPTER 4

EMULATING HIGH-DIMENSIONAL SIMULATION FAILURES

As simulation models grow in complexity and sophistication, a simulation run typically takes longer. Recall that a simulation run refers to the process of providing an input to the simulation model and retrieving an output. Not only do simulation runs take longer, they can often produce inconsistent or unreasonable outputs, termed as simulation failures. Simulation failure is often tied to the input provided to a simulation run, meaning that there is certain failure structure governing the failure behaviors, even though the structure is unknown to the user. Using the taxonomical classification of such input-dependent structure (or constraints) in the simulation-based optimization literature, we are interested in the case of non-quantifiable, unrelaxable, simulation, and hidden (NUSH) constraints [98]. Briefly speaking, the NUSH category includes cases where the constraints are considered unknown a priori so that failure is only discovered by running the simulation model and the only information returned is whether the simulation has failed or not. In contrast, a quantifiable constraint may return information about the extent of constraint violation; and a relaxable constraint may provide a reasonable output even when the constraint is violated, whereas an unrelaxable constraint may cause the simulation run to crash.

The causes of simulation failures are intricate. Failures generally arise from categories such as the following:

(i) Outputs are undefined. The quantity of interest in the simulated system may be undefined given certain inputs. For example, simulation of storm surges produces a surge only when the surge height is higher than the ground elevation at that location [127]. Otherwise, no outputs are returned. In another example, inputs within certain region of the input space may cause the calculation of nuclei properties to crash, due to internal inconsistencies [21].

(ii) Outputs are non-convergent. Many simulation models involve iterative schemes to calculate an output, e.g., nuclear physics calculations such as [91] and [21], molecular dynamics [43], and steady-state estimation of system behaviors [71, ch.15, 16]. Depending on the type of iteration methods, solutions may not converge at the end of the simulation run and time out. The outputs in a non-convergent case can exhibit different behaviors of the quantity of interest.

(iii) Simulation runs may be interrupted by extraneous causes. Large-scale simulation runs nowadays are typically distributed among computational nodes. In the case that one node experiences a failure during a simulation run, no useful output may be returned. Modern simulation management systems, e.g., `libEnsemble` [79], are designed to accommodate various kinds of simulation failures.

The above failure categories are not exhaustive, nor are they mutually exclusive. What is clear is that simulation failures are common in some settings. Also, simulation failures are undesirable, causing wasted computational resources. The key intuition is that some of these simulation failures may be predictable and hence one may want to avoid initiating

that particular simulation run. In this chapter, we are primarily interested in failures from the first category where no outputs are returned. However, in the case where the cause of failures is unknown, the proposed method in this chapter is still applicable.

The ability to predict failures or to circumvent failures in simulation runs is particularly important in calibration, see e.g., earlier calibration of the Fayans density function in Chapter 2, where the goal is to identify a set of inputs that aligns simulation outputs to observations from the physical system in the real world. A sequential calibration procedure, sometimes called active learning, is commonly adopted where the space of input is high-dimensional [61, 33, 146, 138]. Another relevant area is simulation-based optimization, where the goal is to minimize an objective function and the objective can only be evaluated via simulation [100, 60, 10, 41, 99]. In both sequential calibration and simulation-based optimization, a set of new inputs are suggested for simulation runs after each iteration. In the case where the simulation model may fail, information about failures should be incorporated in suggesting new inputs.

Since the simulation models are expensive, an emulator is used in place of the simulation model. Emulators are constructed to provide fast predictions at unseen new inputs, along with uncertainty quantification for the predictions. In the case of simulation failures, an emulator should also represent the failure behavior of the simulation model. A common strategy is to construct two emulators, one for the successful simulation outputs and another for the simulation failures [100, 10, 99]. Then, the two emulators are used in conjunction to suggest new inputs. A Gaussian process emulator is a common choice for emulating the successful simulation outputs [100, 10, 99], but the choice for emulating failures varies. Specifically, the emulator for simulation failures should give a prediction

of the failure probability at the new inputs to guide the suggestions. For examples, Lee et al. [100] have chosen a random forest classifier and Bachoc et al. [10] have considered a Gaussian process classification model [163]. However, these methods consider univariate simulation outputs, and cannot be directly applied to high-dimensional simulation failures.

The task to emulate failures can be seen as a classification problem via the modeling of binary outputs. The classical model for classification follows a generalized linear model [112, 52]. A generalized linear model assumes the relationship $y = g^{-1}(\boldsymbol{X}\boldsymbol{\beta})$, where y is a response variable, $g(\cdot)$ is a link function, $\boldsymbol{X}$ is a set of inputs, and $\boldsymbol{\beta}$ is a set of coefficients to be estimated. In words, the generalized linear model assumes that through transforming the response variable with a link function, the transformed variable can be modeled with a linear model. When the response variable is binary, choosing the link function to be the logit link function yields the logistic regression [52, ch.6]. Examples that build upon the generalized linear model include Albert and Chib [2] and De Oliveira et al. [38], who have modeled binary outputs using an underlying latent variable along with the logit link function. To overcome the computational burden in parameter estimation that is introduced due to non-conjugacy between Bernoulli and Gaussian likelihoods, approximations are employed, see e.g., Breslow and Clayton [24]. However, they consider only univariate binary outputs.

In the case of high-dimensional failures, few options are available. By extending random forests to a multivariate response setting, Segal and Xiao [139] have considered generating random forests through resampling the data set. Zhang [165] has developed an extension of classification trees to multivariate binary outputs. Both methods require a large amount of data to be effective. In the case where simulation runs are costly, these methods are

impractical. Considering outside of fully nonparametric methods, Hung et al. [81] have considered a binary time series modeling of multivariate binary outputs, and Sung et al. [143] have considered a similar time series framework while introducing an additional GP to model the underlying probabilities for the binary outputs. However, the time-dependent autoregressive structure is not directly applicable to general high-dimensional simulation failures. One particularly effective option is proposed by Chang et al. [29], who have employed logistic principal component analysis (LPCA) [34, 102], a dimension reduction technique for multivariate binary outputs, and modeled the resulting principal components using a GP. However, similar to principal component analysis [83, ch.10], the LPCA method does not consider the inputs in the estimation of principal components. As a result, the principal components are generally non-smooth with respect to the inputs. This violates the intuition that the failure behavior is governed by some input-dependent constraints, which indicates that the failure behavior for nearby inputs may be similar.

In this chapter we focus on the task of emulating high-dimensional simulation failures as multivariate binary outputs. We propose a novel emulator method, termed as Logistic Principal Component Analysis GP (LPCAGP), that jointly estimates a set of logistic principal components and a multivariate GP that represents the failure probabilities. The proposed method improves Chang et al. [29] by estimating the principal components while simultaneously considering the relationship among inputs using the GP. LPCAGP achieves better failure predictions and provides uncertainty quantification corresponding to its predictions.

The remaining of the chapter is structured as follows: Section 4.1 reviews the LPCA algorithm [102] and GP inference. Section 4.2 presents the main results of LPCAGP, along

with the algorithm to estimate its parameters. Section 4.3 illustrates the effectiveness of LPCAGP using a synthetic failure generating function, highlighting its difference with Chang et al. [29]. Section 4.4 provides the implementation details and a set of numerical experiments to assess the performance of LPCAGP. Section 4.5 applies LPCAGP to emulating and predicting failures from the Fayans energy density functional, a nuclear physics model for understanding ground-state nuclei properties. Section 4.6 closes this chapter.

4.1. Preliminary background

This section first reviews the LPCA method by Lee et al. [102], then reviews a univariate GP and the basis representation GP formulation for modeling multivariate outputs.

4.1.1. Logistic principal component analysis. The logistic principal component analysis method is proposed as a dimension reduction technique for multivariate binary outputs [34, 102]. We present here the model assumption and the algorithm for parameter estimation proposed by Lee et al. [102], with adaptation of notations.

Suppose that $\boldsymbol{Y} = ((y_{ij})_{i=1}^{n})_{j=1}^{p}$ contains a set of multivariate binary outputs, where each row represents the output from one simulation run. Lee et al. [102] assumes that all elements of $\boldsymbol{Y}$ are independent, where each element y_{ij} follows a Bernoulli distribution with probability π_{ij}, and the logit link function is defined as

$$h_{ij} = \log\left(\pi_{ij}/(1-\pi_{ij})\right). \tag{4.1}$$

We call h_{ij}'s the latent variables. As a result, the probability mass function follows

$$p(y_{ij}) = \pi_{ij}^{y_{ij}}(1-\pi_{ij})^{(1-y_{ij})} = (1+\exp\{-(2y_{ij}-1)h_{ij}\})^{-1}. \tag{4.2}$$

Lee et al. [102] have proposed to model the latent variables as

$$\boldsymbol{H} = \mathbf{1}_n\boldsymbol{\beta}^\mathsf{T} + \boldsymbol{W}\boldsymbol{\Psi}^\mathsf{T}, \tag{4.3}$$

where $\mathbf{1}_n$ is the vector of n 1s, $\boldsymbol{\beta} \in \mathbb{R}^p$ is a mean vector, $\boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_q) \in \mathbb{R}^{n\times q}$ contains the principal components, and $\boldsymbol{\Psi} \in \mathbb{R}^{p\times q}$ is a transformation basis. Each latent variable h_{ij} is represented as $h_{ij} = \beta_j + \boldsymbol{W}_i\boldsymbol{\Psi}_j^\mathsf{T}$, where $\boldsymbol{W}_i$ is the ith row of $\boldsymbol{W}$, $\boldsymbol{\Psi}_j$ is the jth row of $\boldsymbol{\Psi}$. The log likelihood of $(\boldsymbol{\beta}, \boldsymbol{W}, \boldsymbol{\Psi})$ is given by

$$\ell(\boldsymbol{\beta}, \boldsymbol{W}, \boldsymbol{\Psi}; \boldsymbol{Y}) = \sum_{i=1}^{n}\sum_{j=1}^{p}\log p(y_{ij}) = \sum_{i=1}^{n}\sum_{j=1}^{p}\log\left(1+\exp\{-(2y_{ij}-1)h_{ij}\}\right)^{-1}. \tag{4.4}$$

To estimate the parameters, Lee et al. [102] have devised a majorization-minimization (MM) algorithm [82, 96, ch.8] to minimize the negative log-likelihood function. In Lee et al. [102], a penalized negative log-likelihood is considered to encourage sparsity in the matrix $\boldsymbol{\Psi}$. Here we only consider the original negative log-likelihood function. The MM algorithm proposes to find an objective that majorizes the negative log-likelihood function; and by iteratively constructing and minimizing this majorizing function, the MM algorithm allows each group of parameters to be optimized separately and leads to a tractable procedure [82]. We defer the details of the majorizing function until Section 4.2 as we use a similar construction in developing an MM algorithm. It is important to note that the resulting

majorizing function is quadratic in each of $\boldsymbol{\beta}, \boldsymbol{W}$, and $\boldsymbol{\Psi}$, which in turn admits closed-form solutions.

We summarize the MM algorithm adapted from Lee et al. [102] in Algorithm 1. We present a version that does not include the penalty term in the objective function. We note that throughout Algorithm 1, Lee et al. [102] have chosen to restrict $\boldsymbol{W}$, the principal components, to have orthonormal columns to avoid non-identifiability. In Section 4.2, specifically in the development of our MM algorithm, Algorithm 2, we compare the difference of the two MM algorithms.

Algorithm 1: MM algorithm (without penalty) [102]

Input: Binary simulation outputs $\boldsymbol{Y}$.
Output: Parameter estimates $(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{W}}, \widehat{\boldsymbol{\Psi}})$.
Set $m = 0$. Initialize by selecting $(\boldsymbol{\beta}^{(0)}, \boldsymbol{W}^{(0)}, \boldsymbol{\Psi}^{(0)})$.
while *convergence criteria are not met* **do**
  $\boldsymbol{H}^{(m)} = \mathbf{1}_n \boldsymbol{\beta}^{(m)\mathsf{T}} + \boldsymbol{W}^{(m)} \boldsymbol{\Psi}^{(m)\mathsf{T}}$.
  $\tilde{\boldsymbol{H}}^{(m)} = ((\tilde{h}_{ij}^{(m)})_{i=1}^{n})_{j=1}^{p}, \tilde{h}_{ij}^{(m)} = h_{ij}^{(m)} + 4(2y_{ij} - 1)\left(1 + \exp\{(2y_{ij} - 1)h_{ij}^{(m)}\}\right)^{-1}$.
  `/* Update mean vector */`
  $\boldsymbol{\beta}^{(m+1)} = n^{-1}\mathbf{1}_n^{\mathsf{T}}\left(\tilde{\boldsymbol{H}}^{(m)} - \boldsymbol{W}^{(m)}\boldsymbol{\Psi}^{(m)\mathsf{T}}\right)$.
  `/* Update principal components */`
  $\boldsymbol{W}^{(m+1)} = (\tilde{\boldsymbol{H}}^{(m)} - \mathbf{1}_n \boldsymbol{\beta}^{(m+1)\mathsf{T}})\boldsymbol{\Psi}^{(m)}(\boldsymbol{\Psi}^{(m)\mathsf{T}}\boldsymbol{\Psi}^{(m)})^{-1}$.
  Compute the QR decomposition $\boldsymbol{W}^{(m+1)} = \boldsymbol{Q}\boldsymbol{R}$ and set $\boldsymbol{W}^{(m+1)} = \boldsymbol{Q}$.
  `/* Update transformation basis */`
  $\boldsymbol{\Psi}^{(m+1)} = (\tilde{\boldsymbol{H}}^{(m)} - \mathbf{1}_n \boldsymbol{\beta}^{(m+1)\mathsf{T}})^{\mathsf{T}}\boldsymbol{W}^{(m+1)}$.
  $m = m + 1$.
end
Return $(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{W}}, \widehat{\boldsymbol{\Psi}}) = (\boldsymbol{\beta}^{(m)}, \boldsymbol{W}^{(m)}, \boldsymbol{\Psi}^{(m)})$.

4.1.2. Gaussian process background. A univariate GP assumes that any finite collection of its elements follows a multivariate normal distribution [130]. Define a real-valued vector as $\boldsymbol{w} = (w(\boldsymbol{x}_1), \ldots, w(\boldsymbol{x}_n))^{\mathsf{T}}$ evaluated at a set of inputs $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

To model the function $w(\cdot)$ with a GP means that $\boldsymbol{w} \sim \mathsf{N}(\boldsymbol{\mu}, \boldsymbol{C})$, where $\boldsymbol{\mu} = (\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_n))^\mathsf{T}$ is a mean vector, and $\boldsymbol{C}$ is a covariance matrix which behavior is governed by the $\boldsymbol{\theta}$-parameterized covariance function $c(\cdot, \cdot; \boldsymbol{\theta})$. The (i, j)-th entry of the covariance matrix is $C_{ij} = c(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta})$. We denote $\boldsymbol{\theta}$ as the set of GP hyperparameters. Some common choices of the covariance function include exponential, Gaussian, and the family of Matérn functions [130, 57], representing various degree of smoothness in the function. The choice of covariance function is left open in this exposition as it does not impact the theoretical development. The hyperparameters are estimated by minimizing the negative log-likelihood function, up to an additive constant, which is

$$\ell(\boldsymbol{\theta}; \boldsymbol{w}) = \frac{1}{2} \log |\boldsymbol{C}| + \frac{1}{2} \boldsymbol{w}^\mathsf{T} \boldsymbol{C}^{-1} \boldsymbol{w}, \tag{4.5}$$

where $|\boldsymbol{C}|$ is the determinant of $\boldsymbol{C}$. It is worth noting that estimating the hyperparameters involves the repeated evaluation of this log-likelihood and the typical computation is of $\mathcal{O}(n^3)$, due to the need for inverting $\boldsymbol{C}$ and calculating its determinant. For any unseen input $\boldsymbol{x}_0$, the prediction with a GP follows a normal distribution with mean and variance [130]

$$\widehat{\mu}(\boldsymbol{x}_0) = \mu(\boldsymbol{x}_0) + \boldsymbol{c}(\boldsymbol{x}_0)^\mathsf{T} \boldsymbol{C}^{-1} (\boldsymbol{w} - \boldsymbol{\mu}), \quad \widehat{\sigma}^2(\boldsymbol{x}_0) = c(\boldsymbol{x}_0, \boldsymbol{x}_0) - \boldsymbol{c}(\boldsymbol{x}_0)^\mathsf{T} \boldsymbol{C}^{-1} \boldsymbol{c}(\boldsymbol{x}_0), \tag{4.6}$$

where $\boldsymbol{c}(\boldsymbol{x}_0) = (c(\boldsymbol{x}_1, \boldsymbol{x}_0), \ldots, c(\boldsymbol{x}_n, \boldsymbol{x}_0))^\mathsf{T}$.

When GPs are employed in modeling multivariate outputs, say with output dimension p, the direct modeling using a GP is prohibitive as the number of data points grows to np, assuming the same number of inputs. As a result, the evaluation of the GP log-likelihood requires the inversion of the full $np \times np$ matrix, significantly slowing

down the estimation of hyperparameters. To extend the use of GPs to high-dimensional outputs, a general remedy is to impose structure over the covariance matrix of the GP, including convolution process [73, 32], separable covariance [133, 36, 18, 80, 48] and basis representation [14, 74, 76, 65, 25]. We proceed to describe the basis representation formulation for its relevance to our proposed method. For the full treatment of multivariate GPs, we refer the readers to Fricker et al. [48] and our earlier exposition in Section 3.2.2.

Consider then a p-dimensional output $\boldsymbol{h}(\boldsymbol{x})$, evaluated at an input $\boldsymbol{x}$. The orthonormal basis representation formulation is as follows,

$$\boldsymbol{h}(\boldsymbol{x}_i) = \boldsymbol{\beta} + \boldsymbol{\Psi}\boldsymbol{w}(\boldsymbol{x}_i), \quad \boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{\Psi} = \boldsymbol{I}_q, \tag{4.7}$$

$$w_k(\cdot) \overset{\text{ind}}{\sim} \mathsf{GP}(0, c(\cdot,\cdot;\boldsymbol{\theta}_k)), \quad k = 1, \ldots, q, \tag{4.8}$$

where $\boldsymbol{\beta} \in \mathbb{R}^p, \boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^{\mathsf{T}}$ is a mean vector, $\boldsymbol{\Psi} \in \mathbb{R}^{p\times q}, \boldsymbol{\Psi} = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_q), q \leq p$, is an orthonormal transformation basis, and $c(\cdot,\cdot;\boldsymbol{\theta}_k)$ is the covariance function for the kth GP component with hyperparameter $\boldsymbol{\theta}_k$. Comparing with the basis representation formulation in Equations (3.12–3.14), the mean vector $\boldsymbol{\beta}$ is introduced to represent the baseline failure probability (in the latent variable space). Collect the set of outputs as $\boldsymbol{H} = (\boldsymbol{h}(\boldsymbol{x}_1)^{\mathsf{T}}, \ldots, \boldsymbol{h}(\boldsymbol{x}_n)^{\mathsf{T}})^{\mathsf{T}}$, where the ith row of $\boldsymbol{H}$ represents a full output at $\boldsymbol{x}_i$. Denote the GP components as $\boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_q)$, where each column is $\boldsymbol{w}_k = (w_k(\boldsymbol{x}_1), \ldots, w_k(\boldsymbol{x}_n))^{\mathsf{T}}$. Lastly, the set of GP covariance hyperparameters is denoted as $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_q)$. Assuming the basis representation as fixed (i.e., fixed $\boldsymbol{\beta}$ and $\boldsymbol{\Psi}$), then the set of hyperparameters is

optimized by minimizing the negative log-likelihood for the basis representation GP,

$$\ell(\boldsymbol{\Theta}; \boldsymbol{W}) = \sum_{k=1}^{q} \left(\frac{1}{2} \log |\boldsymbol{C}_k| + \frac{1}{2} \boldsymbol{w}_k^{\mathsf{T}} \boldsymbol{C}_k^{-1} \boldsymbol{w}_k \right), \tag{4.9}$$

where $\boldsymbol{C}_k = (c(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}_k))$ is the covariance matrix for the kth GP component, parameterized by $\boldsymbol{\theta}_k$. Note that the decoupling of log-likelihood into a sum of q independent terms is a direct consequence of the orthonormality of $\boldsymbol{\Psi}$. See Section 3.3 for a general treatment of basis representation GP. The corresponding prediction $\boldsymbol{h}(\boldsymbol{x}_0)$ follows a multivariate normal distribution with mean and variance

$$\widehat{\boldsymbol{\mu}}(\boldsymbol{x}_0) = \boldsymbol{\beta} + \boldsymbol{\Psi}(\widehat{\mu}_1(\boldsymbol{x}_0), \ldots, \widehat{\mu}_q(\boldsymbol{x}_0))^{\mathsf{T}}, \quad \widehat{\boldsymbol{\Sigma}}(\boldsymbol{x}_0) = \boldsymbol{\Psi} \mathrm{diag}(\widehat{\sigma}_1^2(\boldsymbol{x}_0), \ldots, \widehat{\sigma}_q^2(\boldsymbol{x}_0)) \boldsymbol{\Psi}^{\mathsf{T}},$$

where $\widehat{\mu}_k(\boldsymbol{x}_0)$ and$\widehat{\sigma}_k^2(\boldsymbol{x}_0)$ are the predictive mean and variance respectively for the univariate kth GP (Equation (4.6)), and $\mathrm{diag}(a_1, \ldots, a_q)$ forms the diagonal matrix with elements $a_1, \ldots, a_q$.

4.2. Logistic principal component analysis Gaussian process

This section describes our proposed methodology, LPCAGP (Logistic Principal Component Analysis Gaussian Process), to emulate simulation failures as multivariate binary outputs. The two-component statistical model for the multivariate binary outputs is presented, including the modeling of binary variables and the latent variables.

4.2.1. Statistical modeling for multivariate binary outputs. Suppose $\boldsymbol{Y}(\boldsymbol{x}) \in \{0, 1\}^p$ represents a p-dimensional binary output evaluated at a d-dimensional input $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$. To model a specific $Y_j(\boldsymbol{x}_i)$, the jth response at input $\boldsymbol{x}_i$, we assume that each binary output is an independent Bernoulli random variable with probability

$\pi_{ij} = p(Y_j(\boldsymbol{x}_i) = 1)$. We follow Lee et al. [102] by choosing h_{ij} as the logit transformation of π_{ij} (Equation (4.1)). Consider $\boldsymbol{h}(\boldsymbol{x})$ being modeled using the orthonormal basis representation in Equation (4.7), then $\boldsymbol{h}(\boldsymbol{x}_i) = (h_1(\boldsymbol{x}_i), \ldots, h_p(\boldsymbol{x}_i))^{\mathsf{T}}$ corresponds to the latent variables $(h_{i1}, \ldots, h_{ip})^{\mathsf{T}}$. From the relationship imposed by the logit link function, a latent variable much greater than zero corresponds to a high probability for the binary variable to be 1; on the other hand, a latent variable much smaller than zero corresponds to a high probability for it to be 0.

To summarize, the full model for representing the binary multivariate outputs is as follows:

$$\boldsymbol{Y}(\boldsymbol{x}_i) \mid \boldsymbol{h}(\boldsymbol{x}_i) \sim \prod_{j=1}^{p} \text{Bernoulli}((1 + \exp\{-h_j(\boldsymbol{x}_i)\})^{-1}) \tag{4.10}$$

$$\boldsymbol{h}(\boldsymbol{x}_i) \mid \boldsymbol{\beta}, \boldsymbol{\Psi} = \boldsymbol{\beta} + \boldsymbol{\Psi}\boldsymbol{w}(\boldsymbol{x}_i), \quad \boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{\Psi} = \boldsymbol{I}_q, \tag{4.11}$$

$$w_k(\cdot) \mid \boldsymbol{\theta}_k \sim \mathsf{GP}(0, c(\cdot, \cdot; \boldsymbol{\theta}_k)), \quad k = 1, \ldots, q. \tag{4.12}$$

We note that Chang et al. [29] have considered modeling each column of the principal components as an independent GP, while fixing $\boldsymbol{\beta}, \boldsymbol{W}$, and $\boldsymbol{\Psi}$ as the results from Algorithm 1 [102]. In contrast, we consider jointly estimating all parameters, including the basis representation for the latent variables $(\boldsymbol{\beta}, \boldsymbol{W}, \boldsymbol{\Psi})$ and the covariance hyperparameters $\boldsymbol{\Theta}$.

The intuition to the full statistical model is this: the underlying latent variables that represent the probabilities smoothly transition from one input to its neighbors, meaning that the probabilities also behaves similarly. The modeling of latent variables as a Gaussian process serves as a flexible option to adapt to unknown latent variable behaviors

while maintaining certain smoothness structure through the covariance function and its hyperparameters.

4.2.2. Parameter estimation. Fitting the LPCAGP model requires estimating the basis representation for the latent variables and the GP hyperparameters. The full negative log-likelihood function of the parameters is obtained by combining the negative log-likelihood of the binary outputs in Equation (4.4) and those of GP components in Equation (4.9), up to an additive constant, i.e.,

$$\begin{aligned}
&\ell(\boldsymbol{\beta}, \boldsymbol{W}, \boldsymbol{\Psi}, \boldsymbol{\Theta}; \boldsymbol{X}, \boldsymbol{Y}) \\
&= -\sum_{i=1}^{n}\sum_{j=1}^{p} \log\left(1 + \exp\{-(2y_{ij}-1)(\beta_j + \boldsymbol{W}_i \boldsymbol{\Psi}_j^{\mathsf{T}}\}\right)^{-1} \\
&\quad + \sum_{k=1}^{q}\left(\frac{1}{2}\log|\boldsymbol{C}_k| + \frac{1}{2}\boldsymbol{w}_k^{\mathsf{T}}\boldsymbol{C}_k^{-1}\boldsymbol{w}_k\right).
\end{aligned} \tag{4.13}$$

4.2.2.1. *Majorizing function for the negative log-likelihood.* To estimate the parameters, the minimization of the negative log-likelihood is often performed using quasi-Newton methods such as L-BFGS [120, ch.7], which are effective in estimating GP hyperparameters [57, ch.5]. This minimization problem becomes challenging as the number of parameters increases, despite the structure exhibited in Equations (4.10–4.12). The number of parameters include $\boldsymbol{\beta} \in \mathbb{R}^p$, $\boldsymbol{W} \in \mathbb{R}^{n\times q}$, $\boldsymbol{\Psi} \in \mathbb{R}^{p\times q}$, and $\boldsymbol{\theta}_k \in \mathbb{R}^{d+1}, k = 1, \ldots, q$. In total, the number of parameters in this minimization problem is $p+(n+p+d+1)q$. For example, in the case study of Fayans density functional failures (Section 4.5), with a choice of $q = 3$ GP components, the total number of parameters is $198 + (500 + 198 + 13 + 1) \cdot 3 = 2{,}334$.

We develop an MM algorithm [82] to separate the set of parameters in optimization, first by introducing a majorizing function for the negative log-likelihood function.

DEFINITION 4.1 (Majorizing function). *Denote the majorizing function as* $\ell^+(\boldsymbol{\xi}; \boldsymbol{\xi}^{(m)})$, *where* $\boldsymbol{\xi} = (\boldsymbol{\beta}, \boldsymbol{W}, \boldsymbol{\Psi}, \boldsymbol{\Theta})$ *contains all parameters to be estimated, and* $\boldsymbol{\xi}^{(m)}$ *contains the parameters at the* m*th iterate. The function* $\ell^+(\boldsymbol{\xi}; \boldsymbol{\xi}^{(m)})$ *is said to majorize* $\ell(\boldsymbol{\xi})$ *at* $\boldsymbol{\xi}^{(m)}$ *if the following conditions hold:* $\ell(\boldsymbol{\xi}^{(m)}) = \ell^+(\boldsymbol{\xi}^{(m)}; \boldsymbol{\xi}^{(m)})$, *and* $\ell(\boldsymbol{\xi}) \leq \ell^+(\boldsymbol{\xi}; \boldsymbol{\xi}^{(m)})$, *for all* $\boldsymbol{\xi}$.

COROLLARY 4.1 (Descent property [82]). *If the majorizing function* $\ell^+(\boldsymbol{\xi}; \boldsymbol{\xi}^{(m)})$ *attains its minimum value at* $\boldsymbol{\xi}^{(m+1)}$, *then* $\ell(\boldsymbol{\xi}^{(m+1)}) \leq \ell(\boldsymbol{\xi}^{(m)})$.

PROOF. From the definition of the majorizing function, the following inequalities hold:

$$\ell(\boldsymbol{\xi}^{(m+1)}) \leq \ell^+(\boldsymbol{\xi}^{(m+1)}; \boldsymbol{\xi}^{(m)}) \leq \ell^+(\boldsymbol{\xi}^{(m)}; \boldsymbol{\xi}^{(m)}) \leq \ell(\boldsymbol{\xi}^{(m)}).$$

□

We refer to Hunter and Lange [82] for the convergence results of an MM algorithm to a local minimum.

THEOREM 4.1 (Majorizing function of the negative log-likelihood). *A majorizing function for the negative log-likelihood is*

$$\begin{aligned}\ell^+(\boldsymbol{\xi}; \boldsymbol{\xi}^{(m)}) = &\sum_{i=1}^{n}\sum_{j=1}^{p}\left[\frac{1}{8}(\beta_j + \boldsymbol{W}_i\boldsymbol{\Psi}_j^\mathsf{T} - \tilde{h}_{ij}^{(m)})^2 - \log(1 + \exp\{-(2y_{ij}-1)h_{ij}^{(m)}\})^{-1}\right] \\ &+ \sum_{k=1}^{q}\left(\frac{1}{2}\log|\boldsymbol{C}_k| + \frac{1}{2}\boldsymbol{w}_k^\mathsf{T}\boldsymbol{C}_k^{-1}\boldsymbol{w}_k\right), \end{aligned} \tag{4.14}$$

where $h_{ij}^{(m)} = \beta_j^{(m)} + \boldsymbol{W}_i^{(m)} \boldsymbol{\Psi}_j^{(m)\mathsf{T}}$, *and*

$$\tilde{h}_{ij}^{(m)} = h_{ij}^{(m)} + 4(2y_{ij} - 1) \left(1 + \exp\{(2y_{ij} - 1)h_{ij}^{(m)}\}\right)^{-1}.$$

PROOF. Consider $r(h) = (1 + \exp\{-zh\})^{-1}$ and $z = 2y - 1$. Then for any values h_1 and h_2,

$$\begin{aligned} -\log r(h_1) &\leq -\log r(h_2) - z(1 - r(h_2))(h_1 - h_2) + \frac{1}{8}(h_1 - h_2)^2 \\ &\leq -\log r(h_2) + \frac{1}{8}(h_1 - h_2 - 4zr(-h_2))^2. \end{aligned}$$

For the (i, j)th term in the binary log-likelihood term, substitute

$$h_1 = \beta_j + \boldsymbol{W}_i \boldsymbol{\Psi}_j^{\mathsf{T}}, \quad h_2 = h_{ij}^{(m)}.$$

□

With the majorizing function in Equation (4.14), we design the majorization-minimization algorithm in Algorithm 2 for minimizing the negative log-likelihood function.

We note the major differences between Algorithm 1 by Lee et al. [102] and our proposed algorithm. First and foremost, our proposed algorithm includes the covariance hyperparameters $\boldsymbol{\Theta}$ from introducing the basis representation GP to model the latent variables. The additional estimation of the covariance hyperparameters introduces significant computational costs, due to the inner minimization step within each iteration. Second, we choose to keep the transformation basis $\boldsymbol{\Psi}$ to be orthonormal, as opposed to the principal components, following the basis representation GP formulation in Equation (4.11).

Algorithm 2: MM algorithm for negative log-likelihood minimization

Input: Binary simulation outputs $\boldsymbol{Y}$, inputs $\boldsymbol{X}$, covariance function $c(\cdot,\cdot)$.

Output: Parameter estimates $(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{W}}, \widehat{\boldsymbol{\Psi}}, \widehat{\boldsymbol{\Theta}})$.

Set $m = 0$. Initialize by selecting $(\boldsymbol{\beta}^{(0)}, \boldsymbol{W}^{(0)}, \boldsymbol{\Psi}^{(0)}, \boldsymbol{\Theta}^{(0)})$.

while *convergence criteria are not met* **do**

  $\boldsymbol{H}^{(m)} = \mathbf{1}_n \boldsymbol{\beta}^{(m)\mathsf{T}} + \boldsymbol{W}^{(m)} \boldsymbol{\Psi}^{(m)\mathsf{T}}$.

  $\tilde{\boldsymbol{H}}^{(m)} = ((\tilde{h}_{ij}^{(m)})_{i=1}^{n})_{j=1}^{p}, \tilde{h}_{ij}^{(m)} = h_{ij}^{(m)} + 4(2y_{ij} - 1)\left(1 + \exp\{(2y_{ij} - 1)h_{ij}^{(m)}\}\right)^{-1}$.

  `/* Update mean vector */`

  $\boldsymbol{\beta}^{(m+1)} = n^{-1}\mathbf{1}_n^{\mathsf{T}}\left(\tilde{\boldsymbol{H}}^{(m)} - \boldsymbol{W}^{(m)}\boldsymbol{\Psi}^{(m)\mathsf{T}}\right)$.

  `/* Update transformation basis */`

  $\boldsymbol{\Psi}^{(m+1)} = \left(\tilde{\boldsymbol{H}}^{(m)} - \mathbf{1}_n\boldsymbol{\beta}^{(m)\mathsf{T}}\right)\boldsymbol{W}^{(m)}\left(\boldsymbol{W}^{(m)\mathsf{T}}\boldsymbol{W}^{(m)}\right)^{-1}$.

  for $k = 1, \ldots, q$ **do**

    `/* Update GP components and hyperparameters */`

    $\boldsymbol{w}_k^{(m+1)} = \frac{1}{4}\left(\boldsymbol{C}_k^{(m)^{-1}} + \frac{1}{4}\boldsymbol{I}_n\right)^{-1}\left(\tilde{\boldsymbol{H}}^{(m)} - \mathbf{1}_n\boldsymbol{\beta}^{(m)\mathsf{T}}\right)\boldsymbol{\psi}_k^{(m)}$.

    $\boldsymbol{\theta}_k^{(m+1)} = \arg\min\left\{\frac{1}{2}\boldsymbol{w}_k^{(m)\mathsf{T}}\boldsymbol{C}_k^{-1}\boldsymbol{w}_k^{(m)} + \frac{1}{2}\log|\boldsymbol{C}_k|\right\}$.

  end

  `/* Retain orthonormality of Ψ */`

  $\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{\Psi}^{(m+1)}$ by QR decomposition.

  Update $\boldsymbol{\Psi}^{(m+1)} = \boldsymbol{Q}, \boldsymbol{W}^{(m+1)} = \boldsymbol{R}\boldsymbol{W}^{(m+1)}$.

  $m = m + 1$.

end

Return $(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{W}}, \widehat{\boldsymbol{\Psi}}, \widehat{\boldsymbol{\Theta}}) = (\boldsymbol{\beta}^{(m)}, \boldsymbol{W}^{(m)}, \boldsymbol{\Psi}^{(m)}, \boldsymbol{\Theta}^{(m)})$.

4.2.2.2. *Computational details of Algorithm 2.* We present several details in Algorithm 2 that are critical for its computational effectiveness.

First, the initialization of parameters. We use the estimated basis representation parameters from LPCA [102] as our iterate for initialization, i.e., $(\boldsymbol{\beta}^{(0)}, \boldsymbol{W}^{(0)}, \boldsymbol{\Psi}^{(0)})$. Although a computationally more advantageous algorithm has been presented by Lee and Huang [101], the descent property is lost. We instead adopt the implementation by Landgraf [94] of the original sparse LPCA algorithm for numerical stability. The sparsity penalty in Lee et al.

[102] is set to 0, because we are not concerned with sparsity in $\boldsymbol{\Psi}$. The full implementation details are presented in Section 4.4. Instead of estimating the basis representation and the covariance hyperparameters together, Chang et al. [29] have constructed independent GPs for the principal components resulted from LPCA. As a result, the performance of emulating simulation failures is suboptimal, especially when the underlying probabilities are smooth in the input space. We include an illustration for this separation of optimization in Section 4.3.

Second, the update of parameters. We ensure that the negative log-likelihood function value decreases after each iteration. Each parameter update in Algorithm 2 is obtained by minimizing the convex quadratic majorizing function while fixing all but one of the parameter blocks at a time. As a result, cycling through all parameter blocks ensures overall decrease in the negative log-likelihood by the descent property in Corollary 4.1. Algorithm 2 states that all parameters are updated using the previous iterate, however in practice, each parameter block can be updated using the most recent iterate of other parameters [82]. The cyclic parameter update often leads to fewer iterations of the algorithm [82, 102].

Third, the maintenance of orthonormality of $\boldsymbol{\Psi}$. From the $\boldsymbol{\Psi}$ update step in Algorithm 2, it is not necessarily the case that the update returns an orthonormal $\boldsymbol{\Psi}^{(m+1)}$. We may obtain the QR decomposition $\boldsymbol{QR} = \boldsymbol{\Psi}^{(m+1)}$. The QR decomposition is unique when we ensure the diagonal elements of $\boldsymbol{R}$ are positive [70, ch.6]. Then we may update $\boldsymbol{\Psi}^{(m+1)} = \boldsymbol{Q}, \boldsymbol{W}^{(m+1)} = \boldsymbol{W}^{(m+1)}\boldsymbol{R}^{\mathsf{T}}$ to retain orthonormality. Since the product $\boldsymbol{W}^{(m+1)}{\boldsymbol{\Psi}^{(m+1)}}^{\mathsf{T}}$ remains unchanged, the descent property of the algorithm is preserved.

Finally, the optimization of GP hyperparameters. Within each iteration, the GP hyperparameters need to be optimized. Even though the independence among GP components

reduces the computational burden, the repeated evaluation of the GP log-likelihood in Equation (4.5) can become a bottleneck for this otherwise fast algorithm. In place of a full GP, we use a sparse GP [142, 129], or predictive process [12], to speed up the repeated evaluation of GP log-likelihood. A sparse GP replaces $\boldsymbol{C}$, the original GP covariance matrix among the set of inputs $\boldsymbol{X}$, with an approximation. A popular choice of approximation is the Nyström approximation [162, 130, ch.8]. The Nyström approximation has the form $\boldsymbol{C} \approx \boldsymbol{C}_{\boldsymbol{X},\boldsymbol{u}}\boldsymbol{C}_{\boldsymbol{u},\boldsymbol{u}}^{-1}\boldsymbol{C}_{\boldsymbol{X},\boldsymbol{u}}^{\mathsf{T}} + \Delta$, where $\boldsymbol{C}_{\boldsymbol{X},\boldsymbol{u}}$ is the cross-covariance matrix between $\boldsymbol{X}$ and a set of inducing points $\boldsymbol{u}$, $\boldsymbol{C}_{\boldsymbol{u},\boldsymbol{u}}$ is the covariance matrix among $\boldsymbol{u}$, and $\boldsymbol{\Delta}$ is a correctional diagonal factor [142, 129]. The required matrix inverse and determinant calculations of $\boldsymbol{C}$ can then be carried out via the Woodbury identities in Equations (3.1–3.2). The number of inducing points is typically chosen to be much smaller than the original number of inputs to alleviate the computational burden. One widely adopted strategy in selecting the inducing points is by minimizing a variational objective [152], while other strategies including sequential selections are still under active research [33, 155].

4.2.3. Prediction with LPCAGP. To predict the binary outputs at an unseen $\boldsymbol{x}_0$, we first predict the latent variables at $\boldsymbol{x}_0$ and transform the latent variables to the failure probability predictions. Let $\pi_j(\boldsymbol{x}_0) = (1 + \exp\{-h_j(\boldsymbol{x}_0)\})^{-1}, j = 1, \ldots, p$, be the failure probabilities at $\boldsymbol{x}_0$.

Definition 4.2 (Logit-normal distribution). *If a univariate Gaussian variable X has mean μ and variance σ^2, then $P = (1 + \exp\{-X\})^{-1}$ follows a logit-normal distribution* $\text{LogitN}(\mu, \sigma^2)$ *[1, 47] with parameters* (μ, σ^2).

The failure probabilities at a new input $\boldsymbol{x}_0$ is $(\pi_j(\boldsymbol{x}_0))_{j=1}^p$, where

$$\pi_j(\boldsymbol{x}_0) \sim \text{LogitN}(\mu_j(\boldsymbol{x}_0), \Sigma_j(\boldsymbol{x}_0)),$$
$$\mu_j(\boldsymbol{x}_0) = \widehat{\beta}_j + (\widehat{\mu}_1(\boldsymbol{x}_0), \ldots, \widehat{\mu}_q(\boldsymbol{x}_0))^\mathsf{T}\, \widehat{\boldsymbol{\Psi}}_j^\mathsf{T},$$
$$\Sigma_j(\boldsymbol{x}_0) = \widehat{\boldsymbol{\Psi}}_j^\mathsf{T} \text{diag}(\widehat{\sigma}_1^2(\boldsymbol{x}_0), \ldots, \widehat{\sigma}_q^2(\boldsymbol{x}_0)) \widehat{\boldsymbol{\Psi}}_j,$$

and $\widehat{\mu}_k(\boldsymbol{x}_0), \widehat{\sigma}_k^2(\boldsymbol{x}_0), k = 1, \ldots, q$, are GP predictive means and variances at $\boldsymbol{x}_0$ respectively.

While $\pi_j(\boldsymbol{x}_0)$ has a known distribution, its expected value and variance do not have a closed form [47]. Numerical integration is required to compute the moments of the probabilities. To provide an efficient prediction, the median of this random variable may be used as a point prediction for the probabilities. The median of $\pi_j(\boldsymbol{x}_0)$ [1, 47, 143] is

$$\widehat{p}_j(\boldsymbol{x}_0) = \left(1 + \exp\left\{-\mu_j(\boldsymbol{x}_0)\right\}\right)^{-1}.$$

To produce binary predictions, one typically assigns a dichotomous prediction, (see e.g., Chang et al. [29]), where $\widehat{Y}_j(\boldsymbol{x}_0) = 1, \text{if } \widehat{p}_j(\boldsymbol{x}_0) > 0.5$, and 0 otherwise.

Related to predictions, we consider two measures of predictive quality, misclassification error and area under an ROC curve (AUC) [69, 23]. Let the testing inputs be $\boldsymbol{x}_i', i = 1, \ldots, n'$, the misclassification error is defined as

$$\text{Misclassification error} = \frac{1}{n'p} \sum_{i=1}^{n'} \sum_{j=1}^{p} \mathbf{1}\{\widehat{Y}_j(\boldsymbol{x}_i') \neq Y_j(\boldsymbol{x}_i')\},$$

where $Y_j(\boldsymbol{x}_i')$'s are testing samples from the testing inputs. The ROC curve, or the receiver-operator-characteristic curve, summarizes the trade-off between true and false positives [3]. The ROC curve is produced by plotting the true positive rate versus the false positive

rate, comparing the predictive probability $\widehat{p}_j(\boldsymbol{x}_0)$ with a classification threshold increasing from 0 to 1. The AUC is a scalar measure that captures the area under the ROC curve. The larger the AUC, the better the classification. The best possible value for AUC = 1.0, whereas a random classification has a AUC = 0.5.

4.3. Illustrative sine failure example

This section presents an illustration of the advantage of jointly optimizing the basis representation and the GP hyperparameters. We compare the emulator method proposed in Chang et al. [29], in which the principal components $\boldsymbol{W}$ are modeled by independent GPs while keeping the basis representation fixed. For distinction purpose, we refer to the method by Chang et al. [29] as "LPCA" and our proposed method as "LPCAGP."

Consider the following function as the ground truth probability for simulation failures,

$$p_j(x) = \begin{cases} 0, & \text{if } f_j(x) < 0 \\ 1, & \text{if } f_j(x) > 1 \\ f_j(x), & \text{otherwise} \end{cases}, \quad f_j(x) = x^6 + \sin\left(\frac{2\pi j}{p}\right), j = 1, \ldots, p. \tag{4.15}$$

The failure behavior depends on a sine function. A visualization of the behavior can be found in Figure 4.1.

A set of training data $\boldsymbol{Y} = ((Y_j(x_i))_{i=1}^n)_{j=1}^p$ is collected by sampling Bernoulli random variables with probabilities defined in Equation (4.15). The set of training inputs are chosen as $x_i = (i-1)/(n-1), i = 1, \ldots, n$, and the set of testing inputs are chosen as $x_i' = i/n, i = 1, \ldots, n'$. We set $n = 100, p = 50, n' = 800$ for this illustration. We consider the testing data to be nonrandom and $Y_j(x_i') = \mathbf{1}\{p_j(x_i') < 0.5\}$. This illustration

is repeated for 10 replications. The performance measures for LPCA and LPCAGP are presented in Table 4.1. LPCAGP uniformly performs better than LPCA in both lower misclassification error and higher AUC.

	Training		Testing	
	LPCAGP	LPCA	LPCAGP	LPCA
Misclassification error, $\times 10^{-2}$	3.4 (1.1)	6.3 (0.3)	3.2 (1.7)	5.6 (0.4)
AUC	0.997 (0.006)	0.982 (0.005)	0.997 (0.006)	0.986 (0.005)

Table 4.1. Average misclassification error (standard deviation) and average AUC (standard deviation) for the sine failure illustration are presented. LPCAGP uniformly achieves lower misclassification error and higher AUC than LPCA.

In Figure 4.1, we illustrate the differences between the joint parameter estimation versus only optimizing the GP covariance hyperparameters as in Chang et al. [29]. The sparse logistic PCA method [102] returns principal components that are highly non-smooth with respect to the inputs. Since LPCA takes the components as is when fitting their GPs, it inherits the non-smoothness in the components as well. However, when we jointly estimate the basis representation and covariance hyperparameters, smoother components can be obtained and hence improve the predictions.

4.4. Implementation and numerical comparisons

This section presents the implementation details of the proposed LPCAGP method. Numerical results are reported to highlight the strength of the method and identify scenarios where this method is suited for.

4.4.1. Implementation details. The proposed methodology is implemented in PyTorch [122], an open-source Python framework that allows streamlined development

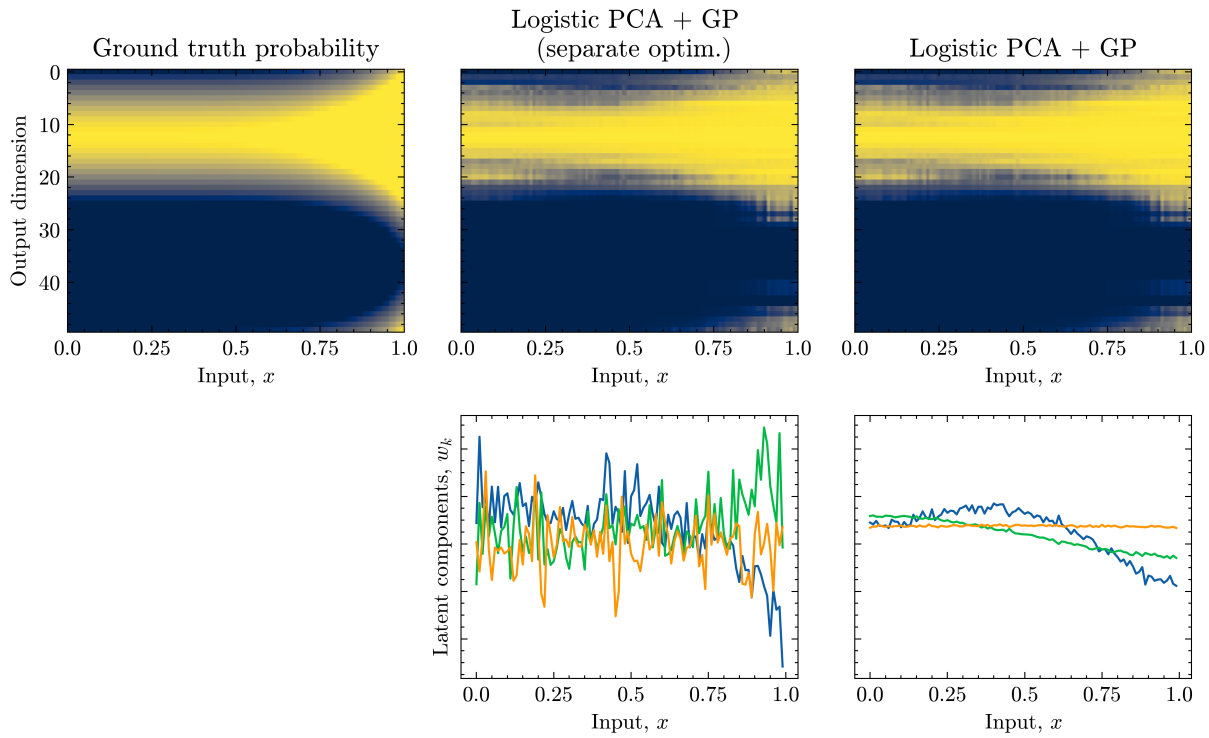


Figure 4.1. Comparison between separate and joint optimization of parameters. Top panel shows the ground truth probability, the predictive probability with LPCA, and the predictive probability with LPCAGP (left to right.) The bright yellow region corresponds to a high probability for $\pi_j(x)$, whereas the dark navy region corresponds to a low probability. Bottom panel shows the estimated GP components $\boldsymbol{w}_k, k = 1, 2, 3$, with LPCA and LPCAGP. The estimated latent components used in LPCA are non-smooth, as they directly result from the sparse logistic PCA method, whereas LPCAGP recovers smoother latent components with the joint optimization.

and deployment of statistical and machine learning methods. The main usage of PyTorch in the proposed method is in the optimization of GP hyperparameters. The estimation of GP hyperparameters is conducted using `PyTorch-LBFGS` [140], an implementation of the L-BFGS algorithm [120]. For the choice of GP covariance function, the exponential and the Matérn 3/2 covariance functions are available as choices. The default choice is the

exponential, following the understanding that there may not be sufficient information to infer the covariance hyperparameters in the Matérn function [29].

Regarding the choice of number of latent components, Lee et al. [102] have suggested using the Bayesian information criterion as a heuristic to determine the number of latent components to use, in the estimation of the initial basis representation. On the other hand, Landgraf and Lee [95] recommend choosing the smallest integer such that a chosen proportion based on the Bernoulli deviance is exceeded, analogous to the cumulative variance explained in a standard principal component analysis [83, ch.3]. In our approach, we set the default number of components to be 3, due to the inner optimization of GP hyperparameters. Furthermore, we have found satisfactory results with a smaller number of components as the joint optimization of the basis representation and the GP hyperparameters allows the components room to adapt given the number of components.

Lastly, the inputs are standardized into the unit hypercube for the input dimensions to be comparable, specifically in the setup of GP lengthscale hyperparameters. The latent components are standardized by their respective mean and variance before a GP is trained. This standardization step, or pre-scaling, is commonly used to simplify the specification of GP hyperparameters and/or improve numerical stability [76, 57].

4.4.2. Numerical results. This subsection presents the numerical experiments to assess the performance of LPCAGP. We consider two variants of LPCAGP, namely LPCAGP-MM and LPCAGP-LBFGS. The two variants assume the same statistical model in Equations (4.10–4.12), but employ different methods for minimizing the negative log-likelihood. LPCAGP-MM minimizes the negative log-likelihood function via the proposed Algorithm 2, whereas LPCAGP-LBFGS directly minimizes the negative log-likelihood

function in Equation (4.13). In addition to LPCA, we include a logistic regression model (LR) as a benchmark, as logistic regression is one of the standard classification methods [83, ch.5]. For each output location, an independent logistic regression model is fitted. The implementation of logistic regression model in SciPy [159] is employed. Two test functions are included, the first being the illustrative example introduced in Section 4.3, and the second being a circular failure example investigated by Chang et al. [29]. Similar to Section 4.3, we report the misclassification error and the AUC as the performance measures. We also investigate the extra training time required from the joint optimization. The setup and results for each test function are described separately.

4.4.2.1. *Illustrative sine failure example.* The same ground truth probability in Equation (4.15) is used for this experiment. We choose $n = 50, 100, 250, 500$ training points and $n' = 800$ testing points. We choose $p = 10, 25$. The number of components is set to $q = 3$ for all experiments with LPCA, LPCAGP-MM, and LPCAGP-LBFGS. The experiments are repeated for 15 replications. The results for case $p = 10$ are reported. Similar behavior is observed at $p = 25$.

Figure 4.2 shows the training and testing misclassification errors, and Figure 4.3 shows the training and testing AUC across n for each method. The poor performance of LR is expected, since independent logistic regression models assume no relationships across the output locations, while LPCA and LPCAGP variants model the relationship among outputs using the basis representation and latent variables. The LPCAGP variants perform consistently better than LPCA in both performance measures. The superior performance is attributed to the joint optimization of the basis representation and GP covariance

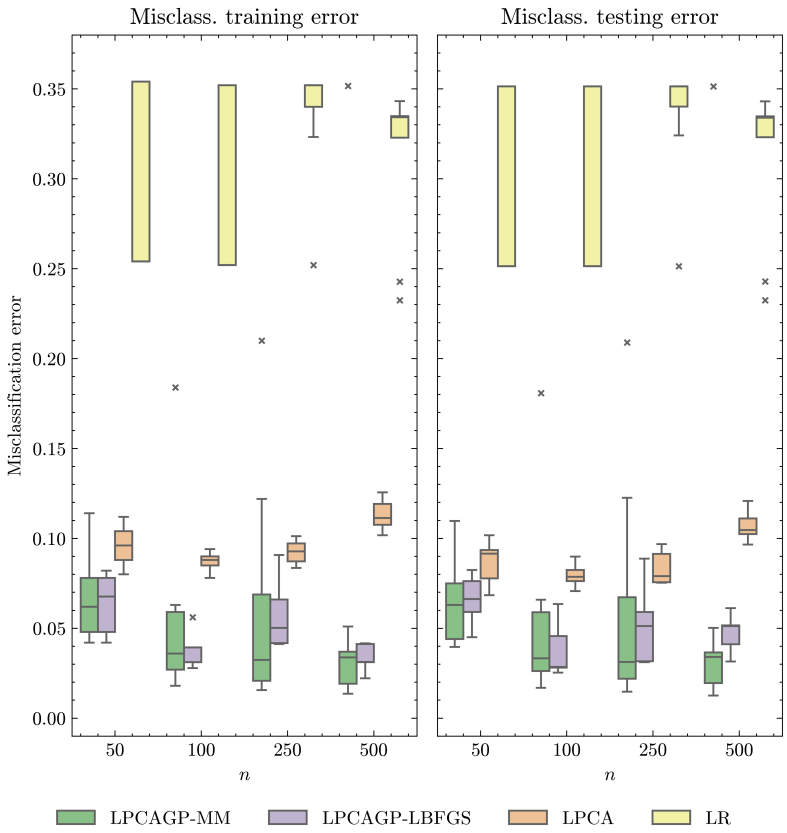


Figure 4.2. Misclassification error comparisons for the sine failure illustrative example. The boxplots of training (left) and testing (right) errors are plotted against number of training points, n. Output dimension $p = 10$. The LPCAGP variants achieve lowest misclassification errors across n, where LR produces the highest errors.

hyperparameters, which leads to relatively smooth principal components with respect to the inputs.

4.4.2.2. *Circular failure example.* Consider the function that exhibits circular failure behavior modified from Chang et al. [29], where

$$Y(\boldsymbol{x}, \boldsymbol{s}) = \begin{cases} 1, & \text{if } U(\boldsymbol{x}, \boldsymbol{s}) < 0 \\ 0, & \text{otherwise} \end{cases}, \quad U(\boldsymbol{x}, \boldsymbol{s}) = 1 - s_1^2 - \left(\frac{s_2}{1.5}\right)^2 - x_1 + x_2(s_2 + 1.5),$$

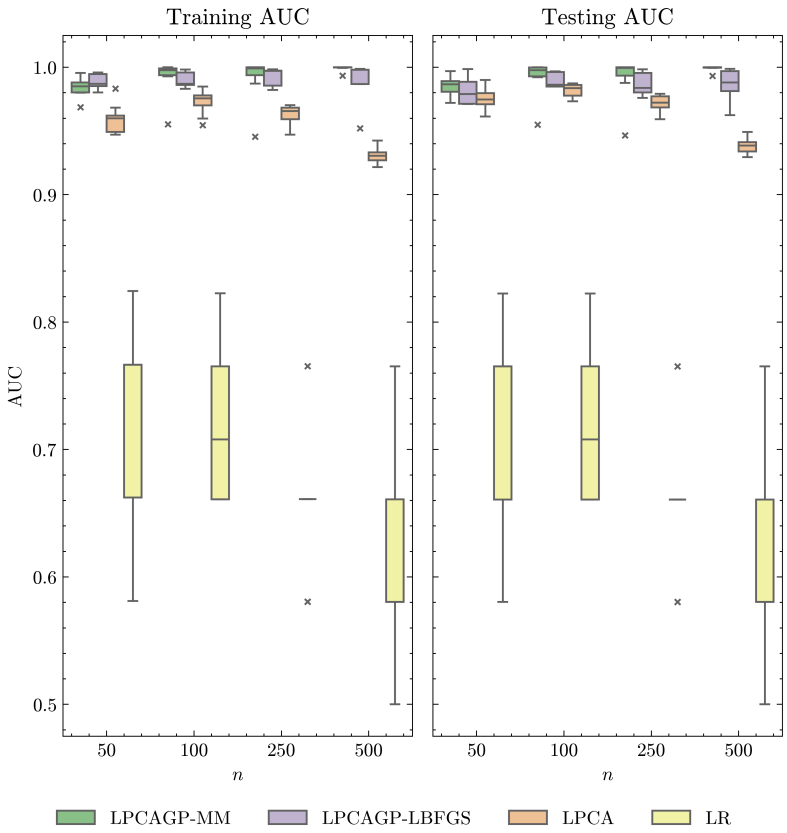


Figure 4.3. AUC comparisons for the sine failure illustrative example. The boxplots of training (left) and testing (right) errors are plotted against number of training points, n. Output dimension $p = 10$. Similarly, the LPCAGP variants achieve the highest AUC across n, where LR performs poorly with the lowest AUCs.

where $\boldsymbol{x} = (x_1, x_2), \boldsymbol{s} = (s_1, s_2)$. The ranges for the inputs are $x_1 \in (0.3, 0.65), x_2 \in (0, 0.2), s_1 \in (0, 1)$, and $s_2 \in (-1.5, 1.5)$. Since the expression $\sqrt{U(\boldsymbol{x}, \boldsymbol{s})}$ does not return a real number when $U(\boldsymbol{x}, \boldsymbol{s}) < 0$, it is considered a simulation failure, i.e., $Y(\boldsymbol{x}, \boldsymbol{s}) = 1$. One major distinction between the circular failure example and the sine failure example is that the cutoff between failure and non-failure is sharp in the circular failure example. Therefore, we expect the LPCAGP variants to have a smaller advantage over LPCA as the underlying probabilities do not transition smoothly around the failure boundary.

In this experiment, we consider training points of size $n = 50, 100, 250, 500$, and testing points of size $n' = 800$, both uniformly sampled from the feasible range of $\boldsymbol{x}$. We fix the set of $\boldsymbol{s}_j, j = 1, \ldots, p$ to be a regular grid formed by the Cartesian product of 20 evenly spaced points in s_1 and s_2 respectively. Therefore, the output dimension is $p = 20 \cdot 20 = 400$. Four examples of the outputs and their corresponding predictive probabilities on the regular grid are visualized in Figure 4.4 for demonstrating the impact of $\boldsymbol{x}$ on the extent of failures within the regular grid. The predictive probabilities can be seen to transition across the failure boundary, as the underlying GP structure enforces certain degree of smoothness in the latent variables.

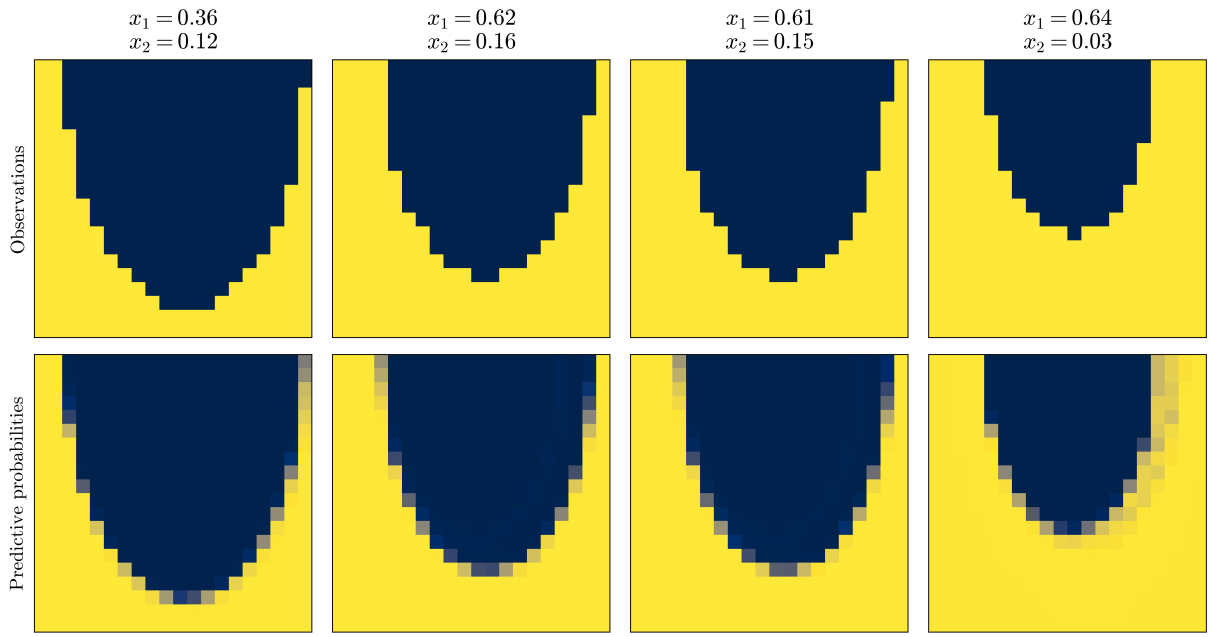


Figure 4.4. Circular failure observations (top) and LPCAGP-MM predictive probabilities (bottom) at four sample inputs. The bright yellow region corresponds to a high probability for $\pi_j(\boldsymbol{x})$, whereas the dark navy region corresponds to a low probability.

	Training error, $\times 10^{-2}$			
n	LPCAGP-MM	LPCAGP-LBFGS	LPCA	LR
50	0.67 (0.02)	0.69 (0.02)	0.35 (0.01)	19.17 (< 0.01)
100	0.73 (0.04)	0.59 (0.10)	0.65 (0.02)	19.49 (< 0.01)
250	0.69 (0.08)	0.62 (0.02)	0.60 (0.10)	20.18 (< 0.01)
500	0.72 (0.07)	0.48 (0.13)	0.83 (0.11)	21.91 (< 0.01)
	Testing error, $\times 10^{-2}$			
n	LPCAGP-MM	LPCAGP-LBFGS	LPCA	LR
50	1.15 (0.01)	1.40 (0.13)	0.87 (0.01)	20.04 (< 0.01)
100	0.74 (0.01)	0.82 (0.02)	0.76 (0.01)	19.81 (< 0.01)
250	0.89 (0.05)	0.81 (0.06)	0.74 (0.12)	21.06 (< 0.01)
500	0.83 (0.06)	0.87 (0.10)	0.90 (0.11)	21.04 (< 0.01)

Table 4.2. Misclassification error (standard deviation) comparisons for the circular failure example. The LPCAGP variants and LPCA achieve comparable error while LR achieves much higher error in both training and testing.

	Training AUC			
n	LPCAGP-MM	LPCAGP-LBFGS	LPCA	LR
50	0.9998	0.9967	1.0000	0.9628
100	0.9996	0.9975	1.0000	0.9678
250	0.9998	0.9989	0.9996	0.9558
500	0.9998	0.9997	0.9997	0.9504
	Testing AUC			
n	LPCAGP-MM	LPCAGP-LBFGS	LPCA	LR
50	0.9994	0.9961	0.9997	0.9606
100	0.9996	0.9962	0.9998	0.9649
250	0.9997	0.9987	0.9996	0.9548
500	0.9997	0.9994	0.9997	0.9510

Table 4.3. AUC comparisons for the circular failure example. LPCAGP-MM and LPCA achieve near-perfect and similar performance in terms of AUC, consistently better than LR across n. LPCAGP-LBFGS seems to achieve slightly lower AUC when n is smaller, and achieves comparable AUC with LPCA and LPCAGP-MM at $n = 500$. The standard deviation over replications of any AUC measure is < 0.001.

Table 4.2 presents the misclassification error and Table 4.3 presents the AUC across n for each method. The LPCAGP variants and LPCA achieve comparable performance with

a misclassification error rate less than 1% in most cases. In the case where the probability of failure changes sharply, in this case from 0 to 1 at the failure boundary, we observe no notable improvement using LPCAGP over LPCA. Similar to the sine failure example, LR performs significantly worse than LPCA and LPCAGP variants, as the assumption of independence across output locations is unreasonable. One noteworthy observation is that, even when the misclassification error for LR is at $\approx 20\%$, the corresponding AUC still hovers at ≈ 0.96. This echoes the caution in using the AUC as a performance measure because of its inability to differentiate methods with moderate and good performances [16].

4.4.3. Computational time. While the performance of LPCAGP variants is laudable, the computational time required by the proposed method is also significantly longer than the other methods, due to the introduction of the inner optimization of covariance hyperparameters. Figure 4.5 shows a comparison of emulator construction times among all emulators. The LPCAGP variants certainly require longer construction times in general, and the timing results present a clear trade-off between improving failure predictions and computational efficiency. LPCAGP-LBFGS requires comparable construction time compared to LPCAGP-MM. However, the variation of construction times is much smaller. From Figure 4.5, the scaling of LPCAGP-LBFGS with increasing n seems to be favorable over LPCAGP-MM, but there is no conclusive differences between the two methods in this set of numerical experiments.

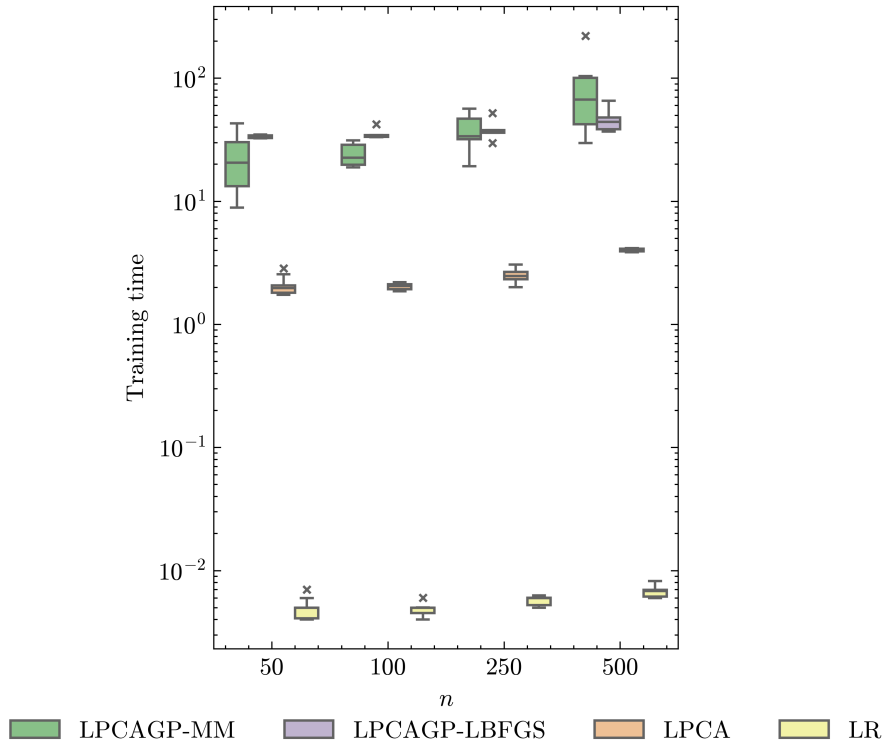


Figure 4.5. Construction time comparisons (in seconds) of LPCAGP-MM, LPCAGP-LBFGS, LPCA, and LR emulators for the sine failure example. The LPCAGP variants require the longest among all methods in emulator constructions. The use of LPCAGP-LBFGS seems to scale better than LPCAGP-MM as n increases, but there is not a clear distinction in this numerical test.

4.5. Emulating failures from the Fayans energy density functional

This section presents the application of LPCAGP to the failures encountered by the Fayans energy density functional [44, 45], for which PCGPwM is developed to emulate its outputs in Chapter 2.

The Fayans energy density functional (EDF) is developed for understanding the behavior of atomic nuclei, through calculating their ground-state properties [44, 45, 131]. The Fayans EDF takes a 13-dimensional input, and produces a 198-dimensional output.

Each response is characterized by an atomic nucleus configuration (the number of proton and neutron) and the type of observable. There are 72 unique configurations and 9 observable types that correspond to the 198 output locations. The full description of the numerical implementation can be found in Bollapragada et al. [21].

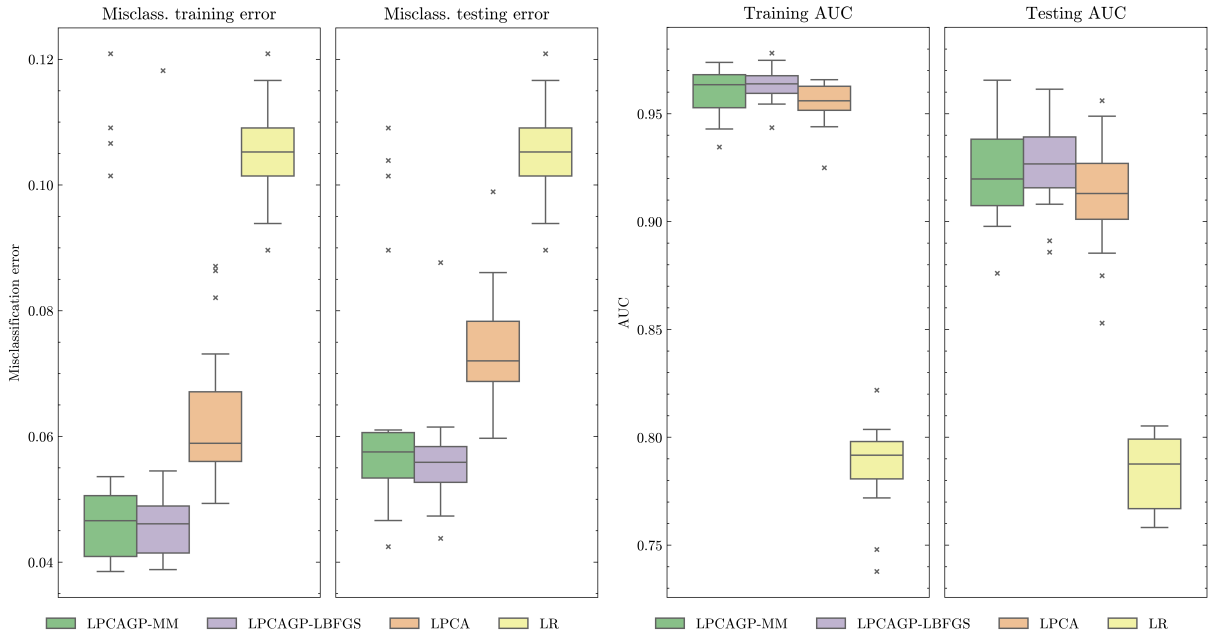
The Fayans EDF can fail and the failures can occur in up to 198 locations in any output. Two categories of failures are identified by Bollapragada et al. [21], which fall under the failures due to non-convergent solutions and undefined outputs. In their work of calibrating the Fayans EDF, Bollapragada et al. [21] implemented modifications of optimization algorithms such that encounters of failures are gracefully handled. With the same EDF, Chan et al. [28] have proposed a GP emulator strategy that handles failures as missing data to facilitate Bayesian calibration. However, the resulting emulator does not inform if the EDF is prone to fail for certain input region. We construct an emulator for these failures using LPCAGP to bridge this gap. We note that Bollapragada et al. [21] have designed specific treatments for each category of failures. Here, we consider only the failures due to undefined outputs as failures.

We apply the same methods, i.e., LPCAGP-MM, LPCAGP-LBFGS, LPCA, and LR, to construct binary emulators for the simulation failures in Fayans EDF. The data set consists of simulation outputs collected at 1,000 well-spaced inputs. We retrieve the simulation failures by assigning 1 to the failures due to undefined outputs, and assigning 0 to the rest. The overall proportion of failures is $\approx 10\%$, where simulation runs at more than two-thirds of inputs encounter at least 1 failure. The failure pattern can be visualized in the first panel of Figure 4.7, where the inputs are ordered by the number of failures encountered for easier visualization.

All inputs reside close to a local minimizer as a result of the point calibration [21], and the inputs have previously been scaled to a unit hypercube. The unscaled input for the Fayans EDF may be recovered by using the centroid of the unscaled hypercube and the scale for each input dimension, which are presented in Table 5 of Bollapragada et al. [21]. The 13-dimensional input poses challenges in producing a numerically stable emulator. We employ an active subspace method via a linear embedding approach [35], i.e., finding an orthogonal $\boldsymbol{A} \in \mathbb{R}^{d \times d'}, d' < d$, to project a d-dimensional input $\boldsymbol{x}$ to a d'-dimensional $\boldsymbol{Ax}$. By investigating the eigenvalues of the estimated expected outer product of the gradient, we find that $d' = 2$ is sufficient for retaining the important directions to represent the inputs. The construction of an emulator is straightforward by using the transformed inputs $\boldsymbol{Ax}$, and for any input $\boldsymbol{x}_0$, the same transformation can yield the reduced input for predictions.

The set of inputs and outputs is randomly split into a training and a testing set of 500 each. The training inputs are first transformed using the active subspace approach and used in the construction of an emulator. Using the predictions from the emulator, the misclassification errors and AUCs on both the training and testing data are recorded. We repeat this procedure for 10 replications.

Figure 4.6 presents the misclassification errors and AUCs for the training and testing data across replications. LPCAGP variants perform best among the emulator methods where LR performs the worst. In fact, LR achieves misclassification errors close to the proportion of failures in the data, meaning that LR does no better than assigning all predictions to 0, which is the majority of the data. As described by Bollapragada et al. [21], failures due to undefined outputs are produced due to the inputs being outside a



(a) Misclassification error (b) AUC

Figure 4.6. Performance comparisons for the Fayans EDF failures. Training and testing misclassification errors (left) and Training and testing AUCs (right) are presented over 10 replications. LPCAGP variants achieve best performance among the emulator methods in both misclassification errors and AUCs.

region of stability. It is likely that the inputs closer to the boundary of the region of stability are more prone to encounter failures, i.e., higher probability of failures. This structural characteristic aligns with the assumption of a smooth transition of probabilities in the input space, and the superior performance of LPCAGP shows evidence of such structure for the Fayans EDF model.

4.6. Concluding remarks

In this chapter, we have demonstrated that a multivariate binary emulator can be constructed to predict simulation failures. The proposed LPCAGP method utilizes the

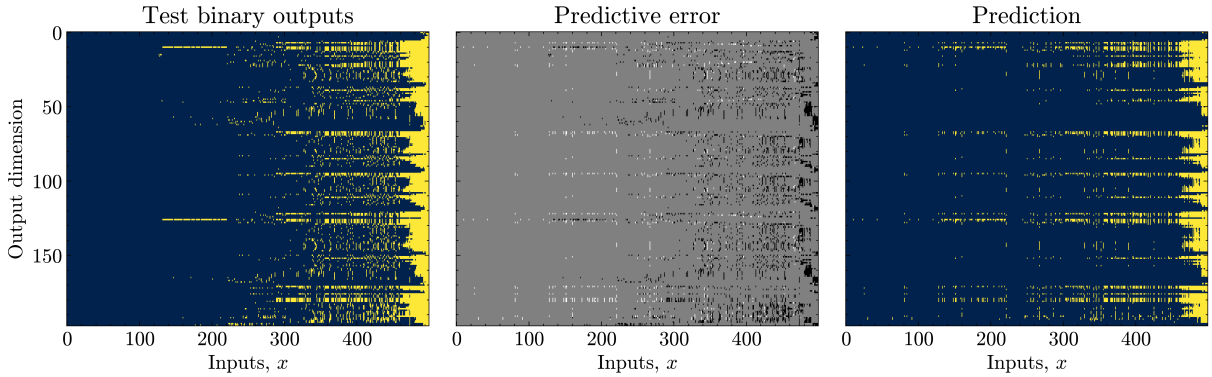


Figure 4.7. Prediction visualization with LPCAGP-MM for the Fayans EDF failures. Testing set binary outputs (left), prediction error (middle) and predictions (right) are plotted, where the inputs are ordered by the number of failures encountered, increasing from left to right for visualization purpose. In the middle panel, a bright spot indicates that the emulator predicted failure where there is not, whereas a dark spot indicates the opposite.

joint estimation of sparse logistic principal components and a multivariate GP for modeling the underlying failure probabilities. Under situations where the failure probabilities exhibit some smooth behavior, the proposed method excels among other binary classification methods.

One straightforward usage of LPCAGP is in facilitating Bayesian calibration of a simulation model that encounters input-dependent failures. An LPCAGP emulator can serve as a guide toward a region of inputs where failures are unlikely. Marcy and Storlie [110] have proposed a Bayesian calibration scheme with a truncated binary random field where a latent GP process describes the failure mechanism. The computational speed of LPCAGP variants is slower than other methods, given the joint estimation of parameters. The slower construction of LPCAGP does not create an issue when a one-off calibration is performed. However, if the use of LPCAGP is considered in a sequential setting where an emulator is repeatedly built given new data, then modifications should be made. One

modification is to replace the inner-optimization of GP hyperparameters with a combination of optimization-free GPs, e.g., [8]. Another possible modification is to use a different formulation for estimating the logistic principal components. Recently Landgraf and Lee [95] have revisited the problem of extracting sparse logistic PCA [102] and have proposed a distinct formulation for representing the latent variables. Instead of $\boldsymbol{H} = \mathbf{1}_n\boldsymbol{\beta}^\mathsf{T} + \boldsymbol{W}\boldsymbol{\Psi}^\mathsf{T}$ in Equation (4.3), Landgraf and Lee [95] suggest to use

$$\boldsymbol{H} = \mathbf{1}_n\boldsymbol{\beta}^\mathsf{T} + (\boldsymbol{Y}' - \mathbf{1}_n\boldsymbol{\beta}^\mathsf{T})\boldsymbol{\Phi}\boldsymbol{\Phi}^\mathsf{T},$$

where $\boldsymbol{Y}' = (L(2y_{ij} - 1)), L$ being a large number for approximating a set of "natural parameters", which serve a similar purpose of the latent variables in LPCAGP. The estimation is similarly set up by minimizing the negative log-likelihood function. One advantage of this formulation is its simple update given new data [95]. This is potentially of great use under a sequential Bayesian calibration scheme where data are collected in batches. In this setting, we may reduce the computations required in the parameter updates in the basis representation.

Finally, the estimation of principal components has extended beyond binary random variables [34, 105]. A similar construction such as the one used by LPCAGP can extend the analysis of multivariate binary outputs to outputs of general distributions, e.g., distributions in the exponential family.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTIONS

In this thesis we have addressed some general issues in constructing an effective emulator with high-dimensional simulation outputs. We have developed and analyzed new emulator strategies under various common scenarios, including missing data due to simulation failures, and unknown stochastic noise structure. The emulator strategies are built on the basis representation formulation to ensure computational efficiency.

In Chapter 2 we have proposed the PCGPwM method (Principal Component Gaussian Process with Missingness) for constructing an emulator without discarding any incomplete simulation outputs. The PCGPwM strategy is composed of an imputation strategy and an adjustment to the GP covariance matrix to account for the missing data. Numerical tests are administered to assess the performance of PCGPwM under various missing data scenarios. The PCGPwM emulator is applied in facilitating the calibration of the Fayans energy density functional, a nuclear physics simulation model that produces missing data when inputs are in a region of instability.

In Chapter 3 we have proposed LCGP (Latent Component Gaussian Process) to construct an emulator for noisy simulation outputs. LCGP extends current emulator strategies to incorporate heterogeneous noise structures. The method of LCGP learns the unknown error structure by modeling the stochasticity of its latent GP components. A key strength of LCGP is that it provides an exact inference and does not require an approximation. The method has shown promising results, both in prediction accuracy and

computation time, when set against other methods in a set of numerical experiments. The LCGP method is applied in a case study of emulating the viscous anisotropic hydrodynamics model, which describes the nuclear dynamics shortly after heavy-ion collisions.

In Chapter 4 we revisited the simulation failures that have caused the data missingness in Chapter 2. We have developed the method of LPCAGP (Logistic Principal Component Analysis Gaussian Process) to emulate failures directly as multivariate binary outputs. The proposed method jointly estimates the basis representation and a multivariate GP, with the GP modeling the unknown failure probabilities. We have presented two failure generating functions as examples and the results confirm that LPCAGP generally provides superior accuracy, especially when the underlying failure probabilities are smoothly transitioning. We have then applied LPCAGP to the simulation failures in the Fayans energy density functional directly.

There are yet many open avenues in studying GP emulator strategies and their usages. We close this thesis by mentioning several lines of possible future work. First, the performance of GP emulators is often impacted by several factors, namely the number of inputs, the dimension of the input, the dimension of the output, and the choice of the covariance function. Many efforts, including ours, seek to tackle one factor and improve on existing strategies. There is tremendous value in combining current strategies, providing a benchmark reference for the quality of methods (and combination of methods), and identifying suitable applicability under various assumptions. For example, Ober et al. [121] have provided some guidance in setting a baseline to benchmark approximate GPs. Second, the use of emulators is often in a sequential calibration setting. In such case, the emulator predictions and uncertainty measures guide the selection of the next set of inputs.

Often there is a balance among the batch size, the emulator construction time, and the simulation run time. Many have proposed strategies for selecting the next set of inputs [61, 33, 146], adapting an emulator toward the goal of calibration is less studied. We note that while GPs are used widely in scientific disciplines, the study for when GPs are appropriate and effective is less communicated. Much work remains in the development, application, and the communication of emulator strategies for increasingly challenging simulation settings.

References

[1] Aitchison, J. and Shen, S. M. (1980), "Logistic-Normal distributions: Some properties and uses," *Biometrika*, 67, 261–272.

[2] Albert, J. H. and Chib, S. (1993), "Bayesian analysis of binary and polychotomous response data," *Journal of the American Statistical Association*, 88, 669–679.

[3] Alonzo, T. A. and Pepe, M. S. (2002), "Distribution-free ROC analysis using binary regression techniques," *Biostatistics*, 3, 421–432.

[4] Altman, N. S. (1992), "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, 46, 175–185.

[5] Álvarez, M., Luengo, D., Titsias, M., and Lawrence, N. D. (2010), "Efficient multioutput Gaussian processes through variational inducing kernels," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, pp. 25–32.

[6] Andrianakis, I. and Challenor, P. G. (2012), "The effect of the nugget on Gaussian process emulators of computer models," *Computational Statistics & Data Analysis*, 56, 4215–4228.

[7] Ankenman, B., Nelson, B. L., and Staum, J. (2010), "Stochastic kriging for simulation metamodeling," *Operations Research*, 58, 371–382.

[8] Appriou, T., Rullière, D., and Gaudrie, D. (2023), "Combination of optimization-free kriging models for high-dimensional problems," *Computational Statistics*.

[9] Ba, S. and Joseph, V. R. (2012), "Composite Gaussian process models for emulating expensive functions," *The Annals of Applied Statistics*, 6, 1838–1860.

[10] Bachoc, F., Helbert, C., and Picheny, V. (2020), "Gaussian process optimization with failures: Classification and convergence proof," *Journal of Global Optimization*, 78, 483–506.

[11] Baker, E., Barbillon, P., Fadikar, A., Gramacy, R. B., Herbei, R., Higdon, D., Huang, J., Johnson, L. R., Ma, P., Mondal, A., et al. (2022), "Analyzing stochastic computer models: A review with opportunities," *Statistical Science*, 37, 64–89.

[12] Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. (2008), "Gaussian predictive process models for large spatial data sets," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70, 825–848.

[13] Bayarri, M. J., Berger, J. O., Kennedy, M. C., Kottas, A., Paulo, R., Sacks, J., Cafeo, J. A., Lin, C.-H., and Tu, J. (2009), "Predicting vehicle crashworthiness: Validation of computer models for functional and hierarchical data," *Journal of the American Statistical Association*, 104, 929–943.

[14] Bayarri, M. J., Walsh, D., Berger, J. O., Cafeo, J., Garcia-Donato, G., Liu, F., Palomo, J., Parthasarathy, R. J., Paulo, R., and Sacks, J. (2007), "Computer model validation with functional output," *The Annals of Statistics*, 35, 1874–1906.

[15] Ben-Ari, E. N. and Steinberg, D. M. (2007), "Modeling data from computer experiments: an empirical comparison of kriging with MARS and projection pursuit regression," *Quality Engineering*, 19, 327–338.

[16] Berrar, D. and Flach, P. (2012), "Caveats and pitfalls of ROC analysis in clinical microarray research (and how to avoid them)," *Briefings in Bioinformatics*, 13, 83–97.

[17] Bhattacharya, S. (2007), "A simulation approach to Bayesian emulation of complex dynamic computer models," *Bayesian Analysis*, 2, 783–815.

[18] Bilionis, I., Zabaras, N., Konomi, B. A., and Lin, G. (2013), "Multi-output separable Gaussian process: Towards an efficient, fully Bayesian paradigm for uncertainty quantification," *Journal of Computational Physics*, 241, 212–239.

[19] Binois, M., Gramacy, R. B., and Ludkovski, M. (2018), "Practical heteroscedastic Gaussian process modeling for large simulation experiments," *Journal of Computational and Graphical Statistics*, 27, 808–821.

[20] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017), "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, 112, 859–877.

[21] Bollapragada, R., Menickelly, M., Nazarewicz, W., O'Neal, J., Reinhard, P.-G., and Wild, S. M. (2021), "Optimization and supervised machine learning methods for fitting numerical physics models without derivatives," *Journal of Physics G: Nuclear*

and Particle Physics, 48, 024001.

[22] Bonilla, E. V., Chai, K., and Williams, C. (2007), "Multi-task Gaussian process prediction," *Advances in Neural Information Processing Systems*, 20.

[23] Bradley, A. P. (1997), "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, 30, 1145–1159.

[24] Breslow, N. E. and Clayton, D. G. (1993), "Approximate inference in generalized linear mixed models," *Journal of the American Statistical Association*, 88, 9–25.

[25] Bruinsma, W., Perim, E., Tebbutt, W., Hosking, S., Solin, A., and Turner, R. (2020), "Scalable exact inference in multi-output Gaussian processes," in *International Conference on Machine Learning*, PMLR, pp. 1190–1201.

[26] Brynjarsdóttir, J. and O'Hagan, A. (2014), "Learning about physical parameters: The importance of model discrepancy," *Inverse Problems*, 30, 114007.

[27] Cao, Y., Brubaker, M. A., Fleet, D. J., and Hertzmann, A. (2013), "Efficient optimization for sparse Gaussian process regression," *Advances in Neural Information Processing Systems*, 26.

[28] Chan, M. Y.-H., Plumlee, M., and Wild, S. M. (2023), "Constructing a simulation surrogate with partially observed output," *Technometrics*, to appear.

[29] Chang, W., Haran, M., Applegate, P., and Pollard, D. (2016), "Calibrating an ice sheet model using high-dimensional binary spatial data," *Journal of the American Statistical Association*, 111, 57–72.

[30] Chang, W., Haran, M., Olson, R., and Keller, K. (2014), "Fast dimension-reduced climate model calibration and the effect of data aggregation," *The Annals of Applied Statistics*, 8, 649–673.

[31] Chevalier, C. and Ginsbourger, D. (2013), "Fast computation of the multi-points expected improvement with applications in batch selection," in *Lecture Notes in Computer Science*, Heidelberg: Springer Berlin, pp. 59–69.

[32] Chung, S., Al Kontar, R., and Wu, Z. (2022), "Weakly supervised multi-output regression via correlated Gaussian processes," *INFORMS Journal on Data Science*, 1, 115–137.

[33] Cole, D. A., Christianson, R. B., and Gramacy, R. B. (2021), "Locally induced Gaussian processes for large-scale simulation experiments," *Statistics and Computing*,

31, 1–21.

[34] Collins, M., Dasgupta, S., and Schapire, R. E. (2002), "A generalization of principal components analysis to the exponential family," *Advances in Neural Information Processing Systems*, 14.

[35] Constantine, P. G., Dow, E., and Wang, Q. (2014), "Active subspace methods in theory and practice: Applications to kriging surfaces," *SIAM Journal on Scientific Computing*, 36, A1500–A1524.

[36] Conti, S. and O'Hagan, A. (2010), "Bayesian emulation of complex multi-output and dynamic computer models," *Journal of Statistical Planning and Inference*, 140, 640–651.

[37] Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D. (1991), "Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments," *Journal of the American Statistical Association*, 86, 953–963.

[38] De Oliveira, V., Kedem, B., and Short, D. A. (1997), "Bayesian prediction of transformed Gaussian random fields," *Journal of the American Statistical Association*, 92, 1422–1433.

[39] Dobaczewski, J., Nazarewicz, W., and Reinhard, P.-G. (2014), "Error estimates of theoretical models: A guide," *Journal of Physics G: Nuclear and Particle Physics*, 41, 074001.

[40] Dobaczewski, J. and Olbratowski, P. (2005), "Solution of the Skyrme-Hartree-Fock-Bogolyubov equations in the Cartesian deformed harmonic-oscillator basis. (V) HFODD (v2.08k)," *Computer Physics Communications*, 167, 214–216.

[41] Eriksson, D. and Poloczek, M. (2021), "Scalable constrained Bayesian optimization," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 730–738.

[42] Everett, D., Ke, W., Paquet, J.-F., Vujanovic, G., Bass, S. A., Du, L., Gale, C., Heffernan, M., Heinz, U., Liyanage, D., Luzum, M., Majumder, A., McNelis, M., Shen, C., Xu, Y., Angerami, A., Cao, S., Chen, Y., Coleman, J., Cunqueiro, L., Dai, T., Ehlers, R., Elfner, H., Fan, W., Fries, R. J., Garza, F., He, Y., Jacak, B. V., Jacobs, P. M., Jeon, S., Kim, B., Kordell, M., Kumar, A., Mak, S., Mulligan, J., Nattrass, C., Oliinychenko, D., Park, C., Putschke, J. H., Roland, G., Schenke, B., Schwiebert, L., Silva, A., Sirimanna, C., Soltz, R. A., Tachibana, Y., Wang, X.-N., and Wolpert, R. L. (2021), "Multisystem Bayesian constraints on the transport

coefficients of QCD matter," *Physical Review C*, 103, 054904.

[43] Fang, H., Gullett, P. M., Slepoy, A., Horstemeyer, M. F., Baskes, M. I., Wagner, G. J., and Li, M. (2004), "Numerical tools for atomistic simulations," Tech. rep., Sandia National Laboratories.

[44] Fayans, S. (1998), "Towards a universal nuclear density functional," *Journal of Experimental and Theoretical Physics Letters*, 68, 169–174.

[45] Fayans, S., Tolokonnikov, S., Trykov, E., and Zawischa, D. (2000), "Nuclear isotope shifts within the local energy-density functional approach," *Nuclear Physics A*, 676, 49–119.

[46] Forrester, A., Sobester, A., and Keane, A. (2008), *Engineering Design via Surrogate Modelling: A Practical Guide*, Chichester: John Wiley & Sons.

[47] Frederic, P. and Lad, F. (2008), "Two moments of the Logitnormal distribution," *Communications in Statistics - Simulation and Computation*, 37, 1263–1269.

[48] Fricker, T. E., Oakley, J. E., and Urban, N. M. (2013), "Multivariate Gaussian process emulators with nonseparable covariance structures," *Technometrics*, 55, 47–56.

[49] Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018), "GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration," *Advances in Neural Information Processing Systems*, 31.

[50] Gelfand, A. E., Schmidt, A. M., Banerjee, S., and Sirmans, C. (2004), "Nonstationary multivariate process modeling through spatially varying coregionalization," *Test*, 13, 263–312.

[51] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013), *Bayesian Data Analysis*, Boca Raton: CRC Press.

[52] Gelman, A. and Hill, J. (2006), *Data Analysis Using Regression and Multilevel/Hierarchical Models*, Cambridge University Press.

[53] Gentle, J. E. (2007), *Matrix Algebra*, New York: Springer New York.

[54] Geyer, C. J. (1991), "Markov chain Monte Carlo maximum likelihood," in *Computing Science and Statistics, Proceedings of the 23rd Symposium on the Interface*, pp. 156–163.

[55] Gneiting, T. and Raftery, A. E. (2007), "Strictly proper scoring rules, prediction, and estimation," *Journal of the American Statistical Association*, 102, 359–378.

[56] GPy (since 2012), "GPy: A Gaussian process framework in Python," `http://github.com/SheffieldML/GPy`.

[57] Gramacy, R. B. (2020), *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*, Boca Raton, Florida: Chapman Hall/CRC.

[58] Gramacy, R. B. and Apley, D. W. (2015), "Local Gaussian process approximation for large computer experiments," *Journal of Computational and Graphical Statistics*, 24, 561–578.

[59] Gramacy, R. B., Bingham, D., Holloway, J. P., Grosskopf, M. J., Kuranz, C. C., Rutter, E., Trantham, M., and Drake, R. P. (2015), "Calibrating a large computer experiment simulating radiative shock hydrodynamics," *The Annals of Applied Statistics*, 9, 1141–1168.

[60] Gramacy, R. B., Gray, G. A., Le Digabel, S., Lee, H. K., Ranjan, P., Wells, G., and Wild, S. M. (2016), "Modeling an augmented Lagrangian for blackbox constrained optimization," *Technometrics*, 58, 1–11.

[61] Gramacy, R. B. and Lee, H. K. (2009), "Adaptive design and analysis of supercomputer experiments," *Technometrics*, 51, 130–145.

[62] — (2012), "Cases for the nugget in modeling computer experiments," *Statistics and Computing*, 22, 713–722.

[63] Gramacy, R. B. and Lee, H. K. H. (2008), "Bayesian treed Gaussian process models with an application to computer modeling," *Journal of the American Statistical Association*, 103, 1119–1130.

[64] Gu, M. and Berger, J. O. (2016), "Parallel partial Gaussian process emulation for computer models with massive output," *The Annals of Applied Statistics*, 10, 1317–1347.

[65] Gu, M. and Shen, W. (2020), "Generalized probabilistic principal component analysis of correlated data." *Journal of Machine Learning Research*, 21, 1–41.

[66] Gu, M. and Xu, Y. (2020), "Fast nonseparable Gaussian stochastic process with application to methylation level interpolation," *Journal of Computational and Graphical Statistics*, 29, 250–260.

[67] Guillas, S., Sarri, A., Day, S., Liu, X., and Dias, F. (2018), "Functional emulation of high resolution tsunami modelling over Cascadia," *The Annals of Applied Statistics*, 12, 2023–2053.

[68] Handcock, M. S. and Stein, M. L. (1993), "A Bayesian analysis of kriging," *Technometrics*, 35, 403–410.

[69] Hanley, J. A. and McNeil, B. J. (1982), "The meaning and use of the area under a receiver operating characteristic (ROC) curve." *Radiology*, 143, 29–36.

[70] Harville, D. (2008), *Matrix Algebra From a Statistician's Perspective*, Springer New York.

[71] Henderson, S. G. and Nelson, B. L. (eds.) (2006), *Simulation*, Handbooks in Operations Research and Management Science, Amsterdam: Elsevier.

[72] Hensman, J., Durrande, N., Solin, A., et al. (2017), "Variational Fourier features for Gaussian processes." *Journal of Machine Learning Research*, 18, 5537–5588.

[73] Higdon, D. (2002), "Space and space-time modeling using process convolutions," in *Quantitative Methods for Current Environmental Issues*, London: Springer London, pp. 37–56.

[74] Higdon, D., Gattiker, J., Williams, B., and Rightley, M. (2008), "Computer model calibration using high-dimensional output," *Journal of the American Statistical Association*, 103, 570–583.

[75] Higdon, D., Kennedy, M., Cavendish, J. C., Cafeo, J. A., and Ryne, R. D. (2004), "Combining field data and computer simulations for calibration and prediction," *SIAM Journal on Scientific Computing*, 26, 448–466.

[76] Higdon, D., McDonnell, J. D., Schunck, N., Sarich, J., and Wild, S. M. (2015), "A Bayesian approach for parameter estimation and prediction using a computationally intensive model," *Journal of Physics G: Nuclear and Particle Physics*, 42, 034009.

[77] Hoffman, M. and Blei, D. (2015), "Stochastic structured variational inference," in *Artificial Intelligence and Statistics*, PMLR, pp. 361–369.

[78] Huang, J., Gramacy, R. B., Binois, M., and Libraschi, M. (2020), "On-site surrogates for large-scale calibration," *Applied Stochastic Models in Business and Industry*, 36, 283–304.

[79] Hudson, S., Larson, J., Navarro, J.-L., and Wild, S. M. (2021), "libEnsemble: A library to coordinate the concurrent evaluation of dynamic ensembles of calculations," *IEEE Transactions on Parallel and Distributed Systems*, 33, 977–988.

[80] Hung, Y., Joseph, V. R., and Melkote, S. N. (2015), "Analysis of computer experiments with functional response," *Technometrics*, 57, 35–44.

[81] Hung, Y., Zarnitsyna, V., Zhang, Y., Zhu, C., and Wu, C. J. (2008), "Binary time series modeling with application to adhesion frequency experiments," *Journal of the American Statistical Association*, 103, 1248–1259.

[82] Hunter, D. R. and Lange, K. (2004), "A tutorial on MM algorithms," *The American Statistician*, 58, 30–37.

[83] James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013), *An Introduction to Statistical Learning*, vol. 112, Springer.

[84] Joseph, V. R., Gul, E., and Ba, S. (2015), "Maximum projection designs for computer experiments," *Biometrika*, 102, 371–380.

[85] Joseph, V. R., Hung, Y., and Sudjianto, A. (2008), "Blind kriging: A new method for developing metamodels," *Journal of Mechanical Design*, 130, 031102.

[86] Katzfuss, M. and Guinness, J. (2021), "A general framework for Vecchia approximations of Gaussian processes," *Statistical Science*, 36, 124–141.

[87] Kennedy, M. C. and O'Hagan, A. (2001), "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63, 425–464.

[88] Kleijnen, J. P. (2017), "Regression and kriging metamodels with their experimental designs in simulation: A review," *European Journal of Operational Research*, 256, 1–16.

[89] Kortelainen, M., Lesinski, T., Moré, J. J., Nazarewicz, W., Sarich, J., Schunck, N., Stoitsov, M. V., and Wild, S. M. (2010), "Nuclear energy density optimization," *Physical Review C*, 82, 024313.

[90] Kortelainen, M., McDonnell, J. D., Nazarewicz, W., Olsen, E., Reinhard, P.-G., Sarich, J., Schunck, N., Wild, S. M., Davesne, D., Erler, J., and Pastore, A. (2014), "Nuclear energy density optimization: Shell structure," *Physical Review C*, 89, 054314.

[91] Kortelainen, M., McDonnell, J. D., Nazarewicz, W., Reinhard, P.-G., Sarich, J., Schunck, N., Stoitsov, M. V., and Wild, S. M. (2012), "Nuclear energy density optimization: Large deformations," *Physical Review C*, 85, 024304.

[92] Kyprioti, A. P., Taflanidis, A. A., Plumlee, M., Asher, T. G., Spiller, E., Luettich, R. A., Blanton, B., Kijewski-Correa, T. L., Kennedy, A., and Schmied, L. (2021), "Improvements in storm surge surrogate modeling for synthetic storm parameterization, node condition classification and implementation to small size databases," *Natural Hazards*, 109, 1349–1386.

[93] Lam, C. (2016), "Nonparametric eigenvalue-regularized precision or covariance matrix estimator," *The Annals of Statistics*, 44, 928–953.

[94] Landgraf, A. (2013), "SparseLogisticPCA," `https://github.com/andland/SparseLogisticPCA`.

[95] Landgraf, A. J. and Lee, Y. (2020), "Dimensionality reduction for binary data through the projection of natural parameters," *Journal of Multivariate Analysis*, 180, 104668.

[96] Lange, K. (2013), *Optimization*, Springer Texts in Statistics, Springer New York.

[97] Lawrence, E., Heitmann, K., Kwan, J., Upadhye, A., Bingham, D., Habib, S., Higdon, D., Pope, A., Finkel, H., and Frontiere, N. (2017), "The Mira-Titan universe. II. Matter power spectrum emulation," *The Astrophysical Journal*, 847, 50.

[98] Le Digabel, S. and Wild, S. M. (2023), "A taxonomy of constraints in black-box simulation-based optimization," *Optimization and Engineering*, 1–19.

[99] Lee, C., Wang, X., Wu, J., and Yue, X. (2022), "Failure-averse active learning for physics-constrained systems," *IEEE Transactions on Automation Science and Engineering*.

[100] Lee, H., Gramacy, R., Linkletter, C., and Gray, G. (2011), "Optimization subject to hidden constraints via statistical emulation," *Pacific Journal of Optimization*, 7, 467–478.

[101] Lee, S. and Huang, J. Z. (2013), "A coordinate descent MM algorithm for fast computation of sparse logistic PCA," *Computational Statistics & Data Analysis*, 62, 26–38.

[102] Lee, S., Huang, J. Z., and Hu, J. (2010), "Sparse logistic principal components analysis for binary data," *The Annals of Applied Statistics*, 4, 1579.

[103] Lin, L., Bingham, D., Broekgaarden, F., and Mandel, I. (2021), "Uncertainty quantification of a computer model for binary black hole formation," *The Annals of Applied Statistics*, 15, 1604–1627.

[104] Liu, H., Ong, Y.-S., Shen, X., and Cai, J. (2020), "When Gaussian process meets big data: A review of scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, 31, 4405–4423.

[105] Liu, L. T., Dobriban, E., and Singer, A. (2018), "*e*PCA: High dimensional exponential family PCA," *The Annals of Applied Statistics*, 12, 2121–2150.

[106] Liyanage, D., Sürer, Ö., Plumlee, M., Wild, S. M., and Heinz, U. (2023), "Bayesian calibration of viscous anisotropic hydrodynamic simulations of heavy-ion collisions," *arXiv preprint arXiv:2302.14184*.

[107] Ma, P., Mondal, A., Konomi, B. A., Hobbs, J., Song, J. J., and Kang, E. L. (2022), "Computer model emulation with high-dimensional functional output in large-scale observing system uncertainty experiments," *Technometrics*, 64, 65–79.

[108] Macal, C. and North, M. (2010), "Tutorial on agent-based modelling and simulation," *Journal of Simulation*, 4, 151–162.

[109] Mak, S. and Joseph, V. R. (2018), "Support points," *The Annals of Statistics*, 46, 2562–2592.

[110] Marcy, P. W. and Storlie, C. B. (2020), "Bayesian calibration of computer models with informative failures," *arXiv preprint arXiv:2006.07546*.

[111] Marque-Pucheu, S., Perrin, G., and Garnier, J. (2020), "An efficient dimension reduction for the Gaussian process emulation of two nested codes with functional outputs," *Computational Statistics*, 35, 1059–1099.

[112] McCullagh, P. and Nelder, J. (2014), *Generalized Linear Models*, Springer.

[113] McDonnell, J., Schunck, N., Higdon, D., Sarich, J., Wild, S., and Nazarewicz, W. (2015), "Uncertainty quantification for nuclear density functional theory and information content of new measurements," *Physical Review Letters*, 114, 122501.

[114] McKay, M., Beckman, R., and Conover, W. (1979), "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, 239–245.

[115] McNelis, M., Bazow, D., and Heinz, U. (2021), "Anisotropic fluid dynamical simulations of heavy-ion collisions," *Computer Physics Communications*, 267, 108077.

[116] Morris, M. D. and Mitchell, T. J. (1995), "Exploratory designs for computational experiments," *Journal of Statistical Planning and Inference*, 43, 381–402.

[117] Moss, H. B., Ober, S. W., and Picheny, V. (2023), "Inducing point allocation for sparse Gaussian processes in high-throughput Bayesian optimisation," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 5213–5230.

[118] Murphy, S. A. and Van der Vaart, A. W. (2000), "On profile likelihood," *Journal of the American Statistical Association*, 95, 449–465.

[119] Muzellec, B., Josse, J., Boyer, C., and Cuturi, M. (2020), "Missing data imputation using optimal transport," in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, vol. 119, pp. 7130–7140.

[120] Nocedal, J. and Wright, S. J. (2006), *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer Nature, 2nd ed.

[121] Ober, S. W., Burt, D. R., Artemev, A., and van der Wilk, M. (2022), "Recommendations for baselines and benchmarking approximate Gaussian processes," In NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems.

[122] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019), "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, 32.

[123] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011), "scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, 12, 2825–2830.

[124] Peng, C.-Y. and Wu, C. J. (2014), "On the choice of nugget in kriging modeling for deterministic computer experiments," *Journal of Computational and Graphical Statistics*, 23, 151–168.

[125] Phillips, D. R., Furnstahl, R. J., Heinz, U., Maiti, T., Nazarewicz, W., Nunes, F. M., Plumlee, M., Pratola, M. T., Pratt, S., Viens, F. G., and Wild, S. M. (2021), "Get on the BAND Wagon: a Bayesian framework for quantifying model uncertainties in nuclear dynamics," *Journal of Physics G: Nuclear and Particle Physics*, 48, 072001.

[126] Plumlee, M. (2017), "Bayesian calibration of inexact computer models," *Journal of the American Statistical Association*, 112, 1274–1285.

[127] Plumlee, M., Asher, T. G., Chang, W., and Bilskie, M. V. (2021), "High-fidelity hurricane surge forecasting using emulation and sequential experiments," *The Annals of Applied Statistics*, 15, 460–480.

[128] Plumlee, M., Sürer, O., Wild, S. M., and Chan, M. Y.-H. (2023), "surmise 0.2.1 Users Manual," Tech. Rep. Version 0.2.1, NAISE.

[129] Quinonero-Candela, J. and Rasmussen, C. E. (2005), "Analysis of some methods for reduced rank Gaussian process regression," in *Switching and Learning in Feedback Systems*, Springer, pp. 98–127.

[130] Rasmussen, C. E. and Williams, C. K. I. (2005), *Gaussian Processes for Machine Learning*, The MIT Press.

[131] Reinhard, P.-G. and Nazarewicz, W. (2017), "Toward a global description of nuclear charge radii: Exploring the Fayans energy density functional," *Physical Review C*, 95, 064328.

[132] Roberts, G. O. and Stramer, O. (2002), "Langevin diffusions and Metropolis-Hastings algorithms," *Methodology and Computing in Applied Probability*, 4, 337–357.

[133] Rougier, J. (2008), "Efficient emulators for multivariate deterministic functions," *Journal of Computational and Graphical Statistics*, 17, 827–843.

[134] Roweis, S. (1997), "EM algorithms for PCA and SPCA," in *Proceedings of the 10th International Conference on Neural Information Processing Systems*, pp. 626—632.

[135] Sacks, J., Schiller, S. B., and Welch, W. J. (1989), "Designs for computer experiments," *Technometrics*, 31, 41–47.

[136] Salter, J. M., Williamson, D. B., Scinocca, J., and Kharin, V. (2019), "Uncertainty quantification for computer models with spatial output using calibration-optimal bases," *Journal of the American Statistical Association*, 114, 1800–1814.

[137] Santner, T. J., Williams, B. J., and Notz, W. I. (2018), *The Design and Analysis of Computer Experiments*, New York: Springer New York.

[138] Sauer, A., Gramacy, R. B., and Higdon, D. (2023), "Active learning for deep Gaussian process surrogates," *Technometrics*, 65, 4–18.

[139] Segal, M. and Xiao, Y. (2011), "Multivariate random forests," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1, 80–87.

[140] Shi, H.-J. M. (2020), "PyTorch-LBFGS," `https://github.com/hjmshi/PyTorch-LBFGS`.

[141] Silva, P. C., Batista, P. V., Lima, H. S., Alves, M. A., Guimarães, F. G., and Silva, R. C. (2020), "COVID-ABS: An agent-based model of COVID-19 epidemic to simulate health and economic effects of social distancing interventions," *Chaos, Solitons & Fractals*, 139, 110088.

[142] Snelson, E. and Ghahramani, Z. (2005), "Sparse Gaussian processes using pseudo-inputs," *Advances in Neural Information Processing Systems*, 18.

[143] Sung, C.-L., Hung, Y., Rittase, W., Zhu, C., and Wu, C. (2020), "Calibration for computer experiments with binary responses and application to cell adhesion study," *Journal of the American Statistical Association*, 115, 1664–1674.

[144] Sürer, Ö., Nunes, F. M., Plumlee, M., and Wild, S. M. (2022), "Uncertainty quantification in breakup reactions," *Physical Review C*, 106, 024607.

[145] Sürer, Ö. and Plumlee, M. (2021), "Calibration using emulation of filtered simulation results," in *2021 Winter Simulation Conference (WSC)*, IEEE, pp. 1–12.

[146] Sürer, Ö., Plumlee, M., and Wild, S. M. (2023), "Sequential Bayesian experimental design for calibration of expensive simulation models," *Technometrics*, to appear.

[147] Surjanovic, S. and Bingham, D. (Retrieved August 10, 2023), "Virtual Library of Simulation Experiments: Test Functions and Datasets," From `http://www.sfu.ca/~ssurjano`.

[148] Teh, Y. W., Seeger, M., and Jordan, M. I. (2005), "Semiparametric latent factor models," in *International Workshop on Artificial Intelligence and Statistics*, PMLR, pp. 333–340.

[149] The Python Software Foundation (2023), "PyPI - The Python Package Index," `https://pypi.org`, Accessed: November 10, 2023.

[150] Tipping, M. E. (2001), "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, 1, 211–244.

[151] Tipping, M. E. and Bishop, C. M. (1999), "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*,

61, 611–622.

[152] Titsias, M. (2009), "Variational learning of inducing variables in sparse Gaussian processes," in *Artificial Intelligence and Statistics*, PMLR, pp. 567–574.

[153] Tuo, R. and Wu, C. F. J. (2015), "Efficient calibration for imperfect computer models," *The Annals of Statistics*, 43, 2331–2352.

[154] — (2016), "A theoretical framework for calibration in computer models: Parametrization, estimation and convergence properties," *SIAM/ASA Journal on Uncertainty Quantification*, 4, 767–795.

[155] Uhrenholt, A. K., Charvet, V., and Jensen, B. S. (2021), "Probabilistic selection of inducing points in sparse Gaussian processes," in *Uncertainty in Artificial Intelligence*, PMLR, pp. 1035–1044.

[156] Vecchia, A. V. (1988), "Estimation and model identification for continuous spatial processes," *Journal of the Royal Statistical Society: Series B (Methodological)*, 50, 297–312.

[157] Venkatramanan, S., Sadilek, A., Fadikar, A., Barrett, C. L., Biggerstaff, M., Chen, J., Dotiwalla, X., Eastham, P., Gipson, B., Higdon, D., Kucuktunc, O., Lieber, A., Lewis, B. L., Reynolds, Z., Vullikanti, A. K., Wang, L., and Marathe, M. (2021), "Forecasting influenza activity using machine-learned mobility map," *Nature Communications*, 12.

[158] Ver Hoef, J. M. and Barry, R. P. (1998), "Constructing and fitting models for cokriging and multivariable spatial prediction," *Journal of Statistical Planning and Inference*, 69, 275–294.

[159] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020), "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, 17, 261–272.

[160] Wasserman, L. (2004), *All of Statistics: A Concise Course in Statistical Inference*, vol. 26, Springer.

[161] Williams, B., Higdon, D., Gattiker, J., Moore, L., McKay, M., Keller-McNulty, S., et al. (2006), "Combining experimental data and computer simulations, with an application to flyer plate experiments," *Bayesian Analysis*, 1, 765–792.

[162] Williams, C. and Seeger, M. (2000), "Using the Nyström method to speed up kernel machines," *Advances in Neural Information Processing Systems*, 13.

[163] Williams, C. K. and Barber, D. (1998), "Bayesian classification with Gaussian processes," *IEEE Transactions on pattern analysis and machine intelligence*, 20, 1342–1351.

[164] Yu, Y. and Bulgac, A. (2003), "Energy density functional approach to superfluid nuclei," *Physical Review Letters*, 90, 222501.

[165] Zhang, H. (1998), "Classification trees for multiple binary responses," *Journal of the American Statistical Association*, 93, 180–193.

[166] Zhang, H. and Vorobeychik, Y. (2019), "Empirically grounded agent-based models of innovation diffusion: A critical review," *Artificial Intelligence Review*, 52, 707–741.

[167] Zhang, W., Valencia, A., and Chang, N.-B. (2021), "Synergistic integration between machine learning and agent-based modeling: A multidisciplinary review," *IEEE Transactions on Neural Networks and Learning Systems*.

APPENDIX A

Proofs and computational details for PCGPwM

A.1. Proof of Theorem 2.1

PROOF. First, we note that $\tilde{\boldsymbol{G}}_{i,\cdot} = \boldsymbol{G}_{i,\cdot}$ and $u_{ik} = 0$ for all k because

$$\tilde{\boldsymbol{G}}_{i,\cdot} = \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{B}^{\mathsf{T}}_{\mathcal{J}(i),\cdot} \boldsymbol{B}^{-1}_{\mathcal{J}(i),\mathcal{J}(i)} \boldsymbol{F}^{\mathsf{T}}_{i,\mathcal{J}(i)} = \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{B} \boldsymbol{B}^{-1} \boldsymbol{F}^{\mathsf{T}}_{i,\cdot} = \boldsymbol{G}_{i,\cdot},$$

$$u_{ik} = \boldsymbol{\Phi}^{\mathsf{T}}_{\cdot,k} \left(\boldsymbol{B} - \boldsymbol{B} \boldsymbol{B}^{-1} \boldsymbol{B}^{\mathsf{T}} \right) \boldsymbol{\Phi}_{\cdot,k} = 0, \quad \text{for all } k.$$

Then we observe that $\boldsymbol{r}_k(\boldsymbol{\theta}_i)$ is the i row of $\boldsymbol{R}_k$, and thus $\boldsymbol{r}^{\mathsf{T}}_k(\boldsymbol{\theta}_i) \boldsymbol{R}^{-1}_k = \boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is the ith vector of the identity matrix. We conclude that

$$\tilde{\mu}_k(\boldsymbol{\theta}_i) = \boldsymbol{e}^{\mathsf{T}}_i \tilde{\boldsymbol{G}}_{\cdot,k} = \tilde{\boldsymbol{G}}_{ik} = \boldsymbol{G}_{ik} = \boldsymbol{f}(\boldsymbol{\theta}_i)^{\mathsf{T}} \boldsymbol{\Phi}_{\cdot,k},$$

$$\tilde{\sigma}^2_k(\boldsymbol{\theta}_i) = \lambda_k \left(\rho_k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i) - \boldsymbol{e}^{\mathsf{T}}_i \boldsymbol{r}_k(\boldsymbol{\theta}_i) \right) = \lambda_k \left(\rho_k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i) - \rho_k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i) \right) = 0.$$

□

A.2. Boundedness of w_{ik}

LEMMA A.1. *For any $\mathcal{J}(i) \subseteq \{1, \ldots, m\}$, w_{ik} given in expression* (2.7) *is bounded between 0 and* 1.

PROOF. For this proof we will shorten $\mathcal{J}(i)$ to $\mathcal{J}$ and drop the subscript i from w_{ik}. We have that$\lambda_k w_k = \boldsymbol{\Phi}^{\mathsf{T}}_k \left(\boldsymbol{B} - \boldsymbol{B}^{\mathsf{T}}_{\mathcal{J},\cdot} \boldsymbol{B}^{-1}_{\mathcal{J}\mathcal{J}} \boldsymbol{B}_{\mathcal{J},\cdot} \right) \boldsymbol{\Phi}_k$.

Without loss of generality, say that $\mathcal{I} = \{1, \ldots, t\}$ and $\mathcal{J} = \{t+1, \ldots, m\}$. The matrix $\boldsymbol{B}$ can then be written as

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}_{\mathcal{I}\mathcal{I}} & \boldsymbol{B}_{\mathcal{I}\mathcal{J}} \\ \boldsymbol{B}_{\mathcal{J}\mathcal{I}} & \boldsymbol{B}_{\mathcal{J}\mathcal{J}} \end{pmatrix}.$$

Letting $\boldsymbol{v} = \boldsymbol{\Phi}_{\mathcal{I}k}$ and $\boldsymbol{u} = -\boldsymbol{B}_{\mathcal{J}\mathcal{J}}^{-1}\boldsymbol{B}_{\mathcal{J}\mathcal{I}}\boldsymbol{v}$, we have $\boldsymbol{v}^\mathsf{T}\boldsymbol{B}_{\mathcal{I}\mathcal{J}}\boldsymbol{B}_{\mathcal{J}\mathcal{J}}^{-1}\boldsymbol{B}_{\mathcal{J}\mathcal{I}}\boldsymbol{v} = -\boldsymbol{u}^\mathsf{T}\boldsymbol{B}_{\mathcal{J}\mathcal{J}}\boldsymbol{u} - 2\boldsymbol{u}^\mathsf{T}\boldsymbol{B}_{\mathcal{J}\mathcal{I}}\boldsymbol{v}$, and conclude that

$$\begin{aligned} \boldsymbol{v}^\mathsf{T}\left(\boldsymbol{B}_{\mathcal{I}\mathcal{I}} - \boldsymbol{B}_{\mathcal{I}\mathcal{J}}\boldsymbol{B}_{\mathcal{J}\mathcal{J}}^{-1}\boldsymbol{B}_{\mathcal{I}\mathcal{J}}^\mathsf{T}\right)\boldsymbol{v} &= \boldsymbol{v}^\mathsf{T}\boldsymbol{B}_{\mathcal{I}\mathcal{I}}\boldsymbol{v} + \boldsymbol{u}^\mathsf{T}\boldsymbol{B}_{\mathcal{J}\mathcal{J}}\boldsymbol{u} + 2\boldsymbol{u}^\mathsf{T}\boldsymbol{B}_{\mathcal{J}\mathcal{I}}\boldsymbol{v} \\ &= \begin{pmatrix} \boldsymbol{v} & \boldsymbol{u} \end{pmatrix} \begin{pmatrix} \boldsymbol{B}_{\mathcal{I}\mathcal{I}} & \boldsymbol{B}_{\mathcal{I}\mathcal{J}} \\ \boldsymbol{B}_{\mathcal{J}\mathcal{I}} & \boldsymbol{B}_{\mathcal{J}\mathcal{J}} \end{pmatrix} \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{u} \end{pmatrix}. \end{aligned} \tag{A.1}$$

Note that $\boldsymbol{B} = \boldsymbol{\Phi}(\boldsymbol{\Lambda} - \varepsilon\boldsymbol{I})\boldsymbol{\Phi}^\mathsf{T} + \varepsilon\boldsymbol{I}$. By construction, since each diagonal element of $\boldsymbol{\Lambda}$ is larger than ε, $\boldsymbol{B}$ is the sum of two positive-definite matrices and is thus positive definite. Since the right-hand side in (A.1) is larger than zero, we have that $w_k \geq 0$.

Now, since $\boldsymbol{B}_{\mathcal{J}\mathcal{J}}$ is positive definite, $\boldsymbol{B}_{\mathcal{J}\mathcal{J}}^{-1}$ is positive definite and thus

$$(\boldsymbol{B}_{\mathcal{J},\cdot}\boldsymbol{\Phi}_k)^\mathsf{T}\boldsymbol{B}_{\mathcal{J}\mathcal{J}}^{-1}\left(\boldsymbol{B}_{\mathcal{J},\cdot}\boldsymbol{\Phi}_k\right) \geq \boldsymbol{0}.$$

We conclude that $\boldsymbol{\Phi}_k^\mathsf{T}\left(\boldsymbol{B} - \boldsymbol{B}_{\mathcal{J},\cdot}^\mathsf{T}\boldsymbol{B}_{\mathcal{J}\mathcal{J}}^{-1}\boldsymbol{B}_{\mathcal{J},\cdot}\right)\boldsymbol{\Phi}_k \leq \boldsymbol{\Phi}_k^\mathsf{T}\boldsymbol{B}\boldsymbol{\Phi}_k$. Finally, we note that by the orthogonality of $\boldsymbol{\Phi}$ and letting $\boldsymbol{e}_k$ be the kth vector of the identity matrix, we have that

$$\boldsymbol{\Phi}_k^\mathsf{T}\boldsymbol{B}\boldsymbol{\Phi}_k = \boldsymbol{\Phi}_k^\mathsf{T}(\boldsymbol{\Phi}(\boldsymbol{\Lambda} - \varepsilon\boldsymbol{I})\boldsymbol{\Phi}^\mathsf{T} + \varepsilon\boldsymbol{I})\boldsymbol{\Phi}_k = \boldsymbol{e}_k^\mathsf{T}(\boldsymbol{\Lambda} - \varepsilon\boldsymbol{I})\boldsymbol{e}_k + \varepsilon = \lambda_k,$$

and thus $w_k \leq 1$. □

A.3. Inverse of matrices with an increasing diagonal

LEMMA A.2. *If the norm of an n-by-n matrix* $\boldsymbol{A}$ *is bounded (i.e.,* $\|\boldsymbol{A}\| < L$*), then* $\lim_{\eta\to\infty}(\boldsymbol{A}+\eta\boldsymbol{I})^{-1} = \boldsymbol{0}$.

PROOF. For $\eta \geq L, \|\eta^{-1}\boldsymbol{A}\| < 1$. Consequently, for the partial sum $\boldsymbol{T}_r = \boldsymbol{I} + \sum_{j=1}^{r}(-1)^j\eta^{-j}\boldsymbol{A}^j$, we have that $\lim_{r\to\infty}\boldsymbol{T}_r = (\boldsymbol{I}+\eta^{-1}\boldsymbol{A})^{-1}$ [53, p.135]. Consider the norm of the partial sum,

$$\begin{aligned}\|\boldsymbol{T}_r\| = \left\|\boldsymbol{I} + \sum_{j=1}^{r}(-1)^j\eta^{-j}\boldsymbol{A}^j\right\| &\leq \|\boldsymbol{I}\| + \sum_{j=1}^{r}\eta^{-j}\|\boldsymbol{A}\|^j \\ &\leq n - 1 + \frac{1-\eta^{-(r+1)}}{1-\eta^{-1}} \to n - 1 + \frac{1}{1-\eta^{-1}},\end{aligned}$$

as $r \to \infty$. The sequence of matrices $\{\boldsymbol{T}_r\}_{r=1}^{\infty}$ uniformly converges over the entries, therefore,

$$\lim_{\eta\to\infty}(\boldsymbol{A}+\eta\boldsymbol{I})^{-1} = \lim_{\eta\to\infty}\lim_{r\to\infty}\eta^{-1}\boldsymbol{T}_r = \lim_{r\to\infty}\lim_{\eta\to\infty}\eta^{-1}\left(\boldsymbol{I} + \sum_{j=1}^{r}(-1)^j\eta^{-j}\boldsymbol{A}^j\right) = \boldsymbol{0}.$$

□

A.4. Proof of Theorem 2.2

PROOF. From Lemma A.1 in Supplementary Material A.2, without loss of generality, partition $\{1,\ldots,n\}$ into $\mathcal{Q} = \{1,\ldots,t\}$ and $\mathcal{S} = \{t+1,\ldots,n\}$, where $w_{ik} < 1$ for $i \in \mathcal{Q}$ and $w_{ik} = 1$ for $i \in \mathcal{S}$. Let $\boldsymbol{Q}_k$ be an n-by-n diagonal matrix with diagonal entries $\left(\frac{w_{1k}}{(1-w_{1k})^\alpha},\ldots,\frac{w_{tk}}{(1-w_{tk})^\alpha},0,\ldots,0\right)$. Let $\boldsymbol{S}$ be $\begin{bmatrix}0_{t\times(n-t)}\\ \boldsymbol{I}_{n-t}\end{bmatrix}$. Then, once η is big enough such that for all sets $J \subseteq \{1,\ldots,m\}$ with at least one entry, $\frac{w_{ik}}{(1-w_{ik})^\alpha} < \eta$, the adjusted

covariance matrix can be rewritten as

$$\boldsymbol{R}_k + \beta_k \text{diag}(\boldsymbol{v}_k) = \boldsymbol{R}_k + \beta_k(\boldsymbol{Q}_k + \eta \boldsymbol{S}\boldsymbol{S}^\mathsf{T}).$$

Without loss of generality, let $\beta_k = 1$. We drop the subscript k for the remainder of the proof for brevity. Partition the covariance matrix as

$$\boldsymbol{R} + \boldsymbol{Q} + \eta \boldsymbol{S}\boldsymbol{S}^\mathsf{T} = \begin{bmatrix} (\boldsymbol{R} + \boldsymbol{Q})_{\mathcal{Q}\mathcal{Q}} & \boldsymbol{R}_{\mathcal{Q}\mathcal{S}} \\ \boldsymbol{R}_{\mathcal{S}\mathcal{Q}} & \boldsymbol{R}_{\mathcal{S}\mathcal{S}} + \eta \boldsymbol{I} \end{bmatrix}.$$

Recall that for an invertible matrix $\boldsymbol{M} = \begin{bmatrix} \boldsymbol{W} & \boldsymbol{X} \\ \boldsymbol{Y} & \boldsymbol{Z} \end{bmatrix}$, the inverse [53, p.95] is

$$\boldsymbol{M}^{-1} = \begin{bmatrix} \boldsymbol{W} & \boldsymbol{X} \\ \boldsymbol{Y} & \boldsymbol{Z} \end{bmatrix}^{-1} = \begin{bmatrix} (\boldsymbol{W} - \boldsymbol{X}\boldsymbol{Z}^{-1}\boldsymbol{Y})^{-1} & -\boldsymbol{W}^{-1}\boldsymbol{X}(\boldsymbol{Z} - \boldsymbol{Y}\boldsymbol{W}^{-1}\boldsymbol{X})^{-1} \\ -\boldsymbol{Z}^{-1}\boldsymbol{Y}(\boldsymbol{W} - \boldsymbol{X}\boldsymbol{Z}^{-1}\boldsymbol{Y})^{-1} & (\boldsymbol{Z} - \boldsymbol{Y}\boldsymbol{W}^{-1}\boldsymbol{X})^{-1} \end{bmatrix}.$$

The norm $\|\boldsymbol{R}_{\mathcal{S}\mathcal{S}}\|$ is bounded by the largest singular value of $\boldsymbol{R}_{\mathcal{S}\mathcal{S}}$. Using Lemma A.2 in Supplementary Material A.3,

$$\lim_{\eta \to \infty} (\boldsymbol{R}_{\mathcal{S}\mathcal{S}} + \eta \boldsymbol{I})^{-1} = \boldsymbol{0},$$

$$\lim_{\eta \to \infty} \left(\boldsymbol{R}_{\mathcal{S}\mathcal{S}} + \eta \boldsymbol{I} - \boldsymbol{R}_{\mathcal{S}\mathcal{Q}}(\boldsymbol{R} + \boldsymbol{Q})^{-1}_{\mathcal{Q}\mathcal{Q}} \boldsymbol{R}_{\mathcal{Q}\mathcal{S}}\right)^{-1} = \boldsymbol{0}.$$

By the Woodbury identity [53, p.221],

$$\begin{aligned}
&\left((\boldsymbol{R}+\boldsymbol{Q})_{\mathcal{Q}\mathcal{Q}} - \boldsymbol{R}_{\mathcal{Q}\mathcal{S}}(\boldsymbol{R}_{\mathcal{S}\mathcal{S}}+\eta\boldsymbol{I})^{-1}\boldsymbol{R}_{\mathcal{S}\mathcal{Q}}\right)^{-1} \\
&= (\boldsymbol{R}+\boldsymbol{Q})^{-1}_{\mathcal{Q}\mathcal{Q}} + (\boldsymbol{R}+\boldsymbol{Q})^{-1}_{\mathcal{Q}\mathcal{Q}}\boldsymbol{R}_{\mathcal{Q}\mathcal{S}}\left(\boldsymbol{R}_{\mathcal{S}\mathcal{S}}+\eta\boldsymbol{I}-\boldsymbol{R}_{\mathcal{S}\mathcal{Q}}(\boldsymbol{R}+\boldsymbol{Q})^{-1}_{\mathcal{Q}\mathcal{Q}}\boldsymbol{R}_{\mathcal{Q}\mathcal{S}}\right)^{-1}\boldsymbol{R}_{\mathcal{S}\mathcal{Q}}(\boldsymbol{R}+\boldsymbol{Q})^{-1}_{\mathcal{Q}\mathcal{Q}} \\
&\to (\boldsymbol{R}+\boldsymbol{Q})^{-1}_{\mathcal{Q}\mathcal{Q}}.
\end{aligned}$$

To summarize, as $\eta \to \infty$,

$$\left(\boldsymbol{R}+\boldsymbol{Q}+\eta\boldsymbol{S}\boldsymbol{S}^{\mathsf{T}}\right)^{-1} \to \begin{bmatrix} (\boldsymbol{R}+\boldsymbol{Q})^{-1}_{\mathcal{Q}\mathcal{Q}} & \boldsymbol{0}_{t\times(n-t)} \\ \boldsymbol{0}_{(n-t)\times t} & \boldsymbol{0}_{(n-t)\times(n-t)} \end{bmatrix}.$$

Therefore, for any length-n vector $\boldsymbol{a}$,

$$\left(\left(\boldsymbol{R}+\boldsymbol{Q}+\eta\boldsymbol{S}\boldsymbol{S}^{\mathsf{T}}\right)^{-1}\boldsymbol{a}\right)_{\mathcal{Q}} \to (\boldsymbol{R}+\boldsymbol{Q})^{-1}_{\mathcal{Q}\mathcal{Q}}\,\boldsymbol{a}_{\mathcal{Q}},$$

$$\left(\left(\boldsymbol{R}+\boldsymbol{Q}+\eta\boldsymbol{S}\boldsymbol{S}^{\mathsf{T}}\right)^{-1}\boldsymbol{a}\right)_{\mathcal{S}} \to \boldsymbol{0}.$$

Pulling these results through our equations for predictions (i.e., (2.9) and (2.10)), we conclude that the limits of $\tilde{\mu}_k(\boldsymbol{\theta})$ and $\tilde{\sigma}^2_k(\boldsymbol{\theta})$ will not depend on any row i with $w_{ik} = 1$, and thus the result holds. □

A.5. Estimating principal component matrix $\boldsymbol{\Phi}$

The principal component matrix $\boldsymbol{\Phi}$ is estimated with missing values in the observed simulation output $\boldsymbol{F}$. For a certain row i, the index set $\mathcal{J}(i)$ contains the indices where data are not missing. Denote the missing index set as $\sim\mathcal{J}(i) = \{1,\ldots,m\} \setminus \mathcal{J}(i)$. The principal component matrix $\boldsymbol{\Phi}$ is approximated via Algorithm 3 which mirrors the typical

EM structure. The simulation output $\boldsymbol{F}$ is assumed to be scaled to zero mean and unit variance in its columns. The E step is standard, employing conditional equations regarding multivariate normal distribution. However, we note that the M step does not result in the exact optimizer since $\epsilon_{\mathrm{M}} > 0$. In our implementation, $\epsilon_{\mathrm{M}} = 10^{-5}$ provides reasonable performance and ensures the procedure is numerically stable.

Algorithm 3: EM algorithm for estimating $\boldsymbol{\Phi}$

```
   /* Initialization with column means of non-missing values */
 1 for j ∈ {1,...,m} do
 2 |  Let I(j) = {i : j ∈ J(i)}.
 3 |  forall i ∉ I(j) do  f̃_j(θ_i) = (1/|I(j)|) Σ_{i∈I(j)} f_j(θ_i).   /* missing */
 4 |  forall i ∈ I(j) do  f̃_j(θ_i) = f_j(θ_i).                  /* non-missing */
 5 end
   /* EM algorithm */
 6 while convergence criterion not met do
 7 |  Obtain Φ, Λ via SVD of F̃ = (f̃(θ_1)^T,...,f̃(θ_n)^T)^T.   /* M step */
 8 |  for i ∈ {1,...,n} do                                      /* E step */
 9 |  |  f̃_{~J(i)}(θ_i) =
   |  |    Φ_{~J(i),·}(Λ − εI)Φ^T_{J(i),·} ( Φ_{J(i),·}(Λ − εI)Φ^T_{J(i),·} + ε_M I )^{-1} f_{J(i)}(θ_i).
10 |  end
11 end
```

A.6. Computing Inverse submatrix $\boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)}$

The main inferences introduced in (2.6) and (2.7) depend on inverting the matrix $\boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)}$. Recall the definition of the covariance matrix

$$\boldsymbol{B} = \boldsymbol{\Phi}\,\mathrm{diag}\left(\lambda_1 - \varepsilon, \ldots, \lambda_\kappa - \varepsilon\right)\,\boldsymbol{\Phi}^{\mathsf{T}} + \varepsilon \boldsymbol{I}.$$

Let $\boldsymbol{\Lambda}_\varepsilon = \operatorname{diag}(\lambda_1 - \varepsilon, \ldots, \lambda_\kappa - \varepsilon)$. Now consider the inverse of $\boldsymbol{B}$ by using the Woodbury matrix identity,

$$\begin{aligned}
\boldsymbol{B}^{-1} &= \left(\varepsilon \boldsymbol{I} + \boldsymbol{\Phi}\boldsymbol{\Lambda}_\varepsilon \boldsymbol{\Phi}^\mathsf{T}\right)^{-1} \\
&= \varepsilon^{-1}\boldsymbol{I} - \varepsilon^{-2}\boldsymbol{\Phi}\left(\boldsymbol{\Lambda}_\varepsilon^{-1} + \varepsilon^{-1}\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^\mathsf{T} \\
&= \varepsilon^{-1}\boldsymbol{I} - \varepsilon^{-1}\boldsymbol{\Phi}\operatorname{diag}\left(\frac{\lambda_1 - \varepsilon}{\lambda_1}, \ldots, \frac{\lambda_\kappa - \varepsilon}{\lambda_\kappa}\right)\boldsymbol{\Phi}^\mathsf{T} \\
&= \varepsilon^{-1}\left(\boldsymbol{I} - \boldsymbol{\Phi}\operatorname{diag}\left(\frac{\lambda_1 - \varepsilon}{\lambda_1}, \ldots, \frac{\lambda_\kappa - \varepsilon}{\lambda_\kappa}\right)\boldsymbol{\Phi}^\mathsf{T}\right).
\end{aligned}$$

The expression above is simple to compute and requires no matrix inversion. But we are concerned about the inverse of the submatrix $\boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)}$, where

$$\boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)} = \varepsilon\boldsymbol{I} + \boldsymbol{\Phi}_{\mathcal{J}(i),\cdot}\,\boldsymbol{\Lambda}_\varepsilon\,\boldsymbol{\Phi}_{\mathcal{J}(i),\cdot}^\mathsf{T}.$$

Also by Woodbury identity, we then have

$$\begin{aligned}
\boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)}^{-1} &= \left(\varepsilon\boldsymbol{I} + \boldsymbol{\Phi}_{\mathcal{J}(i),\cdot}\,\boldsymbol{\Lambda}_\varepsilon\,\boldsymbol{\Phi}_{\mathcal{J}(i),\cdot}^\mathsf{T}\right)^{-1} \\
&= \varepsilon^{-1}\boldsymbol{I} - \varepsilon^{-2}\boldsymbol{\Phi}\left(\boldsymbol{\Lambda}_\varepsilon^{-1} + \varepsilon^{-1}\boldsymbol{\Phi}_{\mathcal{J}(i),\cdot}^\mathsf{T}\,\boldsymbol{\Phi}_{\mathcal{J}(i),\cdot}\right)^{-1}\boldsymbol{\Phi}^\mathsf{T}.
\end{aligned}$$

The resulting expression cannot be further simplified since $\boldsymbol{\Phi}_{\mathcal{J}(i),\cdot}^\mathsf{T}\boldsymbol{\Phi}_{\mathcal{J}(i),\cdot}$ no longer equals the identity matrix or some simple known form. However, the algebraic manipulation may still be beneficial when we consider only the inverse of the inner κ-by-κ matrix, as opposed to the generally larger matrix $\boldsymbol{B}_{\mathcal{J}(i),\mathcal{J}(i)}$.

A.7. Investigation on hyperparameter values for missingness

We construct surrogates for the four test functions with MCAR responses at 5%. The details of the test functions are given in Supplementary Material A.8. The root mean squared error (RMSE) between the function values and the surrogate's predicted values was evaluated with a set of holdout simulation runs. Figure A.1 shows the RMSE for selected α values with $\beta_k = 1$ or β_k optimized. The RMSE generally increases as α increases. When β_k's are fixed to be 1, the error increases for larger ranges of α, specifically for the wingweight function, the error continuously increases for $\alpha > 0$. The benefit in including β_k's is shown in the generally lower RMSE achieved. We suggest to include β_k's in the hyperparameter estimation in constructing the surrogate. When β_k's are optimized, smaller α values are preferred.

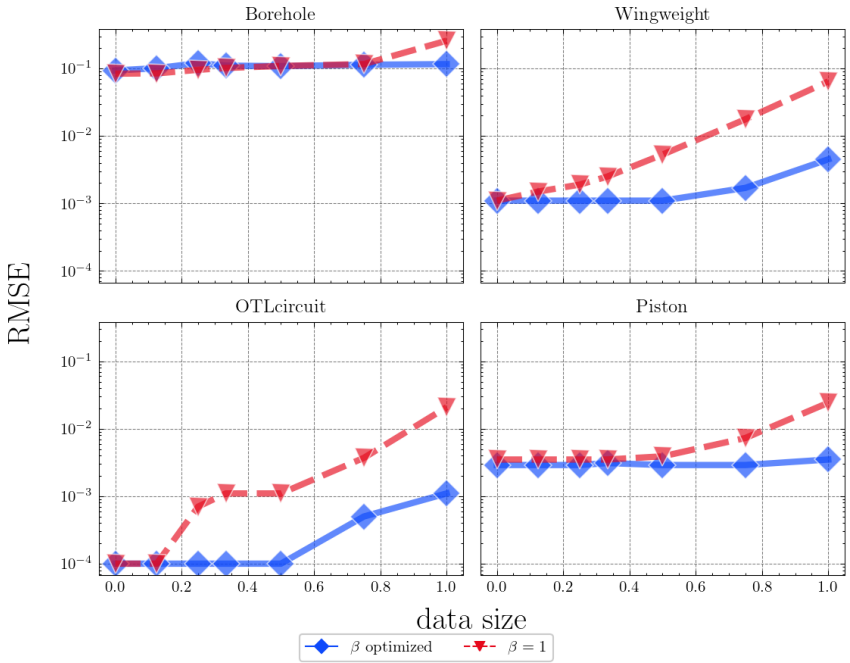


Figure A.1. RMSE with $\alpha \in [0, 1]$ and β_k's either optimized or $= 1$.

A.8. Test functions

This section provides details of the four test functions used in evaluating the performance of the algorithm. The subscripts in the expressions in this section refer to the elements in each of the dimension of vector variables $\boldsymbol{\theta}$ and $\boldsymbol{x}$, whereas in all other sections, the subscripted $\boldsymbol{\theta}_i$ and $\boldsymbol{x}_j$ refer to the ith parameter and the jth locations, respectively. The reference parameter $\boldsymbol{\theta}^\star$ refers to the center of the parameter space, for example $(0.5, 0.5, 0.5)^\mathsf{T}$ in a 3D unit hypercube.

For all functions, under MCAR, missingness is assigned with equal probability everywhere with the desired percentage. Under MAR, missingness is randomly assigned to a subset of locations, according to the missing-at-random scheme reported in Muzellec et al. [119]. For MNAR, the missingness mechanism is included in each function below.

A.9. Modified Borehole function

This test function is developed for the purpose of calibration and is modified from the Borehole function [137]. The function is defined as

$$f(\boldsymbol{\theta}, \boldsymbol{x}) = \frac{2\pi(\theta_1 - x_1)}{2(\theta_2/x_2^2) + \theta_3} \exp\left(\theta_4 x_2\right), \tag{A.2}$$

where the ranges of the variables are included in Table A.1. The first four variables are to be tuned as the parameter, and the last two variables are fixed as locations.

$\boldsymbol{\theta}$	θ_1	[990, 1110]
	θ_2	[0.074, 1.12]
	θ_3	[0.05, 0.5]
	θ_4	[-0.5, 0.5]
$\boldsymbol{x}$	x_1	[700, 820]
	x_2	[0.05, 0.5]

Table A.1. Variable ranges for borehole function in (A.2).

Furthermore, a missingness of MNAR is introduced into the model evaluations, in the following manner:

$$f(\boldsymbol{\theta}, \boldsymbol{x}) = \begin{cases} f(\boldsymbol{\theta}, \boldsymbol{x}), & \text{if } f(\boldsymbol{\theta}, \boldsymbol{x}) \leq c_1 f(\boldsymbol{\theta}^\star, \boldsymbol{x}) \\ \text{NA}, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{\theta}^\star$ is the center of the standardized parameter space $U[0,1]^4$. The constant $c_1 > 0$ is adjusted to achieve a desired probability of missingness.

A.10. Piston function

The piston function [15] is given as

$$f(\boldsymbol{\theta}, \boldsymbol{x}) = 2\pi \sqrt{\frac{M}{k + S^2 \frac{P_0 V_0}{T_0} \frac{T_a}{V^2}}}\,, \tag{A.3}$$

where

$$V = \frac{S}{2k}\left(\sqrt{A^2 + 4k\frac{P_0 V_0}{T_0} T_a} - A\right),$$
$$A = P_0 S + 19.62M - \frac{kV_0}{S},$$

and where the variable partition $(\boldsymbol{\theta}, \boldsymbol{x})$ and their ranges are shown in Table A.2.

	Variable	Range
$\boldsymbol{\theta}$	k	[1000, 5000]
	P_0	[90000, 110000]
	T_a	[290, 296]
$\boldsymbol{x}$	M	[30, 60]
	S	[0.005, 0.02]
	V_0	[0.0002, 0.01]
	T_0	[340, 360]

Table A.2. Variable ranges for the piston function in (A.3).

The function produces a MNAR that follows the "logistic missing not-at-random" scheme used in Muzellec et al. [119], where the probability of an entry missing is according to a logistic model over the columns.

A.11. Wingweight function

The wingweight function [46, ch. 1] is given as

$$f(\boldsymbol{\theta}, \boldsymbol{x}) = 0.036 S_w^{0.758} W_{fw}^{0.0035} \left(\frac{A}{\cos^2(\Lambda)}\right)^{0.6} q^{0.006} \lambda^{0.04} \left(\frac{100 t_c}{\cos(\Lambda)}\right)^{-0.3} (N_z W_{dg})^{0.49} + S_w W_p, \tag{A.4}$$

and the variable partition $(\boldsymbol{\theta}, \boldsymbol{x})$ and the respective ranges are shown in Table A.3.

The function produces a MNAR in the following manner:

$$f(\boldsymbol{\theta}, \boldsymbol{x}) = \begin{cases} f(\boldsymbol{\theta}, \boldsymbol{x}), & \text{if } f(\boldsymbol{\theta}, \boldsymbol{x}) \leq c_2 f(\boldsymbol{\theta}^\star, \boldsymbol{x}) \\ \text{NA}, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{\theta}^\star$ is the center of the standardized parameter space $U[0,1]^4$. The constant $c_2 > 0$ is adjusted to achieve a desired probability of missingness.

	Variable	Range
$\boldsymbol{\theta}$	A	[6, 10]
	Λ	$[-10°, 10°]$
	q	[16, 45]
	λ	[0.5, 1]
$\boldsymbol{x}$	S_w	[150, 200]
	W_{fw}	[220, 300]
	t_c	[0.08, 0.18]
	N_z	[2.5, 6]
	W_{dg}	[1700, 2500]
	W_p	[0.025, 0.08]

Table A.3. Variable ranges for wingweight function in (A.4).

A.12. OTLcircuit function

The OTLcircuit function [15] is given as

$$f(\boldsymbol{\theta}, \boldsymbol{x}) = \frac{(V_{b1} + 0.74)\beta(R_{c2} + 9)}{\beta(R_{c2} + 9) + R_f} + \frac{11.35R_f}{\beta(R_{c2} + 9) + R_f} + \frac{0.74R_f\beta(R_{c2} + 9)}{(\beta(R_{c2} + 9) + R_f)R_{c1}}, \quad \text{(A.5)}$$

where

$$V_{b1} = \frac{12R_{b2}}{R_{b1} + R_{b2}},$$

and the variable partition $(\boldsymbol{\theta}, \boldsymbol{x})$ and the respective ranges are shown in Table A.4.

	Variable	Range
$\boldsymbol{\theta}$	R_f	[0.5, 3]
	β	[50, 300]
$\boldsymbol{x}$	R_{b1}	[50, 150]
	R_{b2}	[25, 70]
	R_{c1}	[1.2, 2.5]
	R_{c2}	[0.25, 1.2]

Table A.4. Variable ranges for the OTLcircuit function in (A.5).

The function produces a MNAR that follows the"logistic missing not-at-random" scheme used in Muzellec et al. [119], where the probability of missing is according to a logistic model over the columns.

A.13. Full results for surrogate comparison experiment

This section presents the full simulation results following the numerical experiments detailed in Section 2.4. Figures in this section are relegated to Appendix B. The surrogate methods compared are the proposed method PCGPwM, two simplistic imputation methods PCGP-kNN and PCGP-BR, EMGP, colGP, and GP-OM. The surrogate methods are tested under nine missingness scenarios: (MCAR, 1%), (MCAR, 5%), (MCAR, 25%), (MNAR, 1%), (MNAR, 5%), (MNAR, 25%), (MAR, 1%), (MAR, 5%), and (MAR, 25%), where results for the scenario (MNAR, 5%) is reported in the main text of the document, in Section 2.4. To summarize the metrics of surrogate quality, the test RMSE is recorded to measure the accuracy of the surrogate; the 90% coverage and the 90% width are used to assess the uncertainty quantification of the surrogate.

A.14. Results under MNAR

The results under MNAR are presented, in the order of 1% and 25%. The results for 5% missingness is presented in the main text.

Figure B.1 shows the test RMSEs for the surrogate methods under 1% missingness. The simplistic imputation methods fail to reduce the errors (i.e., fail to improve surrogate predictions) as N grows for two of the functions (borehole and wingweight). The corresponding missingness mechanism is MNAR, depending on the unobserved function value. The simplistic imputation methods reduce the errors in the other two functions (piston

and OTLcircuit). The RMSEs from PCGPwM and the other methods decrease as N increases in all four functions. The EMGP method consistently results in larger error than PCGPwM, and colGP. The colGP method performs well across functions. However, colGP will not be able to compete in terms of its computation time at higher output dimensions.

Figure B.2 shows the 90% coverage under 1% missingness, and Figure B.3 shows the 90% width. GP-OM results in a high coverage in the borehole function but poor coverages in the other functions. The 90% widths show that GP-OM results in a confidence interval too wide in the borehole and too narrow in the others. The EMGP method achieves the prescribed coverage in the piston and OTLcircuit functions, while undercovering in the borehole function. The coverage from EMGP improves as N grows and achieves the prescribed level for larger Ns. PCGPwM, the simplistic imputation methods, and colGP result in adequate coverages in all functions. However, the simplistic imputation methods maintain wider intervals than PCGPwM in all four cases.

Figure B.4 shows the test RMSEs under 25% MNAR. Similar conclusions to the 1% case can be drawn about the simplistic imputation methods, where they fail to improve in two functions given more data. The other methods exhibit similar behaviors in predictive accuracy.

Figure B.5 shows the coverages under 25% MNAR and Figure B.6 shows the widths of the intervals. We observe that GP-OM and the simplistic imputation methods result in similar behavior as the other cases, meaning that GP-OM results in high coverage in only one function, and the simplistic imputation method achieves high coverage with generally wide intervals. PCGPwM results in adequate coverages with narrower intervals for all functions except the OTLcircuit function for $n = 2500$.

A.15. Results under MCAR

The results under MCAR are presented, in the order of 1%, 5%, and 25%.

Figure B.7 shows the test RMSEs for the surrogate methods under 1% MCAR. The RMSEs decrease as N increases for all methods in all four test functions. Among the principal component-based methods, PCGPwM generally achieves lower errors than the simplistic imputation methods. The colGP method performs well across the functions, and the EMGP method generally results in higher error compared to the other methods.

Figure B.8 shows the 90% coverages, and Figure B.9 shows the corresponding 90% width of the intervals. Similar to the MNAR case, GP-OM produces a high coverage in the borehole function, but near-zero coverages in the others. The corresponding 90% widths reflect the high coverage with wide intervals and the undercoverages with intervals that are too narrow. The EMGP method achieves adequate coverage for the piston and OTLcircuit functions, but undercovers in the borehole function. The coverage for the wingweight function improves as N increases and achieves the prescribed level at larger Ns. All the other methods achieve adequate coverages, where the corresponding widths decrease as N increases. PCGPwM generally produces narrower widths than do the simplistic imputation methods while attaining comparable coverages.

Figure B.10 shows the test RMSEs under 5% MCAR. Figures B.11 and B.12 show the 90% coverages and the 90% widths, respectively. The results of the surrogate accuracy with 1% MCAR extends to the scenario with 5% MCAR, except the levels of coverage for PCGPwM has decreased in large N. The 90% widths decrease for all methods as N increases.

Figure B.13 shows the test RMSEs under 25% MCAR. The RMSEs for all the methods generally decrease with N. Figures B.14 and B.15 show the 90% coverages and the 90% widths, respectively. Similar conclusions can be drawn for GP-OM, compared to lower missingness scenarios. The simplistic imputation methods maintain adequate coverages while maintaining slightly wider intervals when compared with PCGPwM. The EMGP method exhibits similar behavior, where coverage improves with the borehold and wingweight functions as N increases.

A.16. Results under MAR

The results under MAR are presented, in the order of 1%, 5%, and 25%. The numerical results under MAR follow similar trends as MCAR for all percentages of missingness. We refer the analysis to the reporting of results in the MCAR section. Figures B.16, B.19, and B.22 present the corresponding RMSEs. Figures B.17, B.20, and B.23 present the 90% coverages. And Figures B.18, B.21, and B.24 present the 90% widths.

A.17. Comparison of PCGPwM against a baseline method

We compare PCGPwM against a baseline method where we build the principal components as if the complete data are provided (i.e., $\mathbf{\Phi}$ and $\mathbf{\Lambda}$). We label it the "PCGP-benchmark" method. The purpose of including this method is to examine the significance of the estimation of the principal components from missing data. Although improved estimation of principal components from missing data is outside the scope of this chapter, by studying this method we can offer insights into the limitations of the proposed method.

The two methods are compared under the 5% MNAR scenario. The results in terms of RMSE, 90% coverage, and the 90% interval width show that PCGPwM performs near

identically to PCGP-benchmark, thus the figures are left out. In other words, the proposed imputation recovers the unknown principal components well. In certain cases, when compared with colGP (which uses the data without dimension reduction), both PCGPwM and PCGP-benchmark cannot further reduce error even with larger data size. This points to a potential limitation in the use of principal component methods.

A.18. Scaling of Fayans EDF parameter space

Table A.5 contains the unscaled centroid, the length scales, and the lower and upper bounds for each dimension of the parameter. This information is reproduced from Table 5 of Bollapragada et al. [21].

Parameter	unscaled center	scale	lb	ub
ρ_{eq}	0.1642	0.004	0.146	0.167
E/A	-15.86	0.1	-16.21	-15.50
K	206.6	25	137.2	234.4
J	28.3	3.2	19.5	37.0
L	35.9	32	2.2	69.6
h^{v}_{2-}	11.34	19.01	0	100
a^{s}_{+}	0.562	0.06	0.418	0.706
h^{s}_{∇}	0.460	0.24	0	0.516
κ	0.188	0.02	0.076	0.216
κ'	0.045	0.17	-0.892	0.982
f^{ξ}_{ex}	-4.46	1.16	-4.62	-4.38
h^{ξ}_{∇}	4.18	1.68	3.94	4.27
h^{ξ}_{+}	3.44	1.4	-0.96	3.66

Table A.5. Fayans EDF model parameters and their scaling information, reproduced from Table 5 of Bollapragada et al. [21].

APPENDIX B

Additional figures from PCGPwM numerical experiments

This appendix contains additional figures from numerical experiments in Chapter 2.

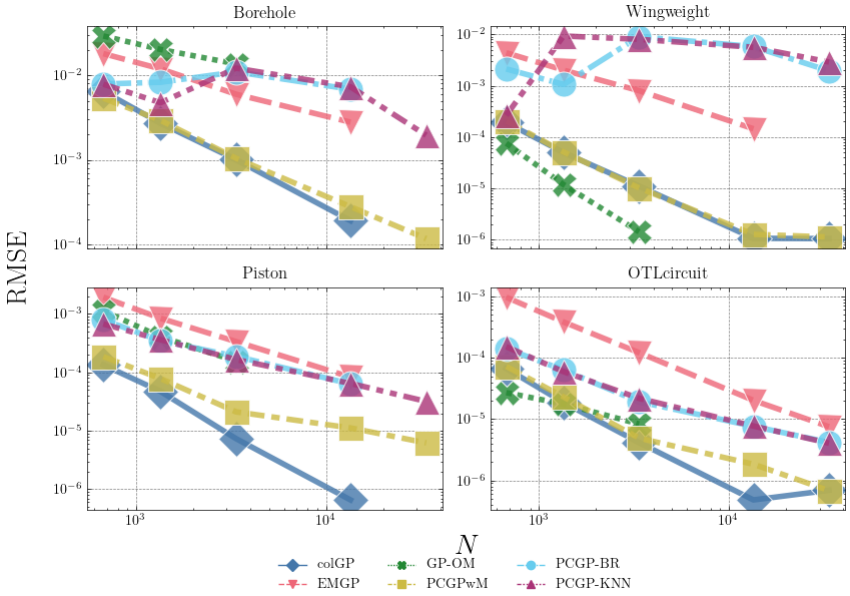


Figure B.1. Comparison of prediction accuracy of surrogate methods, under 1% MNAR.

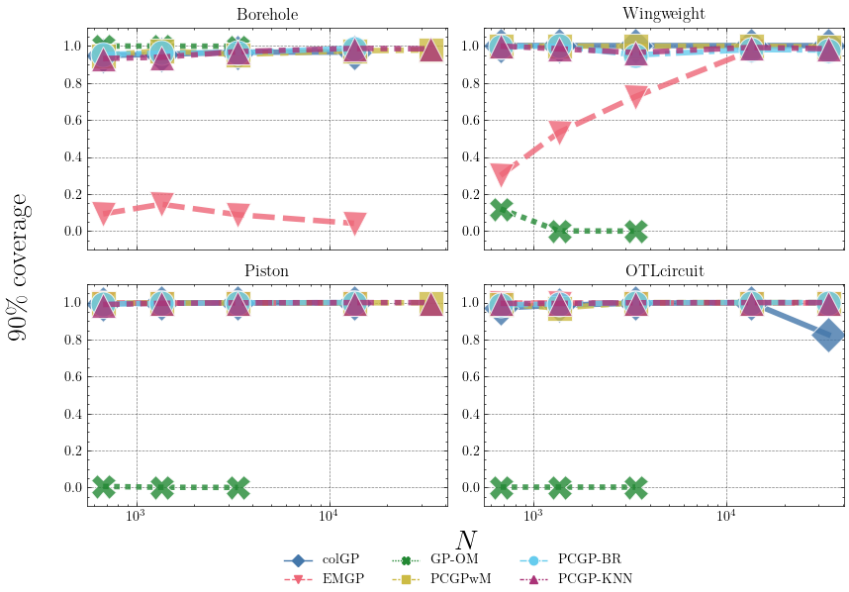


Figure B.2. Comparison of 90% coverage of surrogate methods, under 1% MNAR.

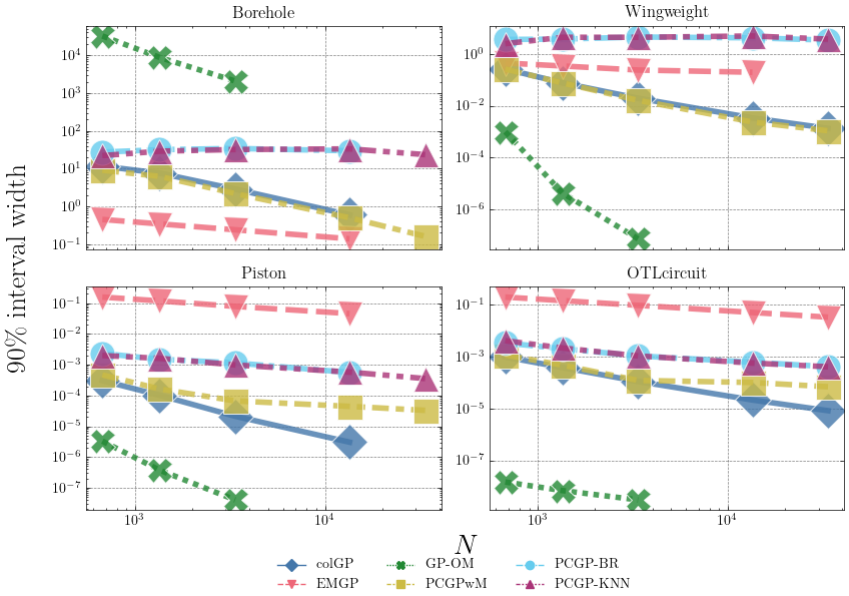


Figure B.3. Comparison of 90% width of surrogate methods, under 1% MNAR.

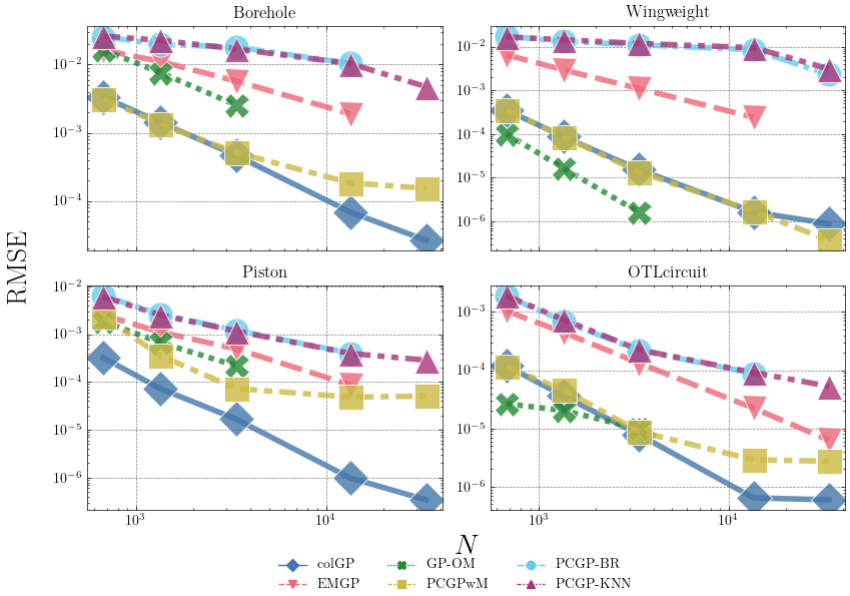


Figure B.4. Comparison of prediction accuracy of surrogate methods, under 25% MNAR.

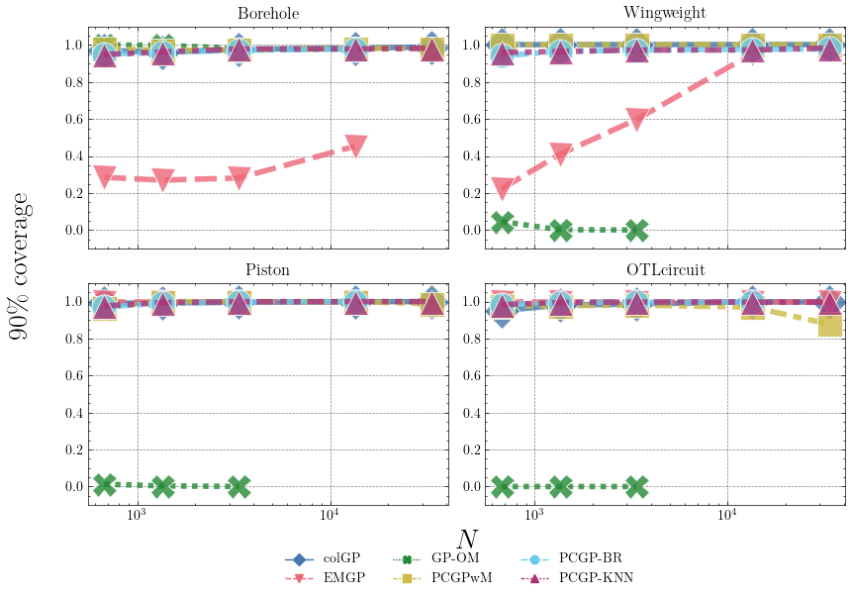


Figure B.5. Comparison of 90% coverage of surrogate methods, under 25% MNAR.

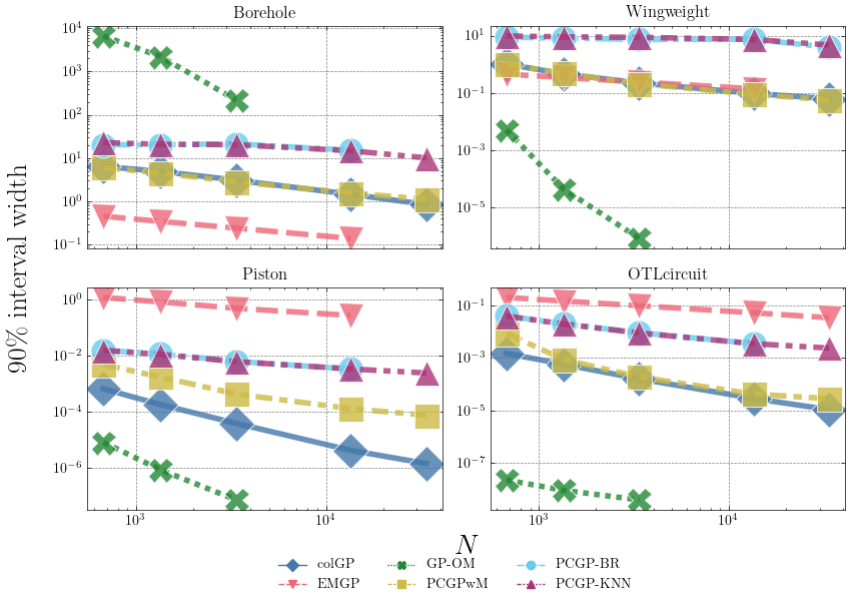


Figure B.6. Comparison of 90% width of surrogate methods, under 25% MNAR.

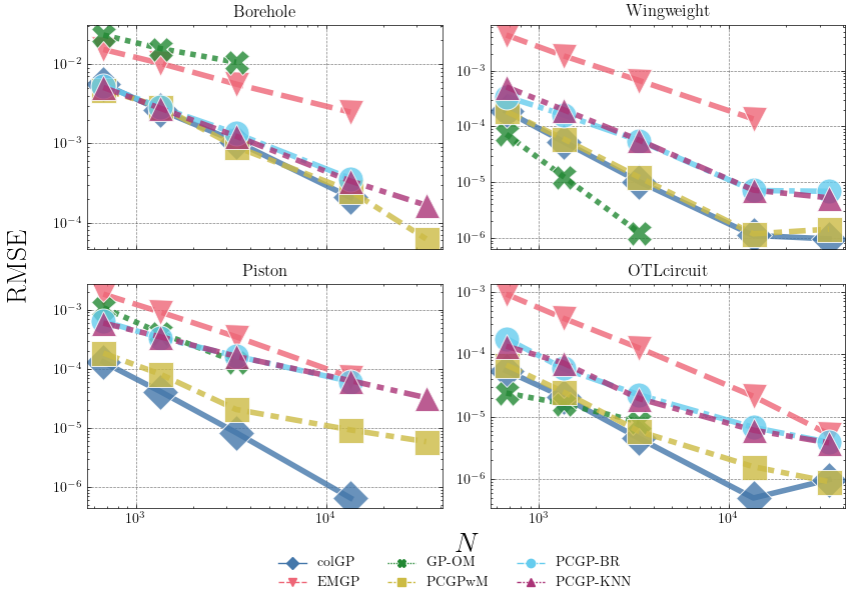


Figure B.7. Comparison of prediction accuracy of surrogate methods, under 1% MCAR.

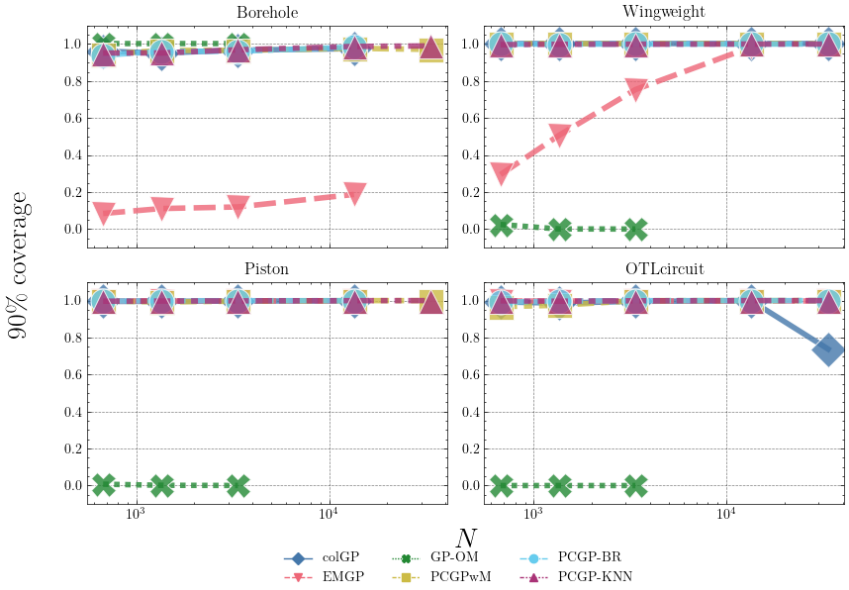


Figure B.8. Comparison of 90% coverage of surrogate methods, under 1% MCAR.

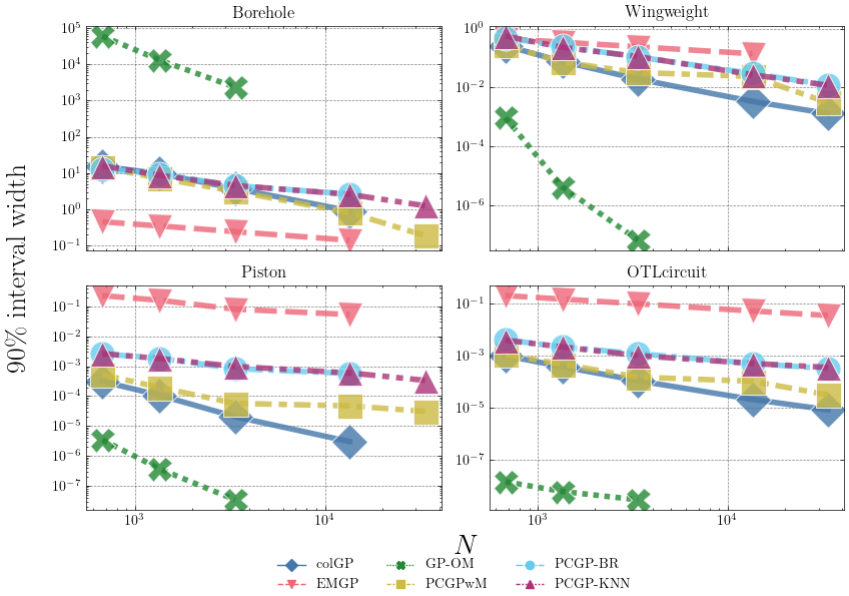


Figure B.9. Comparison of 90% width of surrogate methods, under 1% MCAR.

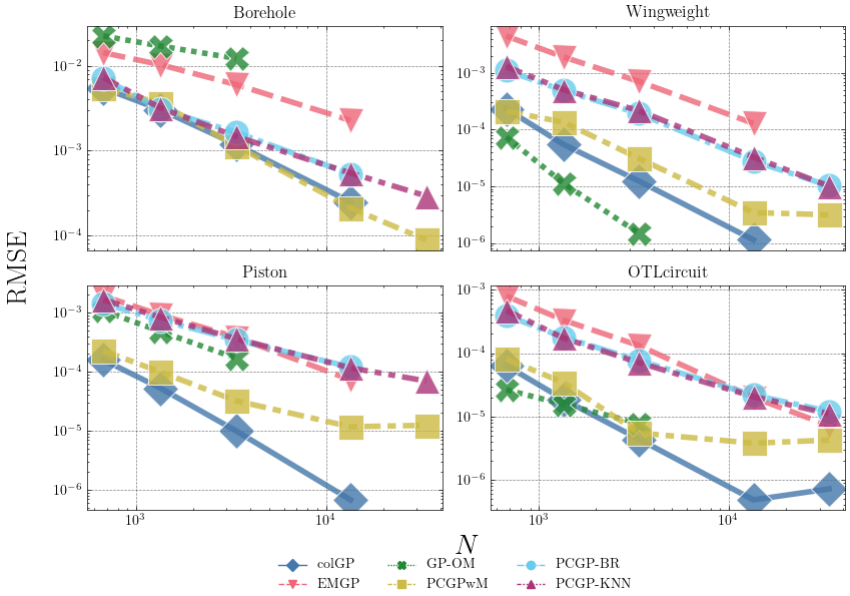


Figure B.10. Comparison of prediction accuracy of surrogate methods, under 5% MCAR.

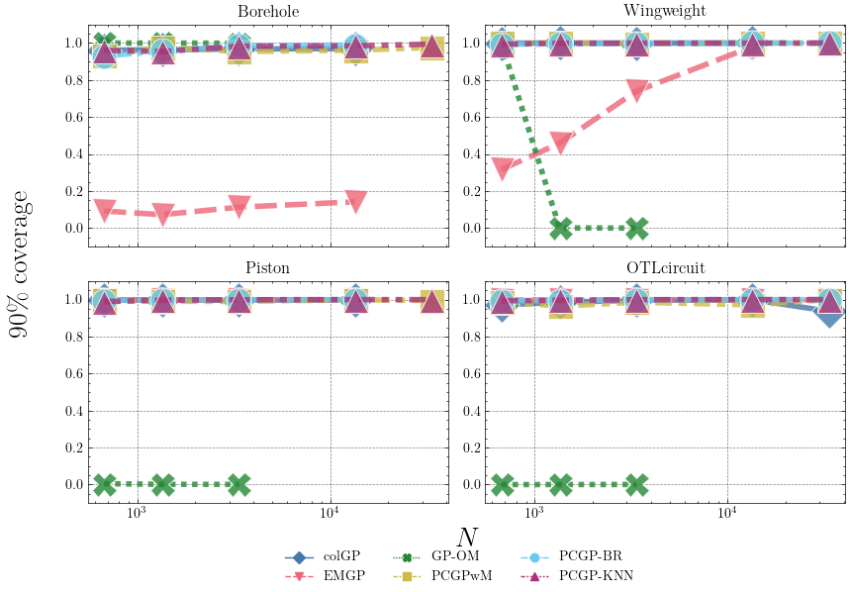


Figure B.11. Comparison of 90% coverage of surrogate methods, under 5% MCAR.

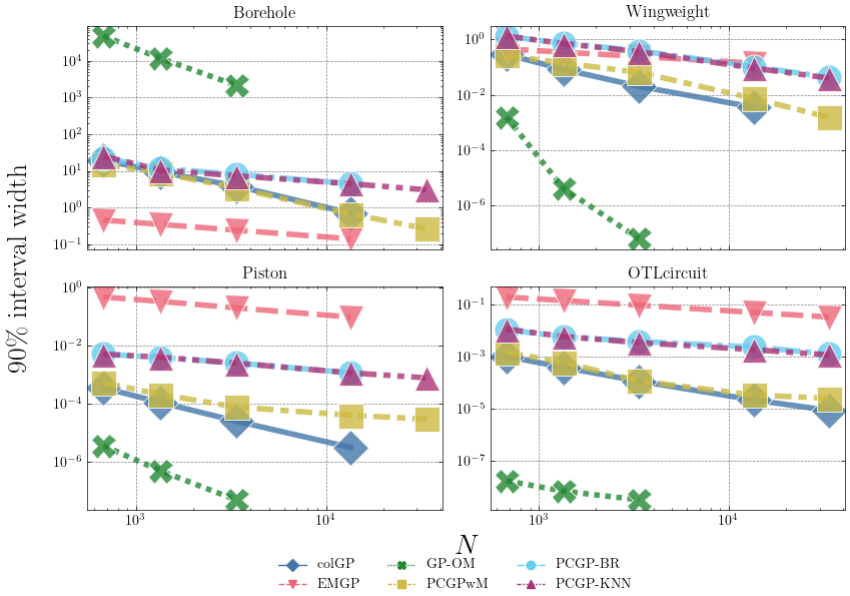


Figure B.12. Comparison of 90% width of surrogate methods, under 5% MCAR.

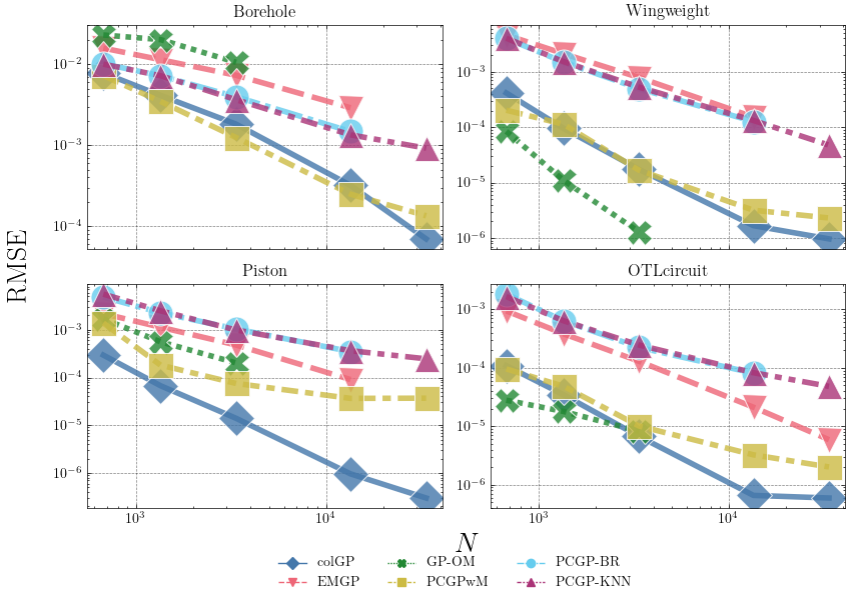


Figure B.13. Comparison of prediction accuracy of surrogate methods, under 25% MCAR.

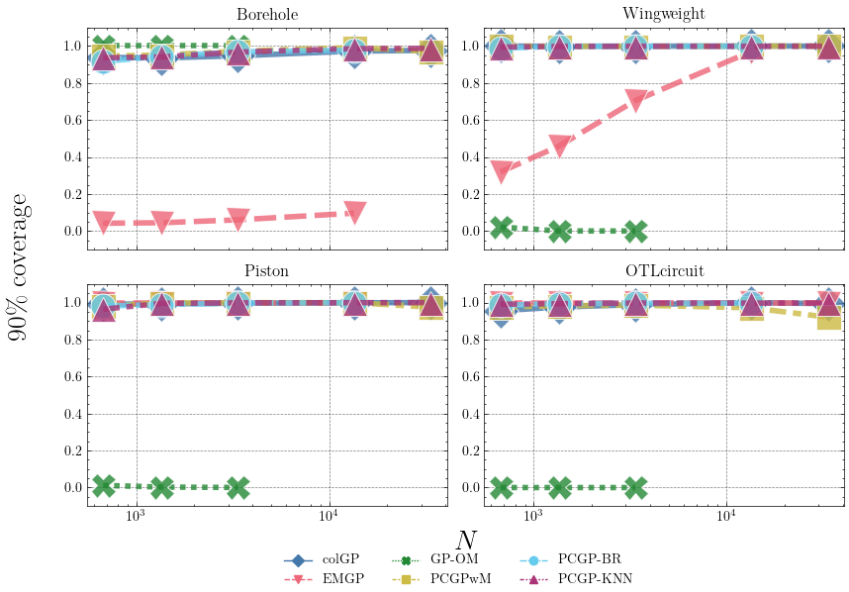


Figure B.14. Comparison of 90% coverage of surrogate methods, under 25% MCAR.

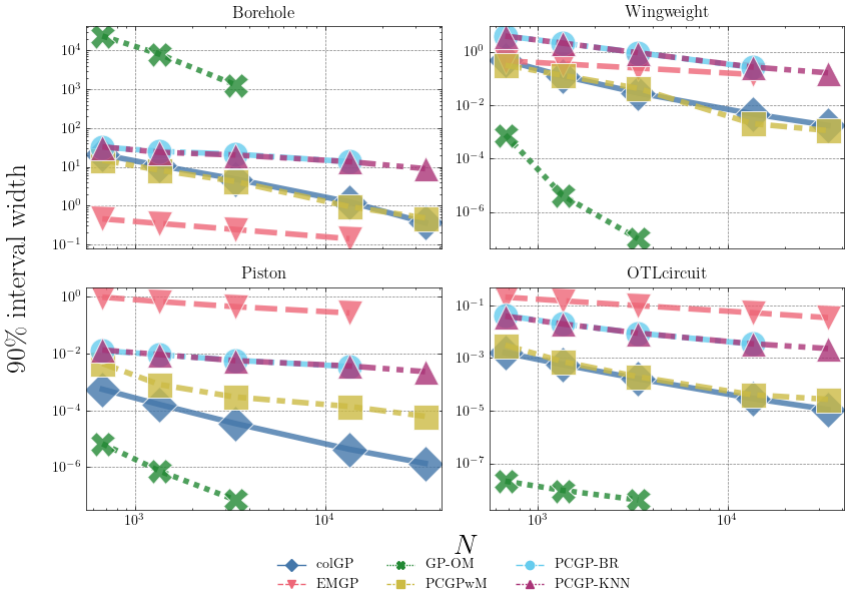


Figure B.15. Comparison of 90% width of surrogate methods, under 25% MCAR.

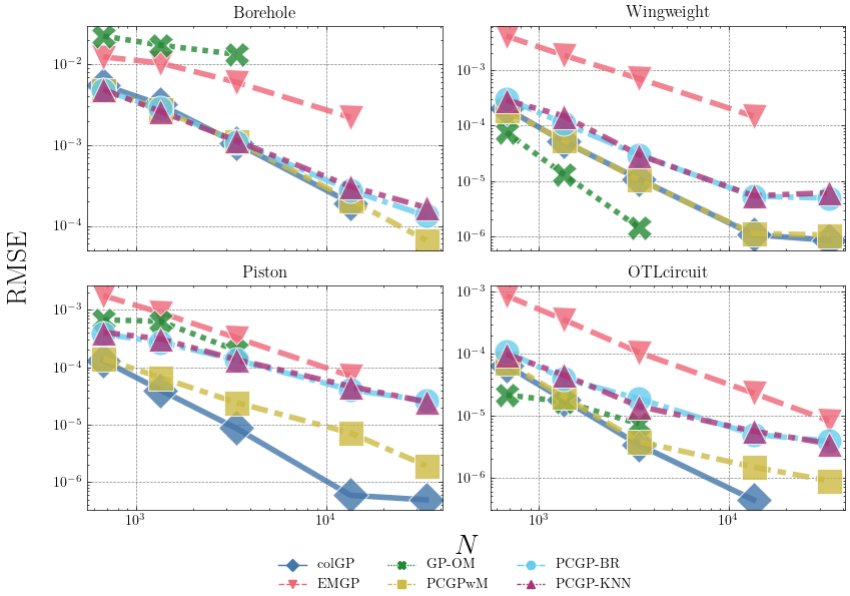


Figure B.16. Comparison of prediction accuracy of surrogate methods, under 1% MAR.

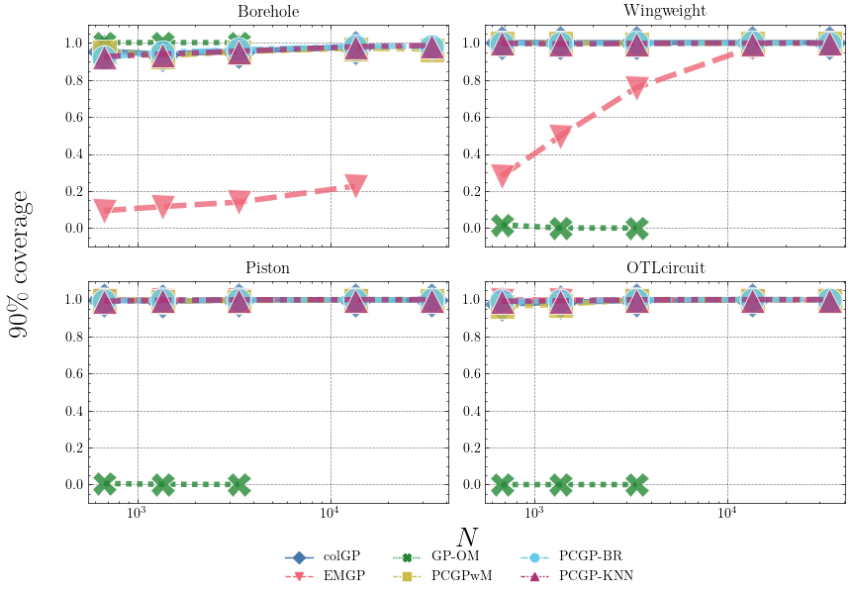


Figure B.17. Comparison of 90% coverage of surrogate methods, under 1% MAR.

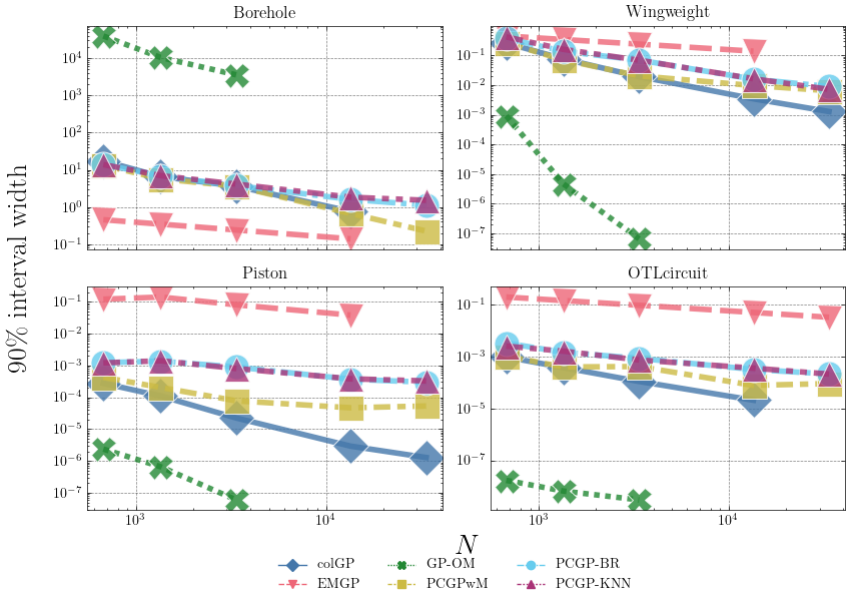


Figure B.18. Comparison of 90% width of surrogate methods, under 1% MAR.

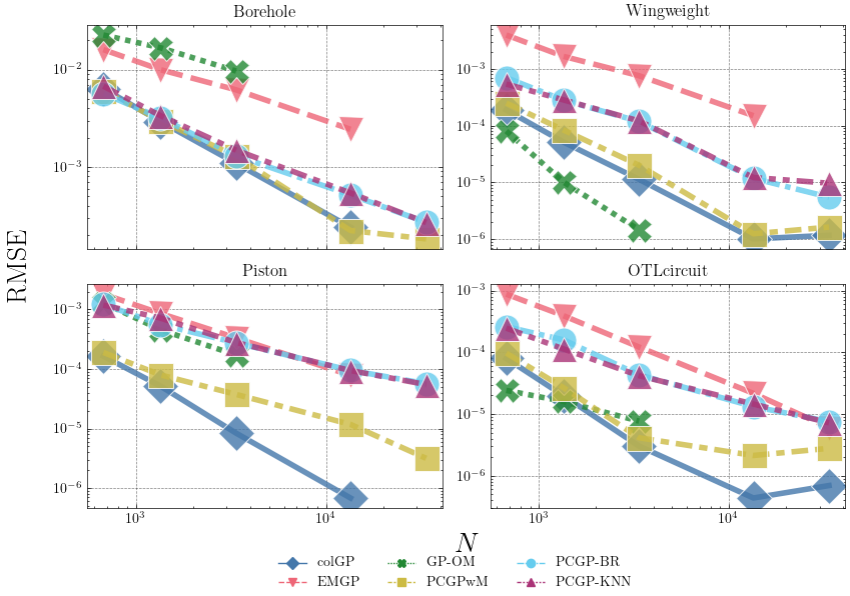


Figure B.19. Comparison of prediction accuracy of surrogate methods, under 5% MAR.

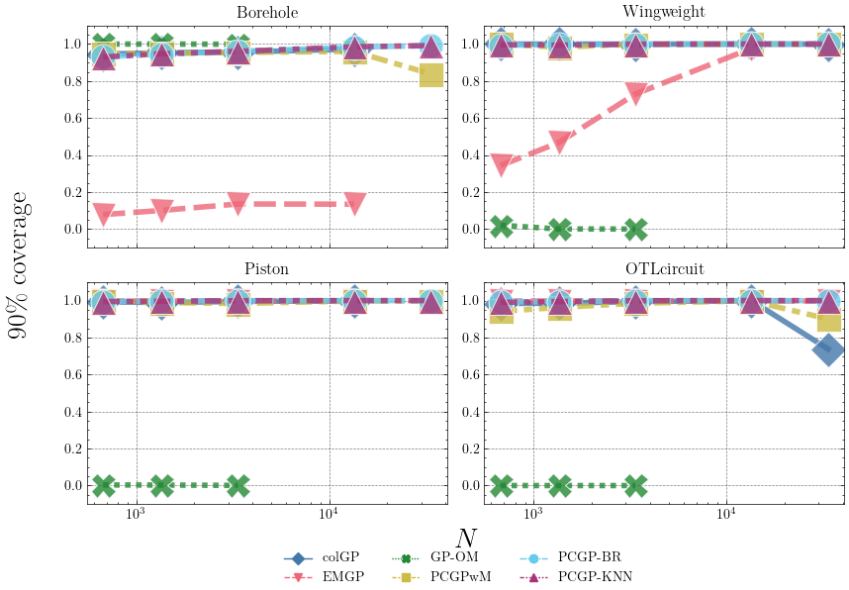


Figure B.20. Comparison of 90% coverage of surrogate methods, under 5% MAR.

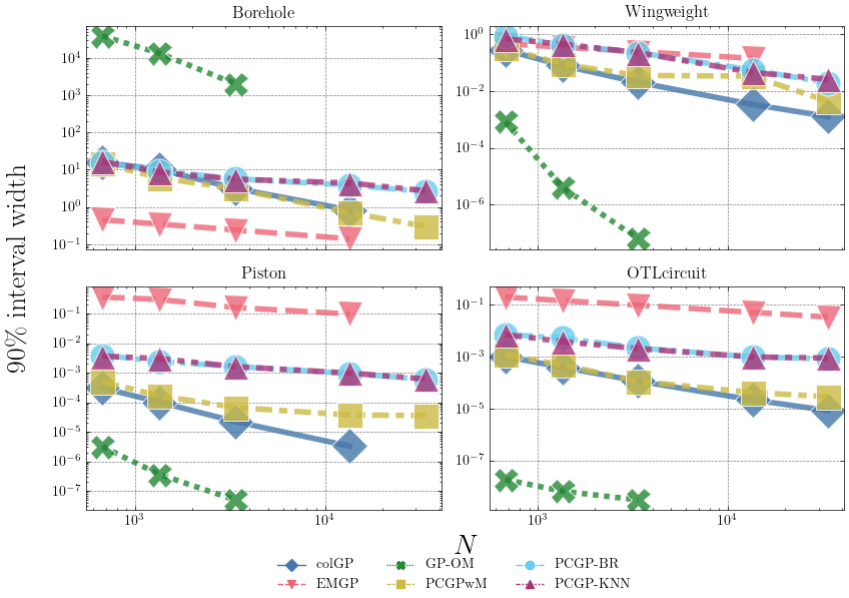


Figure B.21. Comparison of 90% width of surrogate methods, under 5% MAR.

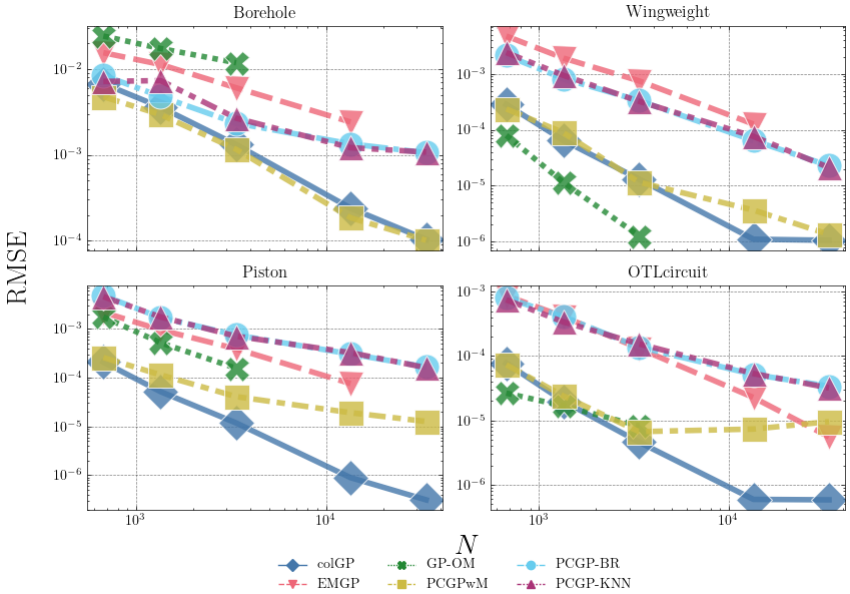


Figure B.22. Comparison of prediction accuracy of surrogate methods, under 25% MAR.

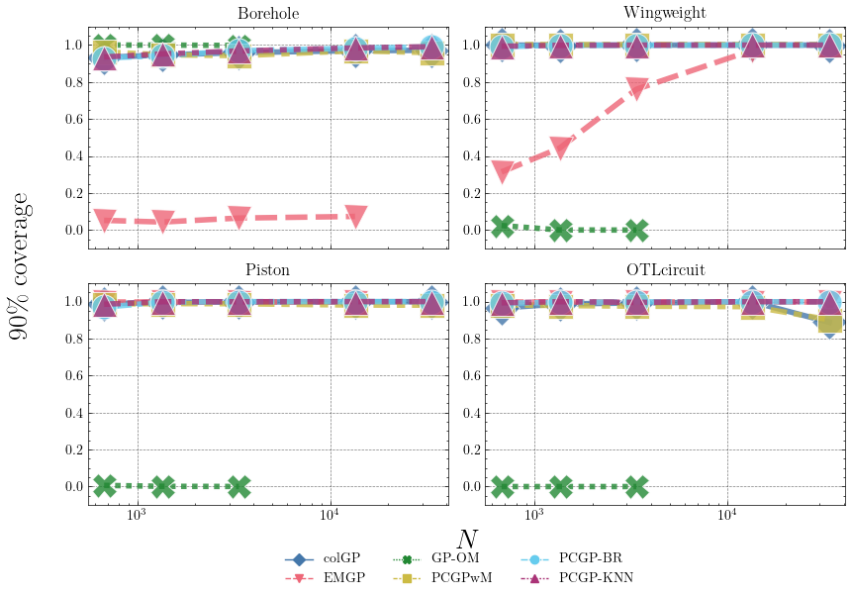


Figure B.23. Comparison of 90% coverage of surrogate methods, under 25% MAR.

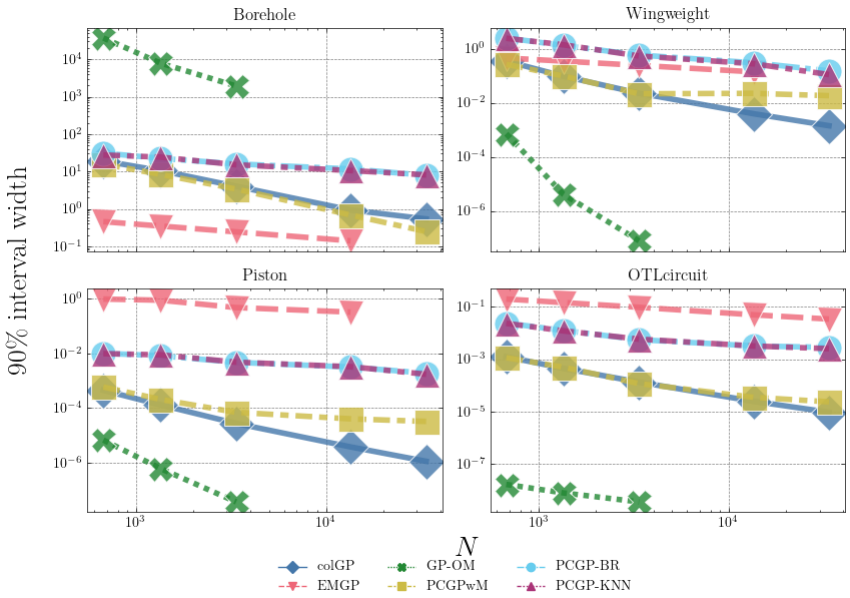


Figure B.24. Comparison of 90% width of surrogate methods, under 25% MAR.

APPENDIX C

Proofs and computational details for LCGP

C.1. Proofs of LCGP results

This section contains proofs to the results presented alongside LCGP.

C.1.1. Proof of Theorem 3.1.

PROOF. Note that the posterior of $\text{vec}(\boldsymbol{G})$, proportional to the product of two Gaussian distributions, is another Gaussian distribution

$$
\begin{aligned}
p(\boldsymbol{G} \mid \boldsymbol{Y}, \boldsymbol{\Sigma}, \boldsymbol{\Theta}) &\propto p(\boldsymbol{Y} \mid \boldsymbol{G}, \boldsymbol{\Sigma}, \boldsymbol{\Theta}) p(\boldsymbol{G} \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta}) \\
&= (2\pi)^{-\frac{np}{2}} |\boldsymbol{\Sigma}|^{-\frac{n}{2}} \exp\left\{ -\frac{1}{2} \text{vec}(\boldsymbol{Y} - \boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{T}})^{\mathsf{T}} (\boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_n) \text{vec}(\boldsymbol{Y} - \boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{T}}) \right\} \\
&\quad \times (2\pi)^{-\frac{nq}{2}} |\mathbf{C}|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} \text{vec}(\boldsymbol{G})^{\mathsf{T}} \mathbf{C}^{-1} \text{vec}(\boldsymbol{G}) \right\} \\
&= (2\pi)^{-\frac{n(p+q)}{2}} |\boldsymbol{\Sigma}|^{-\frac{n}{2}} |\mathbf{C}|^{-\frac{1}{2}} \exp\Bigg\{ -\frac{1}{2} \text{vec}(\boldsymbol{Y})^{\mathsf{T}} (\boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_n) \text{vec}(\boldsymbol{Y}) \\
&\quad + \text{vec}(\boldsymbol{G})^{\mathsf{T}} (\boldsymbol{\Psi}^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_n) \text{vec}(\boldsymbol{Y}) \\
&\quad - \frac{1}{2} \text{vec}(\boldsymbol{G})^{\mathsf{T}} (\boldsymbol{\Psi}^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{\Psi} \otimes \boldsymbol{I}_n + \mathbf{C}^{-1}) \text{vec}(\boldsymbol{G}) \Bigg\}.
\end{aligned} \tag{C.1}
$$

Letting $\boldsymbol{\Psi} = \boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{\Phi}$, isolating the quadratic and linear terms with respect to $\text{vec}(\boldsymbol{G})$ yields

$$p(\boldsymbol{G} \mid \boldsymbol{Y}, \boldsymbol{\Sigma}, \boldsymbol{\Theta})$$

$$
\begin{aligned}
=&(2\pi)^{-\frac{n(p+q)}{2}}|\boldsymbol{\Sigma}|^{-\frac{n}{2}}|\mathbf{C}|^{-\frac{1}{2}}\exp\Bigg\{-\frac{1}{2}\mathrm{vec}(\boldsymbol{Y})^{\mathsf{T}}(\boldsymbol{\Sigma}^{-1}\otimes\boldsymbol{I}_n)\mathrm{vec}(\boldsymbol{Y})\\
&+\mathrm{vec}(\boldsymbol{G})^{\mathsf{T}}(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Sigma}^{-\frac{1}{2}}\otimes\boldsymbol{I}_n)\mathrm{vec}(\boldsymbol{Y})\\
&-\frac{1}{2}\mathrm{vec}(\boldsymbol{G})^{\mathsf{T}}(\boldsymbol{D}\otimes\boldsymbol{I}_n+\mathbf{C}^{-1})\mathrm{vec}(\boldsymbol{G})\Bigg\}.
\end{aligned}
$$

□

C.1.2. Proof of Theorem 3.2.

PROOF. Denote $\boldsymbol{A}=\boldsymbol{D}\otimes\boldsymbol{I}_n+\mathbf{C}^{-1}, \boldsymbol{b}=(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Sigma}^{-\frac{1}{2}}\otimes\boldsymbol{I}_n)\mathrm{vec}(\boldsymbol{Y})$. From the results in Theorem 3.1, the marginal distribution of $\mathrm{vec}(\boldsymbol{Y})$ can be obtained from integrating out the latent components, i.e.,

$$
\begin{aligned}
p(\boldsymbol{Y}\mid\boldsymbol{\Sigma},\boldsymbol{\Theta})&=\int_{\mathbb{R}^{nq}}p(\boldsymbol{Y},\boldsymbol{G}\mid\boldsymbol{\Sigma},\boldsymbol{\Theta})\mathrm{dvec}(\boldsymbol{G})\\
&=(2\pi)^{-\frac{n(p+q)}{2}}|\boldsymbol{\Sigma}|^{-\frac{n}{2}}|\mathbf{C}|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}\mathrm{vec}(\boldsymbol{Y})^{\mathsf{T}}(\boldsymbol{\Sigma}^{-1}\otimes\boldsymbol{I}_n)\mathrm{vec}(\boldsymbol{Y})\right\}\\
&\quad\times\int_{\mathbb{R}^{nq}}\exp\left\{-\frac{1}{2}\mathrm{vec}(\boldsymbol{G})^{\mathsf{T}}\boldsymbol{A}\mathrm{vec}(\boldsymbol{G})+\mathrm{vec}(\boldsymbol{G})^{\mathsf{T}}\boldsymbol{b}\right\}\mathrm{dvec}(\boldsymbol{G})\\
&=(2\pi)^{-\frac{np}{2}}|\boldsymbol{\Sigma}|^{-\frac{n}{2}}|\mathbf{C}|^{-\frac{1}{2}}|\boldsymbol{A}|^{-\frac{1}{2}}\times\exp\Bigg\{\\
&-\frac{1}{2}\mathrm{vec}(\boldsymbol{Y})^{\mathsf{T}}\Big(\boldsymbol{\Sigma}^{-1}\otimes\boldsymbol{I}_n-(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Phi}\otimes\boldsymbol{I}_n)\boldsymbol{A}^{-1}(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Sigma}^{-\frac{1}{2}}\otimes\boldsymbol{I}_n)\Big)\mathrm{vec}(\boldsymbol{Y})\Bigg\}.
\end{aligned}
$$

The final equality follows the results from evaluating a multiple integral of an exponentiated quadratic form [70, ch.15], with

$$
\int_{\mathbb{R}^{nq}}\exp\left\{-\frac{1}{2}(\mathrm{vec}(\boldsymbol{G})-\boldsymbol{A}^{-1}\boldsymbol{b})^{\mathsf{T}}\boldsymbol{A}(\mathrm{vec}(\boldsymbol{G})-\boldsymbol{A}^{-1}\boldsymbol{b})\right\}\mathrm{dvec}(\boldsymbol{G})=(2\pi)^{\frac{nq}{2}}|\boldsymbol{A}|^{-\frac{1}{2}}.
$$

□

C.1.3. Proof of Theorem 3.5.

Proof. Each of the three terms in the ELBO function, Equation (3.26) is evaluated. Recall that all expectations are taken with respect to $q(\cdot)$, the variational density. The first term is

$$
\begin{aligned}
&\mathbb{E}\log p(\boldsymbol{Y} \mid \boldsymbol{G}) \\
=&\mathbb{E}\left[-\frac{np}{2}\log(2\pi) - \frac{n}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\mathrm{vec}(\boldsymbol{Y}-\boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{T}})^{\mathsf{T}}(\boldsymbol{\Sigma}^{-1}\otimes\boldsymbol{I}_n)\mathrm{vec}(\boldsymbol{Y}-\boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{T}})\right] \\
=&c - \frac{n}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\mathrm{vec}(\boldsymbol{Y}-\boldsymbol{M}\boldsymbol{\Psi}^{\mathsf{T}})^{\mathsf{T}}(\boldsymbol{\Sigma}^{-1}\otimes\boldsymbol{I}_n)\mathrm{vec}(\boldsymbol{Y}-\boldsymbol{M}\boldsymbol{\Psi}^{\mathsf{T}}) \\
&- \frac{1}{2}\mathrm{tr}\left((\boldsymbol{D}\otimes\boldsymbol{I}_n)\mathrm{diag}(\mathrm{vec}(\boldsymbol{V}))\right) \\
=&c - \frac{n}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\mathrm{vec}(\boldsymbol{Y}-\boldsymbol{M}\boldsymbol{\Psi}^{\mathsf{T}})^{\mathsf{T}}(\boldsymbol{\Sigma}^{-1}\otimes\boldsymbol{I}_n)\mathrm{vec}(\boldsymbol{Y}-\boldsymbol{M}\boldsymbol{\Psi}^{\mathsf{T}}) - \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{q} d_k V_{ik}.
\end{aligned}
$$

The second term is

$$
\begin{aligned}
\mathbb{E}\log p(\boldsymbol{G}) =& \sum_{k=1}^{q}\mathbb{E}\left[-\frac{1}{2}\log|\boldsymbol{C}_k| - \frac{1}{2}\boldsymbol{G}_k^{\mathsf{T}}\boldsymbol{C}_k^{-1}\boldsymbol{G}_k\right] + c \\
=& \sum_{k=1}^{q}\left(-\frac{1}{2}\log|\boldsymbol{C}_k| - \frac{1}{2}\left(\boldsymbol{M}_k^{\mathsf{T}}\boldsymbol{C}_k^{-1}\boldsymbol{M}_k + \mathrm{tr}(\boldsymbol{C}_k^{-1}\mathrm{diag}(\boldsymbol{V}_k))\right)\right) + c \\
=& -\frac{1}{2}\log|\mathbf{C}| - \frac{1}{2}\mathrm{vec}(\boldsymbol{M})^{\mathsf{T}}\mathbf{C}^{-1}\mathrm{vec}(\boldsymbol{M}) - \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{q}(\boldsymbol{C}_k^{-1})_{ii}V_{ik} + c.
\end{aligned}
$$

The final term is

$$
\begin{aligned}
\mathbb{E}\log q(\boldsymbol{G}) &= \mathbb{E}\left[-\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{q}\log(2\pi V_{ik}) - \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{q}\frac{(G_{ik}-M_{ik})^2}{V_{ik}}\right] \\
&= -\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{q}\log(V_{ik}) + c.
\end{aligned}
$$

We reach the results by combining the terms. □

C.1.4. Proof of Theorem 3.3.

Proof. For the kth latent component, by the principles of conditional expectations and variances [160, ch.3],

$$\mathbb{E}[g_k(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\Sigma}, \boldsymbol{\theta}_k] = \mathbb{E}[\mathbb{E}[g_k(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{G}_k; \boldsymbol{\Sigma}, \boldsymbol{\theta}_k]] = \mathbb{E}[c_k^{\mathsf{T}}(\boldsymbol{x})\boldsymbol{C}_k^{-1}\boldsymbol{G}_k \mid \boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\Sigma}, \boldsymbol{\theta}_k].$$

$$\begin{aligned}\mathbb{V}[g_k(\boldsymbol{x}) \mid \boldsymbol{X}; \boldsymbol{\Sigma}, \boldsymbol{\theta}_k] &= \mathbb{V}[\mathbb{E}[g_k(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{G}_k; \boldsymbol{\Sigma}, \boldsymbol{\theta}_k]] + \mathbb{E}[\mathbb{V}[g_k(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{G}_k; \boldsymbol{\Sigma}, \boldsymbol{\theta}_k]] \\ &= c_k^{\mathsf{T}}(\boldsymbol{x})\boldsymbol{C}_k^{-1}\boldsymbol{S}_k\boldsymbol{C}_k^{-1}c_k^{\mathsf{T}}(\boldsymbol{x}) + c_k(\boldsymbol{x}, \boldsymbol{x}) - c_k^{\mathsf{T}}(\boldsymbol{x})\boldsymbol{C}_k^{-1}c_k^{\mathsf{T}}(\boldsymbol{x}).\end{aligned}$$

□

C.1.5. Proof of Theorem 3.4.

Proof. Collecting the latent component predictions as $\boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), \ldots, g_q(\boldsymbol{x}))^{\mathsf{T}}$,

$$\mathbb{E}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{Y}; \boldsymbol{\Sigma}, \boldsymbol{\Theta}] = \mathbb{E}[\mathbb{E}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{Y}, \boldsymbol{g}(\boldsymbol{x}); \boldsymbol{\Sigma}, \boldsymbol{\Theta}]] = \mathbb{E}[\boldsymbol{\Psi}\boldsymbol{g}(\boldsymbol{x}) \mid \boldsymbol{Y}; \boldsymbol{\Sigma}, \boldsymbol{\Theta}].$$

$$\begin{aligned}\mathbb{V}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta}] &= \mathbb{V}[\mathbb{E}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{g}(\boldsymbol{x}); \boldsymbol{\Sigma}, \boldsymbol{\Theta}]] + \mathbb{E}[\mathbb{V}[\boldsymbol{y}(\boldsymbol{x}) \mid \boldsymbol{X}, \boldsymbol{g}(\boldsymbol{x}); \boldsymbol{\Sigma}, \boldsymbol{\Theta}]] \\ &= \mathbb{V}[\boldsymbol{\Psi}\boldsymbol{g}(\boldsymbol{x}) \mid \boldsymbol{X}; \boldsymbol{\Sigma}, \boldsymbol{\Theta}] + \boldsymbol{\Sigma}.\end{aligned}$$

The results follow by evaluating the expectations. □

C.1.6. Proof of Theorem 3.6.

Proof. Isolating the ELBO function with respect to $\boldsymbol{M}$,

$$\mathsf{ELBO}(\boldsymbol{M}) = -\frac{1}{2}\left(\mathrm{vec}(\boldsymbol{Y} - \boldsymbol{M}\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Sigma}^{\frac{1}{2}})^{\mathsf{T}}(\boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_n)\mathrm{vec}(\boldsymbol{Y} - \boldsymbol{M}\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Sigma}^{\frac{1}{2}})\right.$$

$$
\begin{aligned}
&- \frac{1}{2}\mathrm{vec}(\boldsymbol{M})^{\mathsf{T}}\mathbf{C}^{-1}\mathrm{vec}(\boldsymbol{M}) \\
&= c - \frac{1}{2}\mathrm{vec}(\boldsymbol{M})^{\mathsf{T}}(\boldsymbol{D}\otimes\boldsymbol{I}_n + \mathbf{C}^{-1})\mathrm{vec}(\boldsymbol{M}) + \mathrm{vec}(\boldsymbol{Y})(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Phi}\otimes\boldsymbol{I}_n)\mathrm{vec}(\boldsymbol{M}) \\
&= c - \frac{1}{2}\mathrm{vec}(\boldsymbol{M})^{\mathsf{T}}\boldsymbol{A}_M\mathrm{vec}(\boldsymbol{M}) + \boldsymbol{b}_M^{\mathsf{T}}\mathrm{vec}(\boldsymbol{M}),
\end{aligned}
$$

resulting in a quadratic form in $\mathrm{vec}(\boldsymbol{M})$. Since $\boldsymbol{A}_M = \boldsymbol{D}\otimes\boldsymbol{I}_n + \mathbf{C}^{-1}$ is positive definite, $\mathrm{vec}(\widehat{\boldsymbol{M}}) = \boldsymbol{A}_M^{-1}\boldsymbol{b}_M$ maximizes $\mathsf{ELBO}(\boldsymbol{M})$. With respect to $\boldsymbol{V}$,

$$
\mathsf{ELBO}(\boldsymbol{V}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{q}\left(\log V_{ik} - \left[d_k + (\boldsymbol{C}_k^{-1})_{ii}\right]V_{ik}\right).
$$

Therefore,

$$
0 = \frac{\partial}{\partial \boldsymbol{V}}\mathsf{ELBO}(\boldsymbol{V}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{q}\left(\frac{1}{V_{ik}} - \left[d_k + (\boldsymbol{C}_k^{-1})_{ii}\right]\right).
$$

Solving for V_{ik} for each term leads to the result. □

C.2. Closely related determinants and inverses

The eigendecomposition of a symmetric matrix $\boldsymbol{A}$ is $\boldsymbol{U}\boldsymbol{W}\boldsymbol{U}^{\mathsf{T}}$, for some orthogonal $\boldsymbol{U}$ and diagonal $\boldsymbol{W}$. If we obtain the eigendecomposition of $\boldsymbol{C}_k$ as $\boldsymbol{C}_k = \boldsymbol{U}_k\boldsymbol{W}_k\boldsymbol{U}_k^{\mathsf{T}}$, then $\boldsymbol{C}_k^{-1} = \boldsymbol{U}_k\boldsymbol{W}_k^{-1}\boldsymbol{U}_k^{\mathsf{T}}$, and the matrix $\boldsymbol{I}_n + d_k\boldsymbol{C}_k$ has inverse

$$
(\boldsymbol{I}_n + d_k\boldsymbol{C}_k)^{-1} = (\boldsymbol{I}_n + d_k\boldsymbol{U}_k\boldsymbol{W}_k\boldsymbol{U}_k^{\mathsf{T}})^{-1} = \boldsymbol{U}_k(\boldsymbol{I}_n + d_k\boldsymbol{W}_k)^{-1}\boldsymbol{U}_k^{\mathsf{T}},
$$

and determinant

$$
|\boldsymbol{I}_n + d_k\boldsymbol{C}_k| = |\boldsymbol{I}_n + \boldsymbol{U}_k(d_k\boldsymbol{W}_k)\boldsymbol{U}_k^{\mathsf{T}}| = |\boldsymbol{I}_n||d_k\boldsymbol{W}_k||d_k^{-1}\boldsymbol{W}_k^{-1} + \boldsymbol{I}_n| = \prod_{i=1}^{n}(1 + d_kW_{ik}).
$$

Additionally, the matrix inverse for $d_k \boldsymbol{I}_n + \boldsymbol{C}_k^{-1}$ is

$$(d_k \boldsymbol{I}_n + \boldsymbol{C}_k^{-1})^{-1} = \boldsymbol{U}_k (d_k \boldsymbol{I}_n + \boldsymbol{W}_k^{-1})^{-1} \boldsymbol{U}_k^{\mathsf{T}}.$$

C.3. Storage of auxiliary quantities

We note that some auxiliary quantities in the predictive equations remain unchanged, given fixed hyperparameters, e.g., $\boldsymbol{C}_k^{-1}\boldsymbol{m}_k$ in Equation (3.22), and $\boldsymbol{T}_k$ in Equation (3.23). For efficiency in computation, these quantities are stored and reused for predictions. The two quantities are further simplified to

$$\boldsymbol{C}_k^{-1}\boldsymbol{m}_k = \boldsymbol{C}_k^{-1}(d_k \boldsymbol{I}_n + \boldsymbol{C}_k^{-1})^{-1} \boldsymbol{Y} \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\phi}_k = (\boldsymbol{I}_n + d_k \boldsymbol{C}_k)^{-1} \boldsymbol{Y} \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\phi}_k,$$

$$\boldsymbol{T}_k = \boldsymbol{C}_k^{-1} - \boldsymbol{C}_k^{-1}(d_k \boldsymbol{I}_n + \boldsymbol{C}_k^{-1})^{-1} \boldsymbol{C}_k^{-1} = \boldsymbol{C}_k^{-1} \left(\boldsymbol{I}_n - (\boldsymbol{I}_n + d_k \boldsymbol{C}_k)^{-1} \right).$$

Specifically, it is advantageous to compute $\boldsymbol{T}_k$ from

$$\begin{aligned}
\boldsymbol{T}_k &= \boldsymbol{C}_k^{-1} \left(\boldsymbol{I}_n - (\boldsymbol{I}_n + d_k \boldsymbol{C}_k)^{-1} \right) \\
&= \boldsymbol{U}_k \boldsymbol{W}_k^{-1} \boldsymbol{U}_k^{\mathsf{T}} \boldsymbol{U}_k (\boldsymbol{I}_n - (\boldsymbol{I}_n + d_k \boldsymbol{W}_k)^{-1} \boldsymbol{U}_k^{\mathsf{T}} \\
&= \boldsymbol{U}_k \boldsymbol{W}_k^{-1} [\boldsymbol{I}_n - (\boldsymbol{I}_n + d_k \boldsymbol{W}_k)^{-1}] \boldsymbol{U}_k^{\mathsf{T}} \\
&= \boldsymbol{U}_k [\boldsymbol{W}_k^{-1} - \boldsymbol{W}_k^{-1} (\boldsymbol{I}_n + d_k \boldsymbol{W}_k)^{-1}] \boldsymbol{U}_k^{\mathsf{T}} \\
&= \boldsymbol{U}_k \left[\boldsymbol{W}_k^{-1} - (\boldsymbol{W}_k + d_k \boldsymbol{W}_k^2)^{-1} \right] \boldsymbol{U}_k^{\mathsf{T}} \\
&= \boldsymbol{U}_k (d_k \boldsymbol{W}_k^2) \left(\boldsymbol{W}_k^2 + d_k \boldsymbol{W}_k^3 \right)^{-1} \boldsymbol{U}_k^{\mathsf{T}}.
\end{aligned}$$

This result verifies that $\boldsymbol{T}_k$ is always invertible and the last equality provides an alternative way to store $\boldsymbol{T}_k$ as $\boldsymbol{T}_k^{\frac{1}{2}}\boldsymbol{T}_k^{\frac{1}{2}}$.

C.4. Computational details for log-posterior function

Denote $\boldsymbol{P}_k = \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\phi}_k\boldsymbol{\phi}_k^{\mathsf{T}}\boldsymbol{\Sigma}^{-\frac{1}{2}}$ and $\boldsymbol{Q}_k = \left(d_k\boldsymbol{I}_n + \boldsymbol{C}_k^{-1}\right)^{-1}$. Using identities in Harville [70, ch.16], the quadratic term in the log-posterior Equation (3.21) is

$$\begin{aligned}
&-\frac{1}{2}\mathrm{vec}(\boldsymbol{Y})^{\mathsf{T}}\Big(\boldsymbol{\Sigma}^{-1}\otimes\boldsymbol{I}_n - \sum_{k=1}^{q}\boldsymbol{P}_k\otimes\boldsymbol{Q}_k\Big)\mathrm{vec}(\boldsymbol{Y})\\
=&-\frac{1}{2}\mathrm{tr}(\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y}\boldsymbol{\Sigma}^{-1}) + \frac{1}{2}\sum_{k=1}^{q}\mathrm{tr}(\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Q}_k\boldsymbol{Y}\boldsymbol{P}_k^{\mathsf{T}}),
\end{aligned}$$

preventing the creation of $np\times np$ matrices in the evaluation of the log-posterior function.

ProQuest Number: 30693493

INFORMATION TO ALL USERS
The quality and completeness of this reproduction is dependent on the quality
and completeness of the copy made available to ProQuest.



Distributed by ProQuest LLC (2023).
Copyright of the Dissertation is held by the Author unless otherwise noted.

This work may be used in accordance with the terms of the Creative Commons license
or other rights statement, as indicated in the copyright statement or in the metadata
associated with this work. Unless otherwise specified in the copyright statement
or the metadata, all rights are reserved by the copyright holder.

This work is protected against unauthorized copying under Title 17,
United States Code and other applicable copyright laws.

Microform Edition where available © ProQuest LLC. No reproduction or digitization
of the Microform Edition is authorized without permission of ProQuest LLC.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346 USA